

Kalle Braun · Jürgen Buchmüller · Frank Seger

Das  
**Colour-Genie**  
Buch 1

**TCSA**  
COMPUTER GMBH

39,-

## **Das Colour-Genie Buch 1**

© 1983 TCS Computer GmbH  
ISBN 3-88965-003-1

Alle Rechte vorbehalten, insbesondere auch diejenigen aus der spezifischen Gestaltung, Anordnung und Einteilung des angebotenen Stoffes. Der auszugsweise oder teilweise Nachdruck sowie fotomechanische Wiedergabe oder Übertragung auf Datenträger zur Weiterverarbeitung ist untersagt und wird als Verstoß gegen das Urheberrechtsgesetz und als Verstoß gegen das Gesetz gegen den unlauteren Wettbewerb gerichtlich verfolgt. Für etwaige technische Fehler, sowie für die Richtigkeit aller in diesem Buch gemachten Angaben, übernehmen der Herausgeber und Autor keine Haftung.

---

## Vorwort

Lieber Colour-Genie Freund!

Das Colour-Genie erfreut sich immer größerer Beliebtheit. So ist es auch nicht verwunderlich, daß immer wieder der Ruf nach einem Buch laut wurde, das speziell für das Colour-Genie geschrieben wurde und das auf die vielfältigen Möglichkeiten des Colour-Genies eingeht.

Nun, das Warten hat ein Ende: Vor Ihnen liegt das "Colour-Genie Buch 1" !

Ein Buch, auf das wir stolz sind, denn es ist vollgepackt mit interessanten Programmen und Informationen und es bietet sowohl dem Anfänger als auch dem fortgeschrittenen Programmierer, sowohl dem Spielefreund als auch dem "ernsthaften" Anwender Programmlistings und Erklärungen, die helfen, das Colour-Genie besser zu verstehen und besser zu nutzen.

Schauen Sie ruhig einmal in das Inhaltsverzeichnis auf den nächsten beiden Seiten, blättern Sie einmal durch das Buch und Sie werden sehen, wie vielfältig der Inhalt ist.

Wir hoffen, daß Sie Ihr Colour-Genie so vielseitig einsetzen, wie es seinen Möglichkeiten entspricht und daß Sie viel Freude mit Ihrem Computer haben werden.

In diesem Sinne wünschen wir Ihnen nun viel Spaß beim Lesen, Abtippen und Ausprobieren!

Kalle Braun  
Jürgen Buchmüller  
Frank Seger

Bonn, im August 1983

## Inhaltsverzeichnis

Zur Gliederung .....	Seite	4
Bumm - Bumm .....	Seite	5
Türme von Hanoi .....	Seite	12
Schiffe versenken .....	Seite	15
Hektik - Ein komplexes Spielprogramm analysiert..	Seite	21
Einführung .....	Seite	21
Programmlisting .....	Seite	21
Die Zeichen von Hektik .....	Seite	28
Erläuterung des Programms .....	Seite	29
Bubble Sort .....	Seite	38
Tilgungsplan .....	Seite	40
Gleichungen mit 3 Unbekannten .....	Seite	42
Pascalsches Dreieck .....	Seite	43
Garbage Collect .....	Seite	44
Ein Maschinensprache-Monitor in Basic .....	Seite	46
Einführung .....	Seite	46
Programmlisting .....	Seite	46
Die Monitor-Befehle .....	Seite	51
Erklärung des Programms .....	Seite	53
Die Maschinenroutine zum Bänderschreiben ...	Seite	58
Erklärung obiger Routine .....	Seite	59
Das Format von SYSTEM-Bändern .....	Seite	60
Das Format von CLOAD-Bändern .....	Seite	61
Die Begriffe LSB und MSB .....	Seite	62
Wie werden Basic-Programme abgespeichert?.....	Seite	63
Wie werden Basic-Variablen abgespeichert?.....	Seite	69
Zusammenladen von mehreren Basic-Programmen .....	Seite	72

Reserviert .....	Seite	73
Ausgabe der Bytes 0, 11 oder 12 an den Drucker ..	Seite	74
Ausgabe von Tabulatoren > 40 auf den Drucker ....	Seite	76
Interessante ROM-CALLs .....	Seite	77
Neuer Zeichensatz .....	Seite	79
Ein Screen-Printer .....	Seite	83
Anhang A: Basic-Tokens .....	Seite	91
Anhang B: Dezimal-Hexadezimal Tabelle .....	Seite	93
Anhang C: Softwareliste .....	Seite	95

## Zur Gliederung

Wie Sie vielleicht schon aus dem Inhaltsverzeichnis ersehen konnten, haben wir versucht, die einzelnen Teile dieses Buches in eine sinnvolle Reihenfolge zu bringen.

Zu Beginn finden Sie drei interessante Spielprogramme, die alle bereits auf dem 16K-Grundgerät laufen.

Es folgen dann die Listings zweier Spielprogramme, die mit dem "Colour-Compiler" compiliert werden müssen.

Nach den Spielen folgen mehrere Programme und Erläuterungen, die Ihnen ein tieferes Verständnis für Ihren Computer erleichtern sollen, unter anderem ~~einen~~ kompletten Maschinensprache-Monitor in Basic, der es Ihnen ermöglicht, über die Ebenen der Basicprogrammierung auf die Ebene der Maschinenprogrammierung vorzustoßen.

Etliche Kapitel befassen sich mit der Programmierung eines angeschlossenen Druckers - besondere Freude werden Besitzer des STAR-Printers DP 510/515 an den abgedruckten Programmen haben.

Im Anhang finden Sie dann noch eine Liste der Basic-Tokens, eine Dezimal-Hexadezimal Tabelle, sowie eine Liste der verfügbaren TCS-Colour-Genie Software.

### \*\*\*\* ACHTUNG \*\*\*\*

In den Listings entspricht das Paragrafenzeichen (§) dem "Klammeraffen" (@), der rechts neben der <P>-Taste liegt, und das große Ä (Ä) dem Aufwärtspfeil ([), der zur Exponentialrechnung benötigt wird.

## BUMM BUMM

"Bumm Bumm" ist ein einfaches, jedoch sehr lustiges Telespiel. Zwei Spieler spielen gegeneinander. Jeder Spieler hat eine Kanone und muß nun versuchen, seinen Gegner abzuschießen. Dazu braucht man ein gewisses Gefühl für Ballistik, denn ein Schuß wird durch die Eingabe des Abschußwinkels und der Abschußgeschwindigkeit gesteuert. Das Programm soll hier nicht komplett erklärt werden - einige interessante Teile seien jedoch erläutert:

In Zeile 20 wird in das CRTC-Register für Zeichen-End-Raster eine 0 gePOKEt. Dies bewirkt, daß die FGR-Darstellung nur in halber Höhe erfolgt. Dabei wird der FGR-Speicher zweimal untereinander dargestellt. Diese doppelte Darstellung ist im übrigen leider nicht vermeidbar. In Zeile 690 wird diese CRTC-Programmierung durch POKE 17158,1 wieder rückgängig gemacht. Siehe hierzu auch Anhang A des Handbuchs "Colour Basic - leicht gelernt".

Erwähnenswert ist auch, wie der Titel "BUMM BUMM" im FGR-Modus gezeichnet wird. Dabei werden nicht alle Zeichen über einzelne PLOTs erzeugt, sondern alle Zeichen sind in einer 5 mal 8 Matrix codiert. Die Werte dieser Codierung stehen in den DATA-Zeilen 130 bis 160.

Ausgelesen werden die Werte in den Zeilen 330 bis 430. Die wichtigste Zeile in dieser Routine ist Zeile 380. Hier wird geprüft, ob das Bit, angegeben durch Variable B, in dem aus der DATA-Zeile gelesenen Wert A gesetzt ist oder nicht. Dazu wird mit den Zweierpotenzen 1 bis 16 verglichen, die in A(0) bis A(4) gespeichert sind (siehe Zeilen 60 bis 100).

Analog sind in den DATA-Zeilen 170 bis 260 die Werte für die Ziffern 0 bis 9 gespeichert, die in der Winkel- u. Geschwindigkeitseingabe benötigt werden.

Für Patrioten sei hier noch erklärt, wie das Deutschlandlied am Ende des Programms erzeugt wird. Nötig hierzu sind die DATA-Zeilen 110 bis 120 und der Programmteil von Zeile 2510 bis 2530 und weiter von 2580 bis 2610. Ein Sprung auf Zeile 2510 spielt das Deutschlandlied ab. Zeile 2540 bis 2570 dienen nur dazu, den Namen des Siegers in allen Farben leuchten zu lassen, darum können diese Zeilen weggelassen werden.



```

10 FCLS
20 POKE17158,0
30 FGR
40 DEFINT A-C,E-V
50 DIM Z(9,4)
60 A=128
70 FOR L=7 TO 0 STEP -1
80 A(L)=A
90 A=A/2
100 NEXT L
110 DATA 3,1,3,1,3,1,3,2,3,3,3,3,3,2,3,2,3,4,3,4,3,3,3,3,3,2,2,7,3,1,3,
1,3,6,3,6,3,5,3,5,3,4,3,4,3,3,3,3,3,2,3,2,3,3,3,1,3,5,3,5,3,5,3,5
120 DATA 4,1,4,1,4,1,3,7,3,7,3,6,3,5,3,5,3,6,3,6,3,6,3,5,3,5,3,4,3,3,3,
3,3,2,3,2,3,3,3,4,3,5,3,6,3,4,3,2,3,1,3,1,3,3,3,2,3,1,3,1,3,1
130 DATA 30,17,17,30,17,17,30,0
140 DATA 17,17,17,17,17,17,14,0
150 DATA 17,27,21,21,17,17,17,0
160 DATA 17,27,21,21,17,17,17,0
170 DATA 21,17,17,17,21
180 DATA 1,1,1,1,1
190 DATA 21,1,21,16,21
200 DATA 21,1,5,1,21
210 DATA 17,17,21,1,1
220 DATA 21,16,21,1,1,21
230 DATA 16,16,21,17,21
240 DATA 21,1,1,1,1
250 DATA 21,17,21,17,21
260 DATA 21,17,21,1,1
270 FCLS
280 FGR
290 RESTORE
300 FOR L=1 TO 63
310 READ L1,L2
320 NEXT L
330 FOR C=0 TO 3
340 FOR D=0 TO 7
350 READ A
360 FOR B=4 TO 0 STEP -1
370 X=28+C*38-B*5+D
380 IF (A AND A(B)) <> A(B) THEN 430
390 Y=D*12
400 FCOLOUR 4
410 PLOT X,Y TO X+5,Y TO X+5,Y+12 TO X,Y+12 TO X,Y
420 PAINT X+1,Y+1,3,4
430 NEXT B,D,C
440 FOR L=0 TO 9
450 FOR M=0 TO 4
460 READ Z(L,M)
470 NEXT M,L
480 CLS
490 COLOUR 4
500 LGR

```

```

510 PRINT"Bei diesem Spiel, dass 2 Spieler gegen-"
520 PRINT"einander spielen, geht es darum, die"
530 PRINT"gegnerische Kanone abzuschiessen."
540 PRINT"Dazu muss jeweils der Abschusswinkel"
550 PRINT"und die Abschussgeschwindigkeit der"
560 PRINT"Kanonenkugel eingegeben werden."
570 PRINT
580 PRINT"Gewonnen hat derjenige, der als erster"
590 PRINT"5 Treffer hat."
600 PRINT
610 PRINT"Reihenfolge der Eingaben:"
620 PRINT"Winkel: 0 bis 90 Grad"
630 PRINT"Geschwindigkeit: 0 bis 99 Meter/sec."
640 PRINT
650 PRINT
660 PRINT
670 PRINT"Geben Sie jetzt bitte Ihre Namen ein:"
680 PRINT
690 POKE17158,1
700 COLOUR5
710 INPUT"NAME DES 1. KANONIERS";C$
720 COLOUR2
730 INPUT"NAME DES 2. KANONIERS";B$
740 IFC$=B$THEN700
750 WF=RND(2)
760 IFS1>4ORS2>4THEN2290
770 GOSUB1710
780 P=18440
790 Z=S1
800 GOSUB2090
810 P=18470
820 Z=S2
830 GOSUB2090
840 IFWF=2THEN1210
850 P=18434
860 GOSUB2180
870 W1=A*10
880 P=P+1
890 GOSUB2180
900 W1=W1+A
910 P=18437
920 GOSUB2180
930 G1=A*10
940 P=P+1
950 GOSUB2180
960 G1=G1+A
970 GOSUB1430
980 XP=P1+11
990 YP=Q1-8
1000 GOSUB1570

```

```

1010 IFV<2THENWF=0:GOSUB2350:GOTO1210
1020 IFX0<80THEN1400
1030 S1=S1+1
1040 WF=1
1050 SOUND7,247
1060 SOUND6,31
1070 SOUND8,16
1080 SOUND9,0
1090 SOUND12,150
1100 SOUND13,9
1110 FORA=1TO100
1120 FCOLOURAND(3)+1
1130 CIRCLEX0,Y0,AND7
1140 SOUND6,AND15
1150 NEXTA
1160 FCOLOUR1
1170 FORA=0TO8
1180 CIRCLEX0,Y0,A
1190 NEXTA
1200 GOTO760
1210 P=18464
1220 GOSUB2180
1230 W1=A*10
1240 P=P+1
1250 GOSUB2180
1260 W1=W1+A
1270 P=18467
1280 GOSUB2180
1290 G1=A*10
1300 P=P+1
1310 GOSUB2180
1320 G1=G1+A
1330 GOSUB1430
1340 DX=-DX
1350 XP=P2-1
1360 YP=Q2-8
1370 GOSUB1570
1380 IFV<2THENWF=0:GOSUB2350:GOTO850
1390 IFX0>80THEN1030
1400 S2=S2+1
1410 WF=2
1420 GOTO1050
1430 W1=W1/57
1440 SOUND7,247
1450 SOUND8,10
1460 SOUND9,10
1470 FORL=31TO0STEP-1
1480 SOUND6,L
1490 FORA=1TO4
1500 NEXT

```

```

1510 NEXT
1520 SOUND7,252
1530 SOUND1,1
1540 DX=G1/10*COS(W1)
1550 DY=-G1/10*SIN(W1)
1560 RETURN
1570 SOUND0,(YPAND255)
1580 SOUND2,255-(YPAND255)
1590 IFXP<OORYP<0THEN1640
1600 V=CPOINT(XP, YP)
1610 PLOTXP,YP
1620 XO=XP
1630 YO=YP
1640 XP=XP+DX
1650 YP=YP+DY
1660 DY=DY+.1
1670 NPLOTXO,YO
1680 IFXP<OORYP>95ORXP>159THENRETURN
1690 IFV=0THEN1570
1700 RETURN
1710 FCLS
1720 FGR
1730 FCOLOUR2
1740 P1=RND(40)
1750 P2=RND(40)+110
1760 Y=RND(10)+50
1770 Y=Y*5
1780 Y1=RND(20)-10
1790 FORX=1TO159
1800 PLOTX,Y/5TOX,96
1810 IF(X-P1)<10IF(X-P1)>=0Q1=Y/5:GOTO1840
1820 IF(X-P2)<10IF(X-P2)>=0Q2=Y/5:GOTO1840
1830 IFY+Y1>150IF Y+Y1<480Y=Y+Y1
1840 IF RND(100)<10Y1=RND(20)-10
1850 NEXTX
1860 X=P1
1870 Y=Q1
1880 GOSUB1930
1890 X=P2
1900 Y=Q2
1910 GOSUB2010
1920 RETURN
1930 FCOLOUR3
1940 FORZ=0TO6
1950 PLOTX+Z,Y-ZTOX+6,Y-Z
1960 NEXTZ
1970 FCOLOUR4
1980 PLOTX+7,Y-4TOX+10,Y-7
1990 PLOTX+7,Y-5TOX+10,Y-7
2000 RETURN

```

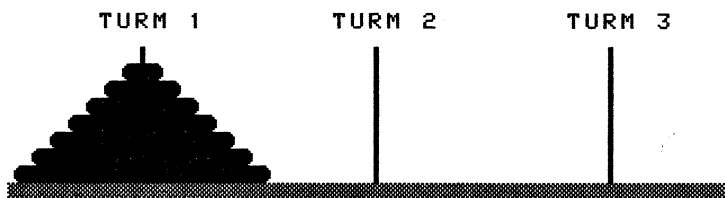
```

2010 FCOLOUR4
2020 FORZ=0TO6
2030 PLOTX+10-Z,Y-ZTOX+4,Y-Z
2040 NEXTZ
2050 FCOLOUR3
2060 PLOTX+3,Y-4TOX,Y-7
2070 PLOTX+3,Y-5TOX,Y-7
2080 RETURN
2090 A=INT(Z/10)
2100 FORY=0TO4
2110 POKEP+Y*40,Z(A,Y)
2120 NEXTY
2130 A=Z-A*10
2140 FORY=0TO4
2150 POKEP+Y*40+1,Z(A,Y)
2160 NEXTY
2170 RETURN
2180 FORY=0TO160STEP40
2190 POKEP+Y,42
2200 A$=INKEY$
2210 IF A$="0" AND A$<="9" THEN 2240
2220 NEXTY
2230 GOTO 2180
2240 A=VAL(A$)
2250 FORY=0TO4
2260 POKEP+Y*40,Z(A,Y)
2270 NEXTY
2280 RETURN
2290 LGR
2300 CLS
2310 IFS1>4 THEN S$=C$ ELSE S$=B$
2320 PRINT S$;" HAT GEWONNEN"
2330 GOSUB 2500
2340 RUN
2350 SOUND 7,247
2360 SOUND 8,15
2370 SOUND 9,0
2380 FOR L=0TO31
2390 FCOLOUR RAND(3)+1
2400 CIRCLE XO,YO, LAND 3
2410 SOUND 6, LAND 7
2420 NEXT L
2430 FCOLOUR 1
2440 FOR L=0TO3
2450 CIRCLE XO,YO,L
2460 NEXT L
2470 SOUND 7,255
2480 FCOLOUR 3
2490 RETURN
2500 RESTORE

```

```
2510 FORL=1TO63
2520 READO,T
2530 PLAY(1,0,T,14)
2540 F=RND(16)
2550 FORM=&HF000TO&HF000+LEN(S$)
2560 POKEM,F
2570 NEXTM
2580 FORM=1TO100-LEN(S$)
2590 NEXTM
2600 NEXTL
2610 PLAY(1,1,1,0)
2620 RETURN
```

# →→→ TÜRME VON HANOI ←←←



Von Turm?

Spielzug Nr. 0

Türme von Hanoi ist ein bekanntes Denkspiel. Die Regeln werden im Programm selbst erklärt.  
Eine Besonderheit für Denkfaule: Das Programm löst das Problem auf Wunsch auch selbst - und zwar mit der Minimalanzahl von 127 Zügen. Die Rechnerlösung ist in den Zeilen 20 und 30 enthalten - passen Sie hier besonders auf, daß Sie sich nicht vertippen.  
Programmlisting:

```

10 CLS: CLEAR 500: O=0: A=7: CHAR4
20 RD$="12132312313212132321312312132312313212313221313212132312313212
132321312312132321313212312321312312132312313212132321312312132312"
30 Z2$="31321231232131321213231231321231232131231213232131321231232131
3212132312313212132321312312132312313212312321313212132312313212"
40 RD=1
50 PRINT "Die Scheiben des linken Turms muessen"
60 PRINT "in der gleichen Reihenfolge auf der"
70 PRINT "mittleren Scheibe liegen!!"
80 PRINT "Dabei darf niemals eine grosse Scheibe"
90 PRINT "auf einer kleineren liegen!!"
100 PRINT$720,"1 - Rechner spielt": PRINT$760,"2 - Sie spielen"
110 X1$=INKEY$: IF X1$<>"1" AND X1$<>"2" THEN 110
120 IF X1$="1" THEN 160
130 FKEY1="1"
140 FKEY2="2"
150 FKEY3="3"
160 DIM B$(3,8), T(3,8), F(3,7): C1=1: O=0
170 POKE 16409,70: POKE 16410,100
180 REM
190 REM FARBEN
200 REM
210 FOR J=1 TO 7
220 READ F(1,J)

```

```

230 NEXT
240 REM
250 REM POSITIONEN
260 REM
270 FOR C=1 TO 3
280 FOR K=7 TO 1 STEP -1
290 READ T(C,K)
300 NEXT K
310 NEXT C
320 REM
330 REM SCHEIBEN-STRING'S ERSTELLEN
340 REM
350 FOR C=1 TO A
360 B$(1,C)=CHR$(238)+STRING$(C1,202)+CHR$(245)
370 C1=C1+2
380 NEXT C
390 CLS
400 GOSUB 640
410 REM
420 REM SCHEIBEN SETZEN
430 REM
440 FOR B=1 TO A
450 COLOUR F(1,B)
460 PRINT$647-A*40+B*40-B,B$(1,B);
470 NEXT:PRINT:PRINT
480 REM
490 REM SPIELEN
500 REM
510 COLOUR 1
520 PRINT$848,"Spielzug Nr."0;
530 IFX1$="1"THENGOSUB1280:GOTO600
540 PRINT$768,STRING$(4,32);
550 PRINT$760,"Von Turm";:INPUTE
560 IFE<10RE>3THEN550
570 PRINT$785,STRING$(4,32);
580 PRINT$775,"nach Turm";:INPUTF
590 IFF<10RF>3THEN580
600 O=O+1:GOTO 860
610 REM
620 REM TUERME AUFBAUEN
630 REM
640 COLOUR1:FOR B=1 TO A+1
650 PRINT$687-B*40,CHR$(225);
660 PRINT$700-B*40,CHR$(225);
670 PRINT$713-B*40,CHR$(225);
680 NEXT
690 PRINT$680,STRING$(40,196);
700 PRINT$285,"TURM 1";:PRINT$298,"TURM 2";:PRINT$311,"TURM 3";
710 PRINT$88,STRING$(3," ");" TUERME VON HANOI ";STRING$(3," ");
720 RETURN
730 REM
740 REM FARBEN FUER SCHEIBEN
750 REM
760 DATA 14,7,12,9,6,15,3
770 REM
780 REM DATAS FUER POSITIONEN
790 REM

```



```

800 DATA 647,607,567,527,487,447,407
810 DATA 660,620,580,540,500,460,420
820 DATA 673,633,593,553,513,473,433
830 REM
840 REM SUBROUTINE FUER SCHEIBEN-LOESCHEN
850 REM
860 FOR V=1 TO A
870 IF B$(E,V) > "" THEN 890
880 NEXT V
890 FOR V1=1 TO A
900 IF B$(F,V1) > "" THEN 910 ELSE NEXT V1
910 V1=V1-1
920 IF V=8 THEN 1190
930 IF V1<7 AND B$(E,V) < B$(F,V1+1) THEN 1140
940 F(F,V1)=F(E,V)
950 FOR C=1 TO INT(LEN(B$(E,V)))/2+1
960 PRINT$(E,V)-INT(LEN(B$(E,V)))/2+C-2,CHR$(32);
970 PRINT$(E,V)-INT(LEN(B$(E,V)))/2+LEN(B$(E,V))-C+1,CHR$(32);
980 NEXT:COLOUR1
990 PRINT$(E,V),CHR$(225);
1000 REM
1010 REM SUBROUTINE FUER SCHEIBEN-SETZEN
1020 REM
1030 COLOUR F(F,V1)
1040 PRINT$(F,V1)-INT(LEN(B$(E,V)))/2,CHR$(238);
1050 PRINT$(F,V1)+INT(LEN(B$(E,V)))/2,CHR$(245);
1060 FOR C=1 TO INT(LEN(B$(E,V)))/2
1070 PRINT$(F,V1)-INT(LEN(B$(E,V)))/2+C,CHR$(202);
1080 PRINT$(F,V1)-INT(LEN(B$(E,V)))/2 +LEN(B$(E,V))-C-1,CHR$(2
02);
1090 NEXT
1100 IF F=2 AND V1=1 THEN 1250
1110 B$(F,V1)=B$(E,V):B$(E,V)=""
1120 REMT1(E)=T1(E)+1:T1(F)=T1(F)-1
1130 GOTO 510
1140 FOR Q=1 TO 5
1150 COLOUR 4:PRINT$0,"Keine grosse Scheibe auf eine kleinere !";
1160 FOR Q1=1 TO 70:NEXT Q1
1170 PRINT$0,STRING$(40,128);
1180 NEXT Q:GOTO 510
1190 FOR Q=1 TO 6
1200 COLOUR 4:PRINT$0,"Turm"E"ist leer !";
1210 FOR Q1=1 TO 70:NEXT Q1
1220 PRINT$0,STRING$(17,128);
1230 NEXT Q:GOTO 510
1240 FKEY1="LIST":FKEY2="RUN":FKEY3="AUTO"
1250 IF O > 140 THEN 1270
1260 CLS:PRINT$440,"BRAVO-Mit"O"Zuegen geschafft !":PRINT:PRINT:END
1270 CLS:PRINT$440,"Mit"O"Zuegen geschafft-da fehlt die Uebung-wei
termachen !":PRINT:PRINT:END
1280 IFRD>128THENRD$=Z2$:RD=1
1290 E=VAL(MID$(RD$,RD,1)):F=VAL(MID$(RD$,RD+1,1)):RD=RD+2:RETURN

```

## Schiffe versenken

Das im folgenden gelistete Programm ist eine Abwandlung des hinreichend bekannten Spiels "Schiffe versenken". Dieses Programm hat zwar nicht den höchsten Schwierigkeitsgrad, aber die Ausgabe des Feldes wird mit einigen grafischen Tricks gemacht. Am auffälligsten ist die Tatsache, das die Wellen tatsächlich rollen. Hier einige Worte dazu:

In den Zeilen 1210 bis 1260 wird in dem Bereich von 409FH bis 40EOH ein kleines Maschinenspracheprogramm initialisiert, das bei jedem 16. Aufruf den gesamten Bildschirm nach Zeichen zwischen 128 und 159 absucht und diese um 1 erhöht. Durch CALL 43D3 wird diese Routine in den Tastatur-DCB eingehängt (siehe Kapitel über die DCBs). Bei jeder 16. Tastaturabfrage werden nun die Wellen bewegt. Mit CALL 43DA wird dies wieder abgeschaltet.

Noch einige Worte zum Spielablauf:

Zunächst stellt der Rechner seine Schiffe auf. Dies sind:

- 1 Schlachtschiff ( 4 Kästchen lang )
- 2 Zerstörer ( 3 Kästchen lang )
- 3 Kreuzer ( 2 Kästchen lang ) und
- 4 Aufklärer ( 1 Kästchen lang )

Sie können nun das rote Fadenkreuz, das oben links auf dem Wasser erscheint, mit den 4 Pfeiltasten bewegen. Wenn Sie eine Position erreicht haben, auf die Sie schießen wollen, drücken Sie <RETURN>. Der Rechner teilt Ihnen dann unten rechts mit ob Sie getroffen oder nur ins Wasser geschossen haben. Außerdem hört man bei einem Treffer eine Explosion und der zerschossene Teil eines Schiffes erscheint. Alle Schiffe liegen waagrecht, mit dem Bug nach rechts. Außerdem sind Sie an Ihrer Farbe erkennbar. In der untersten Bildschirmzeile sehen Sie ständig eine Auswertung Ihrer bisherigen "Arbeit": die Anzahl der abgegebenen Schüsse und die Anzahl der Treffer werden dort angezeigt.

Wenn Sie alle Schiffe zerstört haben, erscheint noch eine Auswertung, wie gut (oder wie schlecht) Sie gespielt haben.

Programmlisting:

```

10 CLS
20 DEFINT A-Z
30 DIM F1(10,10),F(10,10),R(4)
40 GOSUB 700
50 GOTO 1280
60 DATA 0.8640,30.4620
70 DATA 16416.8608,1054.4618
80 DATA 0.18480,135.-31613
90 DATA 4104.18472,391.-31742
100 DATA 0.4620,16609.8608
110 DATA 1026.4618,225.8640
120 DATA 128.-31613,4216.18472
130 DATA 256.-31742,120.18480
140 DATA 16416.8608,1054.4618
150 DATA 0.8640,30.4620
160 DATA 4104.18472,391.-31742
170 DATA 0.18480,135.-31613
180 DATA 1026.4618,225.8640
190 DATA 0.4620,16609.8608
200 DATA 256.-31742,120.18480
210 DATA 128.-31613,4216.18472
220 DATA 225.16416,8865.28
230 DATA 225.0.17280.60
240 DATA 120.4104.-14296.7
250 DATA 120.0.-28576.15
260 DATA 30.1026,12810.193
270 DATA 30.0.9240.195
280 DATA 135.256,3074.240
290 DATA 135.128.-30330.112
300 DATA 225.0.17280.60
310 DATA 225.16416,8865.28
320 DATA 120.0.-28576.15
330 DATA 120.4104.-14296.7
340 DATA 30.0.9240.195
350 DATA 30.1026,12810.193
360 DATA 135.128.-30330.112
370 DATA 135.256,3074.240
380 DATA 1024.1540,1031.1285
390 DATA 0.0.4336.-32544
400 DATA 0.2056,2056.4127
410 DATA 0.0.0.16512
420 DATA 256.7175.13073.25894
430 DATA -16384.6256.-6524.-11598
440 DATA 1024.1540,1286.1285
450 DATA 0.0.-32768.-32672
460 DATA 0.0.0.4127
470 DATA 0.0.0.16512
480 DATA 0.0.0.0
490 DATA 0.0.0.0
500 DATA 0.0.0.0
510 DATA 0.0.0.0
520 DATA 0.0.0.0
530 DATA 0.0.0.0
540 DATA -251.-1.-103.-1

```

```

550 DATA-128,-1,-103,-1
560 DATA-244,-1,-104,-1
570 DATA-224,-12056,16544,128
580 DATA25931,13094,7184,7
590 DATA-11566,-6606,6156,240
600 DATA-251,-15393,-14463,-1
610 DATA-128,-30745,-28799,-8273
620 DATA-4852,25065,-4208,-13
630 DATA-16608,-28504,16544,128
640 DATA0,0,0,0
650 DATA0,0,0,0
660 DATA6407,16673,-32447,-127
670 DATA12480,1032,516,-254
680 DATA-32383,16705,6433,7
690 DATA514,1028,12296,192
700 PRINT"Ich denke mir eine Aufstellung aus!"
710 FORX=0TO511STEP2
720 READA
730 POKEX-3072,AAND255
740 POKEX-3071,INT(A/256)AND255
750 NEXTX
760 GOSUB1210
770 FORX=1TO10
780 FORY=1TO10
790 F(X,Y)=0
800 F1(X,Y)=0
810 NEXTY,X
820 FORZ=1TO10
830 READL
840 FORX=1TO4
850 R(X)=0
860 NEXTX
870 SX=RND(10)
880 SY=RND(10)
890 FORX=1TO2
900 IFR(X)<>0THENNEXTX:GOTO840
910 R=RND(2)
920 IFR(R)<>0THEN910
930 R(R)=1
940 TX=SX
950 TY=SY
960 FORX=1TOL
970 IFF(TX,TY)<>0THEN890
980 IFTX-1>0THENIFF(TX-1,TY)<>0THEN890
990 IFTX+1<11THENIFF(TX+1,TY)<>0THEN890
1000 IFTY-1>0THENIFF(TX,TY-1)<>0THEN890
1010 IFTY+1<11THENIFF(TX,TY+1)<>0THEN890
1020 ONRGOSUB1130,1150,1170,1190
1030 IFTX<10RTX>10ORTY<10RTY>10THEN890
1040 NEXTX
1050 TX=SX
1060 TY=SY
1070 FORX=1TOL
1080 F(TX,TY)=L

```

```

1090 ONRGOSUB1130,1150,1170,1190
1100 NEXTX
1110 NEXTZ
1120 RETURN
1130 TX=TX+1
1140 RETURN
1150 TX=TX-1
1160 RETURN
1170 TY=TY+1
1180 RETURN
1190 TY=TY-1
1200 RETURN
1210 FORX=1TO66
1220 READA
1230 POKE&H439E+X,A
1240 NEXT
1250 RETURN
1260 DATA58,158,67,60,230,15,50,158,67,194,227,3,229,197,33,0,68,126,2
54,128,56,19,254,160,48,15,60,60,71,230,15,254,2,62,0,48,2,62,240,128,
119,35,124,254,72,32,226,193,225,195,227,3,33,159,67,34,22,64,201,33,2
27,3,34,22,64,201
1270 DATA4,3,3,2,2,2,1,1,1,1
1280 COLOUR1
1290 CLS
1300 PRINT$7,"** SCHIFFE VERSENKEN **"
1310 TR=0
1320 SH=0
1330 H1$=CHR$(160)+CHR$(161)
1340 H2$=CHR$(176)+CHR$(177)
1350 B1$=CHR$(162)+CHR$(163)
1360 B2$=CHR$(178)+CHR$(179)
1370 PRINT$40,STRING$(39,217);
1380 PRINT$64,CHR$(242);
1390 FORX=2TO23
1400 PRINT$X*40+24,CHR$(225);
1410 NEXT
1420 COLOUR3
1430 PRINT$146,"Schlachtschiff";
1440 PRINT$186,H1$;H1$;H1$;B1$;
1450 PRINT$226,H2$;H2$;H2$;B2$
1460 COLOUR1
1470 PRINT$306,"Zerstoeerer";
1480 PRINT$346,H1$;H1$;B1$;
1490 PRINT$386,H2$;H2$;B2$;
1500 COLOUR4
1510 PRINT$466,"Kreuzer";
1520 PRINT$506,H1$;B1$;
1530 PRINT$546,H2$;B2$;
1540 COLOUR2
1550 PRINT$626,"Aufklaerer";
1560 PRINT$666,B1$;
1570 PRINT$706,B2$;
1580 COLOUR7
1590 PRINT$786,"Wasser";
1600 CALL43DA
1610 PRINT$826,;
1620 FORX=1TO7

```

```

1630 PRINTCHR$(128);CHR$(129);
1640 NEXTX
1650 PRINT8866.;
1660 FORX=1TO7
1670 PRINTCHR$(144);CHR$(145);
1680 NEXTX
1690 FORX=1TO10
1700 FORY=1TO10
1710 AD=&H4402+Y*80+2*X-2
1720 POKEAD-21504,1
1730 POKEAD-21503,1
1740 POKEAD-21464,1
1750 POKEAD-21463,1
1760 POKEAD,128
1770 POKEAD+1,129
1780 POKEAD+40,144
1790 POKEAD+41,145
1800 NEXTY,X
1810 CALL43D3
1820 X=1
1830 Y=1
1840 PRINT8920,SH;"Schuesse,";TR;"Treffer";
1850 CU=&H4400+Y*80+2*X
1860 POKECU-21504,2
1870 POKECU-21503,2
1880 POKECU-21464,2
1890 POKECU-21463,2
1900 POKECU,188
1910 POKECU+1,189
1920 POKECU+40,190
1930 POKECU+41,191
1940 KB$=INKEY$
1950 IFKB$=""THEN1940
1960 IFF1(X,Y)<>0THENBF=0:GOSUB2140:GOTO2080
1970 CALL43DA
1980 CH=PEEK(&H4400+826)
1990 POKECU-21504,1
2000 POKECU-21503,1
2010 POKECU-21464,1
2020 POKECU-21463,1
2030 POKECU,CH
2040 POKECU+1,CH+1
2050 POKECU+40,CH+16
2060 POKECU+41,CH+17
2070 CALL43D3
2080 IFKB$=CHR$(13)THEN2480
2090 IFKB$="Ä"THENIFY>1THENY=Y-1
2100 IFKB$=CHR$(10)THENIFY<10THENY=Y+1
2110 IFKB$=CHR$(8)THENIFX>1THENX=X-1
2120 IFKB$=CHR$(9)THENIFX<10THENX=X+1
2130 GOTO1840
2140 IFX=10THENCH=168:GOTO2170
2150 IFF(X+1,Y)=0THENCH=168:GOTO2170
2160 CH=166

```

```

2170 AD=&H4400+Y*80+2*X
2180 IFBF=0THEN2360
2190 SOUND6,RND(10)+21
2200 SOUND7,7
2210 SOUND8,16
2220 SOUND12,200
2230 SOUND13,9
2240 FORT=1TO10
2250 POKEAD-21504,6
2260 POKEAD-21503,6
2270 POKEAD-21464,6
2280 POKEAD-21463,6
2290 POKEAD,164
2300 POKEAD+1,165
2310 POKEAD+40,180
2320 POKEAD+41,181
2330 GOSUB2360
2340 NEXTT
2350 RETURN
2360 CO=6-F1(X,Y)
2370 POKEAD-21504,CO
2380 POKEAD-21503,CO
2390 POKEAD-21464,CO
2400 POKEAD-21463,CO
2410 POKEAD,CH
2420 POKEAD+1,CH+1
2430 POKEAD+40,CH+16
2440 POKEAD+41,CH+17
2450 RETURN
2460 IFKBAND64=64THENIFX<10THENX=X+1
2470 GOTO1840
2480 IFF1(X,Y)<>0THEN1840
2490 F1(X,Y)=F(X,Y)
2500 IFF1(X,Y)=0THENPRINT$946,"Wasser ";ELSEPRINT$946,"Getroffen";:B
F=1:GOSUB2140:TR=TR+1
2510 SH=SH+1
2520 IFTR<20THEN1840
2530 CALL43DA
2540 CLS
2550 COLOUR1
2560 PRINT"Sie haben mit";SH:"Schuessen alle "
2570 PRINT"Schiffe getroffen!!"
2580 IFSH=20THENPRINT"Super, Sie sind ein hervorragender":PRINT"Admira
l"
2590 IFSH<30THENPRINT"Sie haben gute Chancen, eine See-":PRINT"schlach
t zu gewinnen"
2600 IFSH<50THENPRINT"Sie sollten noch etwas ueben!"
2610 IFSH>49THENPRINT"Auweia, da haben Sie aber viel Pech":PRINT"gehab
t!!!!"
2620 PRINT$960,"Neues Spiel??? (J/N)";
2630 NS$=INKEY$
2640 IFNS$<>"J"ANDNS$<>"N"THEN2630
2650 IFNS$="J"THENRUN
2660 PRINT$400,"Schade"
2670 END

```

## HEKTIK - ein komplexes Spielprogramm analysiert

"Hektik" wird schon seit längerer Zeit für 39.- DM verkauft und gehört zu den beliebtesten TCS-Spielprogrammen. Vielleicht haben Sie es schon, dann werden Sie beim Spielen sicher eine Menge Spass gehabt haben. Wahrscheinlich haben Sie sich auch gefragt, wie man ein derartiges Spiel programmiert. Diese Frage soll nicht unbeantwortet bleiben!

Zwei Voraussetzungen müssen allerdings erfüllt sein:

Ihr Colour-Genie muß 32K RAM haben, und Sie müssen den "Colour-Compiler" besitzen. Denken Sie bitte nicht, daß wir Sie auf diese Weise zum Kauf des Compilers drängen möchten - der Compiler ist für schnelle Videospiel-Programmierung einfach unerlässlich, es sei denn Sie programmieren in Maschinensprache.

Es folgt nun das Basic-Listing (also der "Source-Code") von Hektik. Wichtig ist, daß Sie beim Einschalten die MOD SEL-Taste gedrückt halten, da sonst der verfügbare Speicher nicht ausreicht !

Nach dem Listing finden Sie eine detaillierte Erklärung des Programms. Bezüglich der Bedienung des "Colour-Compilers" informieren Sie sich ggf. in dessen Handbuch.

Das Basicprogramm geben Sie bitte genauso ein, wie Sie es hier aufgelistet finden. Die Grafikzeichen, die im Listing vorkommen, geben Sie, wie in Kap. 6 des Handbuchs "Colour Basic - leicht gelernt" beschrieben, mit der MOD SEL-Taste ein. ✖

### Programmlisting:

```
10 DATA-127,-32383,-127,-32383
20 DATA-1,-6400,231,-1
30 DATA-61,-6400,231,-1
40 DATA-15487,-6400,231,-1
50 DATA-15487,-15616,231,-1
60 DATA-32383,-32512,129,-61
70 DATA-32383,-32512,129,-32383
80 DATA0,0,0,0
90 DATA-32383,-32257,-32383,-32257
100 DATA-6169,-256,255,-6169
110 DATA-15487,-6400,255,-6169
120 DATA-32383,-15616,231,-6169
130 DATA-32383,-32512,195,-6205
140 DATA-32383,-32512,129,-15487
150 DATA-32383,-32512,129,-32383
160 DATA0,0,0,0
```



```

170 DATA17025,-25,-6247,9342
180 DATA17025,-6334,-26113,15462
190 DATA258,511,1,0
200 DATA0,12736,525,2052
210 DATA-32704,-32513,128,0
220 DATA0,-29693,16560,4128
230 DATA-1538,6264,-31492,1924
240 DATA-1538,6264,18472,3624
250 DATA15900,13106,-25538,-26472
260 DATA31800,-13236,14716,6425
270 DATA-26498,6296,10020,-16352
280 DATA7168,14654,7230,6425
290 DATA6526,6169,-7132,772
300 DATA14336,-25476,14460,-26472
310 DATA-24705,6174,8511,-8159
320 DATA-24705,6174,4628,12308
330 FORL=0T0255:POKEL-3072,PEEK(L+18436)
:NEXTL
340 CHAR2
350 DATA 769,5,769,5,1282,5,1282,5,769,4
,769,4,514,512,770,521,514,0
360 FORL=0T031:POKE30720+L,PEEK(L+18692)
:NEXTL
370 POKE16409,7:POKE16410,6
380 CLS
390 E=2:M2=3:R1=1
400 GOSUB2070
410 B2=0:P2=0:Q=0:Q1=0
420 GOSUB2580:COLOUR11:FORY=80T0880STEP1
60:FORX=0T038STEP2
430 POKE17408+X+Y,129:POKE17409+X+Y,137
440 POKEX+Y-4096,6:POKEY+Y-4095,6
450 NEXTX,Y
460 FORX=0T039:POKE18288+X,202:NEXTX
470 FORY=40T0680STEP160
480 A=Y/20+2:B=37-Y/20
490 FORY1=0T0160STEP40
500 POKE17408+Y+Y1+B,128:POKEY+Y1+B-4096
,3
510 POKE17408+Y+Y1+A,128:POKEY+Y1+A-4096
,3
520 NEXTY1,Y
530 PRINT@920,"PUNKTE:          BONUS:
JAEGER:";
540 PRINT@956,M2;
550 IFT=0LETP2=P2+B2
560 SOUND7,56:SOUND8,16:SOUND9,0:SOUND10
,0:SOUND1,2:SOUND3,4:SOUND5,2:SOUND12,10
570 T=0:T2=30719:02=3:B2=R1*500
580 FORL=0T0E
590 POKE31000+L,L+15:POKE31010+L,1
600 POKE31020+L,1
610 POKE31030+L,3
620 POKE31040+L,32
630 NEXTL

```

```

640 X=19:Y=20:H=32:H1=32:H2=32:X2=1:Y2=0
650 A=PEEK(-1984):IFA=660T0380
660 X1=0:Y1=0
670 IF(AAND32)=32LETX1=-1:60T0710
680 IF(AAND64)=64LETX1=+1:60T0710
690 IF(AAND08)=08LETY1=-1:60T0710
700 IF(AAND16)=16LETY1=+1
710 P=17408+X+Y*40
720 IFI1<160T0750
730 IFH1=238LETH1=32ELSEIFH1=245LETH1=32
740 IFH2=238LETH2=32ELSEIFH2=245LETH2=32
750 C=C+1:C1=(C/2)AND1
760 IFH>143IFH<14660T01120
770 IFH1>143IFH1<14660T01120
780 IFB2>0LETB2=B2-1
790 IF(AAND127)<>0POKEP,H:POKEP+40,H1:P0
KEP+40+X2,H2:POKEP-21504,15:POKEP-21464,1
5
800 IFX+X1=0LETX1=0ELSEIFX+X1=39LETX1=0
810 IFX1<>0LETX2=X1:Y2=0
820 IFY1<>0LETY2=Y1:X2=0
830 P1=P+Y1*40:P=P+X1
840 Z=PEEK(P+80):IFZ<>128IFZ<>129IFZ<>13
7IFZ<>135IFZ<>143IFZ<>202LETX1=0
850 IFZ=13560T01990ELSEIFZ=14360T01990
860 IFPEEK(P1+40)<>128LETY1=0
870 X=X+X1:Y=Y+Y1:P=17408+X+Y*40:IF(AAND
127)<>0LETH=PEEK(P):H1=PEEK(P+40):H2=PEEK
(P+40+X2)
880 IFI1<160T0900
890 IFH2=23860SUB1880ELSEIFH2=24560SUB18
80ELSEIFH2=23060SUB1880
900 POKEP-21504,7:POKEP-21464,8
910 IFX2<>0IFH2=32POKEP-21464+X2,0
920 IFX2=-1POKEP,153:POKEP+40,158+C1:POK
EP+39,148
930 IFX2=+1POKEP,152:POKEP+40,150+C1:POK
EP+41,146
940 IFY2<>0POKEP,155+C1*2:POKEP+40,154+C
1*2
950 IF(AAND128)<>12860T01070
960 IFX2=060T01050
970 SOUND8,16:SOUND13,9:SOUND0,0:SOUND1,
4-C1
980 POKEP+40+X2,148-(X2+1)+C1
990 IFC1=060T01070
1000 P=P+80+X2:Z=PEEK(P)
1010 IFPEEK(-1920)=160T01050
1020 IFZ>128IFZ<135POKEP,Z+1:60T01070
1030 IFZ>136IFZ<143POKEP,Z+1:60T01070
1040 60T01070

```

```

1050 IFZ>129IFZ<136POKEP,Z-1:GOTO1070
1060 IFZ>137IFZ<144POKEP,Z-1:GOTO1070
1070 Q1=Q1+1:IFQ1>(6-R1)/2LETQ1=0:GOSUB1
400ELSEFORM=1T0300:NEXTM
1080 IFZ2=1LETZ2=0:GOTO1120
1090 COLOURRND(16):PRINT@14,"** HEKTIK *
*";:PRINT@927,P2;"0";:PRINT@942,B2;"0 ";
1100 IFT-1=ELETT=0:R1=R1+1:IFE<5LETE=E+1
:GOTO420ELSE420
1110 GOTO650
1120 SOUND7,56:SOUND6,0:FORL=8T010:SOUND
L,16:NEXTL:SOUND12,90:SOUND13,9
1130 FORM=1T03100STEP5:SOUND0,M:SOUND2,M
/2:SOUND4,M/4:NEXTM
1140 SOUND7,255
1150 M2=M2-1:IFM2=0GOTO1180
1160 B2=0
1170 GOTO420
1180 IFR2>P2GOTO1200
1190 R2=P2
1200 FORL=0T08
1210 IFPEEK(31108+L*10)+PEEK(31109+L*10)
*256<P2GOTO1240
1220 NEXTL
1230 GOTO380
1240 SOUND7,7:FORM=8T010:SOUNDM,16:NEXTM
1250 SOUND6,0:SOUND12,100:SOUND13,9
1260 CLS
1270 FORM=0T031:COLOURRND(16):PRINT@210,
"N E U E R R E K O R D";
1280 PRINT@250,"_____";
1290 SOUND6,M
1300 NEXTM
1310 FORM=31190T0(31100+L*10)STEP-10:FOR
N=0T09
1320 POKEM+N,PEEK(M+N-10)
1330 NEXTN:NEXTM
1340 PRINT@568,"Geben Sie Ihren Namen ei
n:"
1350 PRINT@697,".....";:PRINT@697,CHR
$(14);
1360 P=31100+L*10:GOSUB2510
1370 POKE31108+L*10,P2AND255
1380 POKE31109+L*10,P2/256
1390 GOTO380
1400 FORL=31000T031000+E
1410 V=PEEK(L):W=PEEK(L+10):Q=PEEK(L+40)
1420 IFV=255GOTO1660
1430 IFQ>145IFQ<160LETZ2=1:RETURN
1440 IFI1<1IFQ=238LETQ=32ELSEIFQ=245LETQ
=32ELSEIFQ=230LETQ=32
1450 V1=PEEK(L+20)-2:W1=PEEK(L+30)-2
1460 POKE17408+V+W*40,Q
1470 POKEV+W*40-4096,15
1480 IFQ>128IFQ<144POKEV+W*40-4096,6

```

```

1490 IFV+V1=0POKEL+20,3ELSEIFV+V1=39POKE
L+20,1
1500 A=PEEK(17448+V+W*40):B=PEEK(17368+V
+W*40)
1510 IFW=1POKEL+30,3ELSEIFW=21POKEL+30,1
1520 IFA=135LETQ=129:GOTO1740ELSEIFA=143
LETQ=137:GOTO1740
1530 IFW1=1IFA=128LETV1=0:GOTO1550ELSEIF
A<>202IFA>145LETV1=0:GOTO1550
1540 IFW1=1LETW1=0:IFQ=128POKEL+20,RND(2
)>OR1
1550 IFW1=-1IFB=128LETV1=0:GOTO1570ELSEI
FB>145LETV1=0:GOTO1570
1560 IFW1=-1LETW1=0:IFQ=128POKEL+20,RND(
2)>OR1
1570 V=V+V1:W=W+W1
1580 POKEL+40,PEEK(17408+V+W*40)
1590 POKE17408+V+W*40,143+RND(2)
1600 POKEV+W*40-4096,5
1610 POKEL,V:POKEL+10,W
1620 IFPEEK(L+40)<144GOTO1660ELSEIFPEEK(
L+40)>145GOTO1660
1630 FORM=31000T031000+E
1640 IFL<>MIFPEEK(M)=VIFPEEK(M+10)=WPOKE
L+40,PEEK(M+40)
1650 NEXTM
1660 NEXTL
1670 T2=T2+1:IFT2=30752LETT2=30720:IF02<
6LET02=02+1ELSE02=3
1680 A=PEEK(T2):PLAY(3,02,A,10):IFA<>0SO
UND8,0:SOUND10,16:SOUND13,9
1690 IFI=1GOTO1840
1700 IFRND(1000)<R160T01720
1710 RETURN
1720 I1=50:I=1
1730 RETURN
1740 IFRND(100)<(R1*10)GOTO1960
1750 SOUND9,15:SOUND3,2
1760 P=V+W*40+17448:Q1=6:FORM=PT018288ST
EP40
1770 POKEM,Q:POKEM-21504,Q1:Q=PEEK(M+40)
:Q1=PEEK(M-21464):POKEM+40,144:POKEM-2146
4,5
1780 SOUND2,(M/3)AND255
1790 FORN=1T0300:NEXTN
1800 NEXTM:POKEM,Q:POKEM-21504,Q1
1810 POKEL,255:T=T+1
1820 SOUND9,0
1830 P2=P2+E*5:GOTO1660
1840 IFI1>0LETI1=I1-1
1850 IFI1=0 PRINT@699," ";I=0:RETURN
1860 IF(I1AND1)=1 PRINT@699,"●";ELSE P
RINT@699,"●●";
1870 RETURN
1880 I=0:PRINT@698,"1000";
1890 B$=CHR$(1)+CHR$(3)+CHR$(5)+CHR$(4)+
CHR$(6)+CHR$(5)+CHR$(5)+CHR$(1)+CHR$(1)
1900 FORL=0T08:C=PEEK(30752+L)
1910 PLAY(3,4,C,14)
1920 FORM=1T0900:NEXTM,L

```

```

1930 PRINT@698,"      ";
1940 H=32:H1=32:H2=32:P2=P2+100:I=0:I1=0
:A=8
1950 RETURN
1960 POKE17448+V+W*40,129+(VAND1)*8
1970 FORM=7T01STEP-1:PLAY(2,4,M,15):FORN
=1T0300:NEXTN:NEXTM:SOUND9,0
1980 GOTO1530
1990 IFH2=128IFPEEK(P-40)<>128LETH2=32
2000 POKEP-X1,H:POKEP+40-X1,H1:POKEP+40-
X1+X2,H2
2010 X=X+X1:Y=Y+4:X2=X1:P=17408+X+Y*40
2020 H=PEEK(P):H1=PEEK(P+40):H2=PEEK(P+4
0+X1)
2030 IFX2=-1POKEP,153:POKEP+40,159:POKEP
+39,148 ELSE POKEP,152:POKEP+40,151:POKEP
+41,146:X2=1
2040 POKEP-21504,7:POKEP-21464,8:IFH2=32
POKEP-21464+X2,0
2050 SOUND9,15:SOUND10,15:SOUND3,2:SOUND
5,2:FORM=0T0255STEP4:SOUND2,M:FORN=1T010:
NEXTN:SOUND4,255-M:FORN=1T010:NEXTN,M:SOU
ND9,0:SOUND10,0
2060 GOTO650
2070 PRINT"  ■      ■      ■      ■      ■      ■      ■      ■
■      ■      ■
2080 PRINT"  ■      ■      ■      ■      ■      ■      ■
■      ■      ■
2090 PRINT"  ■      ■      ■      ■      ■      ■      ■
■      ■      ■
2100 PRINT"  ■      ■      ■      ■      ■      ■      ■
■      ■      ■
2110 PRINT"  ■      ■      ■      ■      ■      ■      ■
■      ■      ■
2120 FORL=17408T017607:IFPEEK(L)=202 POK
EL,160
2130 NEXTL
2140 FORX=0T038STEP2:PRINT@280+X,CHR$(12
9);CHR$(137);:NEXTX
2150 COLOUR16
2160 PRINT:PRINT
2170 PRINT"    Geschrieben von Juergen Buc
hmueller"
2180 PRINT"    fuer Trommeschlaeger Comput
er Studio"
2190 PRINT:PRINT"                      Copyright (C)
1983"
2200 PRINT:PRINT
2210 PRINT"    REKORDPUNKTZAHL:";R2;"0"
2220 PRINT"    _____"
2230 PRINT"    LETZTER SPIELER:";P2;"0"
2240 PRINT"    _____"
2250 PRINT:PRINT"Druucken Sie (S) um das
Spiel zu starten";
2260 PRINT"    oder (R) fuer die Rekordl
iste.";

```

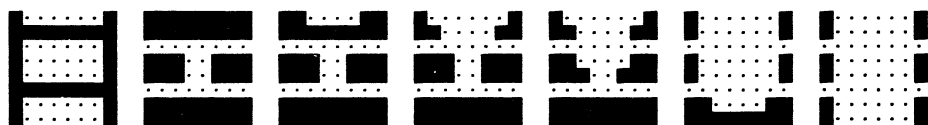
```

2270 FORL=0T07
2280 POKE1-2816,PEEK(((L+C)AND7)-3064)
2290 NEXTL
2300 C=C+1
2310 FORM=1T0500:NEXTM
2320 C#=INKEY$
2330 IFC$="S"RETURN
2340 IFC$(">"R"60T02270
2350 CLS:SOUND7,7:SOUND8,16:SOUND9,16:SOUND10,16:SOUND6,31:SOUND12,50:SOUND13,9
2360 POKE17151,140:POKE17147,19:POKE17144,17:LGR
2370 PRINT@12," REKORDLISTE ":PRINT@52,"
";
2380 FORL=0T08:COLOURL+1
2390 PRINT@128+L*40,CHR$(49+L);". ";
2400 FORM=31100+L*10T031107+L*10
2410 IFPEEK(M)<32POKEM,32
2420 PRINTCHR$(PEEK(M));
2430 NEXTM
2440 PRINT" ";
2450 PRINTPEEK(31108+L*10)+PEEK(31109+L*10)*256;"0"
2460 NEXTL
2470 PRINT@530,"Druecken Sie (S).";
2480 IFPEEK(&HF804)<>860T02480
2490 POKE17151,70:POKE17147,38:POKE17144,32:LGR
2500 RETURN
2510 P1=P:FORM=0T07:POKEP+M,32:NEXTM
2520 Y$=INKEY$:IFY$=""60T02520
2530 A=ASC(Y$):IFA=13PRINTCHR$(15);:RETURN
2540 IFA=8IFP>P1POKEP,32:P=P-1:PRINTCHR$(8);". ";CHR$(24);:60T02520
2550 IFA<3260T02520ELSEIFA>12760T02520
2560 IFP<P1+8POKEP,A:P=P+1:PRINTY$;
2570 60T02520
2580 FORL=17408T018431:POKE1,32:NEXTL
2590 POKE&H4020,0:POKE&H4021,68
2600 RETURN

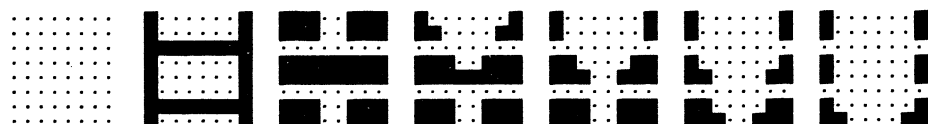
```

# Die Zeichen von HEKTIK

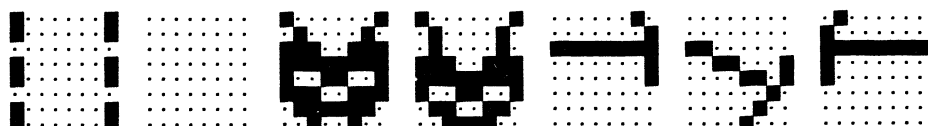
CHR\$(128) CHR\$(129) CHR\$(130) CHR\$(131) CHR\$(132) CHR\$(133) CHR\$(134)



CHR\$(135) CHR\$(136) CHR\$(137) CHR\$(138) CHR\$(139) CHR\$(140) CHR\$(141)



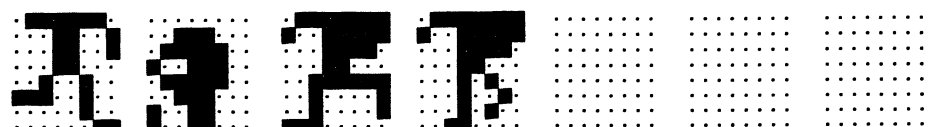
CHR\$(142) CHR\$(143) CHR\$(144) CHR\$(145) CHR\$(146) CHR\$(147) CHR\$(148)



CHR\$(149) CHR\$(150) CHR\$(151) CHR\$(152) CHR\$(153) CHR\$(154) CHR\$(155)



CHR\$(156) CHR\$(157) CHR\$(158) CHR\$(159) CHR\$(160) CHR\$(161) CHR\$(162)



# Erläuterung des Programmes

Zeile(n)	Funktion/Bedeutung
10...320	DATA-Zeilen für die definierbaren Zeichen. Der Colour-Compiler legt DATAs als Doppelbytes ab Adresse 4804H ( = 18436 dez.) ab.
330	Leseschleife für definierbare Zeichen. Kopiert 256 Bytes von 4804H nach F400H.
340	CHAR 2, also sind die Zeichen 128...191 selbst- definierbar und 192...255 vorgegeben.
350	DATA-Zeilen für die HEKTIK-Melodie.
360	Melodie in Buffer nach 30720 kopieren (Adresse von A\$ beim Colour-Compiler)
370	Cursor undefinieren. Sechste und siebte Zeile sind gesetzt und nicht-blinkend.
380	Bildschirm und Farbspeicher löschen.
390	E = Anzahl der Verfolger -1: M2 = Anzahl der Jäger : R1 = Runde
400	Unterprogramm für Ausgabe des Titelbildes aufrufen und auf "S" warten.
410	B2 = Bonus : P2 = Punkte : Q & Q1 = Zähler
420	CLS mit POKEs aufrufen. Y- und X-Schleife für Zeichnen des Bodens eröffnen.
430	Boden besteht abwechselnd aus den Zeichen 129 und 137.
440	Farbe POKEs. 6 = COLOUR 5 = orange.
450	Nächster Ziegel, Ebene.
460	Unterste Zeile wird mit 202 belegt, damit hier das Hacken unmöglich ist.
470	Schleife für Anfangspositionen der Leitern.
480	A und B sind die X-Positionen der jeweils zwei Leitern in einer Etage.



Zeile(n)	Funktion/Bedeutung
490	Zweite Y-Schleife für Länge der Leitern.
500	Sprosse und Farbe (3 = weiß) der ersten Leiter in Bildschirm bzw. Farbspeicher poken.
510	Ebenso mit der zweiten Leiter verfahren.
520	Nächste Sprosse, Etage
530	mit COLOUR 11 in der letzten Zeile anzeigen.
540	Anzahl der restlichen Jäger rechts unten anzeigen
550	Wenn T = 0 (d.h. Alle Verfolger sind abgestürzt) dann Bonus zur Punktzahl addieren.
560	Soundregister vorbelegen.
570	T = Zahl der abgestürzten Verfolger : T2 = Anfang der Melodie -1 : 02 = Oktave : B2 = Bonus
580	Schleife für Vorbelegung der Positionen und Richtungen der Verfolger eröffnen.
590	X- und Y-Position poken.
600	X-Richtung poken (entspr. -1).
610	Y-Richtung poken (entspr. 0).
620	Zeichen unter Verfolger mit 32 (= Space) vorbelegen. Dieses Zeichen wird später anstelle des jeweiligen Verfolgers ausgegeben. Deshalb werden Leitern und Ziegel nicht zerstört.
630	Nächster Verfolger.
640	X = X-Position des Jägers : Y = Y-Position des Jägers : H, H1, H2 = Zeichen unter dem Jäger mit 32 (= Space) vorbelegen. : X2 = Richtung des Jägers für die Berechnung der Zeichen vorbelegen. : Y2 = Y-Richtung vorbelegen.
650	<u>Tastaturabfrage:</u> Bei A = 6 ist (BREAK) & (CLEAR) gedrückt.
660	X- und Y-Richtung gleich null setzen.

Zeile(n)	Funktion/Bedeutung
670	bei A = 32 ist Bit 5 gesetzt. Linkspfeil.
680	bei A = 64 ist Bit 6 gesetzt. Rechtspfeil.
690	bei A = 8 ist Bit 3 gesetzt. Pfeil nach oben.
700	bei A = 16 ist Bit 4 gesetzt. Pfeil nach unten.
710	P = Bildschirmadresse des Jägers.
720	I1 = Zähler für die blinkenden Bonus-Kugeln. Wenn I1 = 0, werden die beiden folgenden Zeilen übersprungen.
730	H1 = Zeichen das vorher unter den Beinen des Jägers stand.
740	H2 = Zeichen unter der Hacke.
750	C = Zähler : C1 ist abwechselnd 0 oder 1.
760	Wenn das Zeichen unter dem Kopf des Jägers gleich einem Verfolger ist, dann EXITUS.
770	Ebenso bei dem Zeichen unter der Hacke.
780	Wenn der Bonus noch größer Null ist, dann dekrementieren.
790	Wenn A AND 127 = 0, bedeutet dies, keine Pfeiltaste gedrückt wurde. Also wird bei einer Bewegung der letzte Jäger im Bildschirm gelöscht. Der Farbspeicher wird auf 15 (= weiß) gesetzt.
800	Wenn X-Position plus Richtung auf den rechten oder linken Rand kommen, dann X-Richtung Null setzen.
810...820	Übernahme der Richtungen in Variablen zur Berechnung der Zeichen für den Jäger.
830	P1 = Position + Y-Richtung P = Position + X-Richtung
840	Z = Zeichen unter den 'Füßen' des Jägers. Wenn alle Vergleiche nicht zutreffen, steht der Jäger nicht auf etwas, auf dem er laufen kann.

Zeile(N)	Funktion/Bedeutung
850	Wenn Z=135 oder Z=143, Absturz des Jägers.
860	Wenn der Jäger nicht auf einer Leiter steht, dann wird die Y-Richtung gleich Null gesetzt.
870	X,Y und P neu berechnen, wenn A AND 127 <> 0, dann neue Zeichen unter dem Jäger in H, H1 und H2 abspeichern.
880	Wenn I1 < 1, dann blinken die Bonuskugeln nicht.
890	Wenn Zeichen unter der Hacke ein Zeichen der Bonuskugel ist, dann Unterprogramm für Bonus-addition ab Zeile 1880 aufrufen.
900	Farben für Jäger POKEn. (P-21504 ergibt Farbspeicheradresse, z.B. 17408-21504=-4096)
910	Wenn seitliche Bewegung und das Zeichen unter der Hacke ein Leerzeichen ist, dann Farbspeicher auf 0 (=grau) setzen.
920...940	In Abhängigkeit von den Richtungen werden die Zeichen des Jägers berechnet und in den Bildschirmspeicher gePOKEt.
950	A AND 128 = 128 bedeutet, daß die Leertaste gedrückt wurde (Hacken). Sonst nach Zeile 1070.
960	Wenn X-Richtung Null ist, kann nicht gehackt werden.
970	Ton beim Hacken.
980	Hacke auf und ab bewegen (Mit C1).
990	Wenn C1=0, dann ist die Hacke oben, und das Loch wird nicht vergrößert.
1000	P wird nun die Bildschirmposition unter der Hacke. Z speichert das dortige Zeichen.
1010	SHIFT gedrückt ? Wenn ja, dann 1050.
1020...1040	In Abhängigkeit vom Ziegeltyp das Loch vergrößern.
1050,1060	In Abhängigkeit vom Ziegeltyp das Loch schließen.

Zeile(n)	Funktion/Bedeutung
1070	In Abhängigkeit von der Spielrunde, wird unterschiedlich oft die Routine zur Bewegung der Verfolger aufgerufen. Ansonsten wird eine Warteschleife ausgeführt.
1080	Wenn Z2=1 ist, wurde man von einem Verfolger erwischt (siehe Unterprogramm ab 1400).
1090	Ausgabe von "*** HEKTIK **", Punktzahl und Bonus in zufälligen Farben.
1100	Wenn alle Verfolger abgestürzt sind, wird Absturzzähler T gleich Null gesetzt, die Rundenzahl R1 um 1 erhöht und, wenn E<5 (E=Anzahl der Verfolger-1) wird E ebenfalls um 1 erhöht. Dann in die nächste Runde (Zeile 420).
1110	Zur Hauptschleife (Tastaturabfrage...).
1120...1140	Ton bei Verlust eines Jägers.
1150	M2 (Jägerzahl) erniedrigen, wenn M2=0 dann Spielende (Zeile 1180 folgende).
1160,1170	Bonus=0, zur Hauptschleife.
1180...1230	Erreichte Punktzahl wird mit der Rekordliste verglichen. Ist man nicht unter den ersten Neun, Sprung zum Titel.
1240...1390	Eingabe des Namens und Einsortieren in die Rekordliste.
1400	Schleife für Verfolgerbewegung eröffnen.
1410	X,Y und Zeichen unter dem jeweiligen Verfolger Q aus dem Speicher in Variablen holen.
1420	Bei V=255 ist der Verfolger schon abgestürzt.
1430	Wenn Q ein Teil des Jägers ist, dann Flag Z2 setzen und Rücksprung.
1440	Wenn Q ein Teil der Bonuskugel ist, aber der Bonus nicht blinkt, wird Q=32 (Leerzeichen) gesetzt.
1450	X- u. Y-Richtung in Variablen speichern.

Zeile(n)	Funktion/Bedeutung
1460	Das letzte Zeichen unter dem jeweiligen Verfolger ( = Q ) wird wieder in den Bildschirmspeicher gePOKEt.
1470	Farbspeicher weiß setzen.
1480	Wenn Q ungleich Leiter (=128) und ungleich Leerzeichen (=32), dann Farbspeicher auf orange (=6).
1490	Wenn X-Position plus X-Richtung gleich linker oder rechter Rand, dann X-Richtung umdrehen.
1500	A wird Wert des Zeichens unterhalb und B oberhalb der Bildschirmposition des Verfolgers.
1510	Wenn Y-Position = 1 (oberste Etage) oder Y-Position = 21 (unterste Etage), dann Y-Richtung entsprechend umdrehen.
1520	Wenn ein Loch im Boden unter dem Verfolger ist (bei A=135 oder A=143), dann stürzt er ab.
1530	Wenn Verfolger auf Leiter, dann X-Richtung auf Null setzen oder wenn er direkt über dem Boden und außerdem auf dem Weg nach unten ist.
1540	Sonst X-Richtung mit 1 oder 3 belegen. ( RND (2) OR 1 ergibt immer 1 oder 3 ).
1550	Wenn Verfolger auf dem Weg nach oben und auf einer Leiter, dann X-Richtung gleich Null setzen.
1560	Sonst wie bei Zeile 1540.
1570	X- und Y-Richtung zu den Positionen addieren.
1580	Neues Q aus dem Bildschirmspeicher in die Tabelle speichern.
1590	Auf die Bildschirmposition einen der beiden Verfolgertypen POKEn.
1600	Farbspeicher mit grün setzen ( = 5 ).
1610	Neue X- und Y-Position in der Positionstabelle abspeichern.

Zeile(n)	Funktion/Bedeutung
1620	Wenn der Verfolger nicht über seinesgleichen steht, dann Sprung zu 1660
1630...1650	Sonst Tabelle nach Verfolger mit gleicher X- und Y-Position durchsuchen, und wenn gefunden, dann dessen Wert für Q übernehmen.
1660	Nächster Verfolger.
1670	Zähler für Noten um eins erhöhen. Wenn Ende der Melodie erreicht ist, dann Zähler zurücksetzen und Zähler für Oktave um eins erhöhen. Wenn Oktave = 7 dann wieder bei 3 beginnen.
1680	A = Ton aus Tabelle. Mit PLAY auf 3. Tonkanal ausgeben. Wenn A<>0 dann Hüllkurve auf Impuls programmieren.
1690	Wenn Bonuskugel sich schon bewegt (Flag gesetzt), dann zu Zeile 1840
1700...1730	Sonst mit einer mit der Rundenzahl zunehmender Wahrscheinlichkeit das Flag und den Zähler für den Bonusblinker belegen.
1740	Absturz eines Verfolgers. Mit steigender Anzahl der Runden immer öfter das Loch schließen.
1750...1830	Absturz eines Verfolgers. Dabei werden die Zeichen und die Farbe unter dem abstürzenden Verfolger immer in Q und Q1 zwischengespeichert. Anschließend wird zur Punktzahl Anzahl der Feinde mal 5 addiert.
1840	Wenn Zähler für Bonuskugel (I1) noch größer Null ist, dann wird er um eins erniedrigt.
1850	Wenn I1 anschließend gleich Null ist, dann wird der Blinker gelöscht und das Flag zurückgesetzt.
1860	Sonst wird abwechselnd (AND 1) eine der zwei verschiedenen Kugeltypen ausgegeben.
1870	Zurück zur Hauptschleife.
1880	Bonuskugel wurde berührt. Dann wird das Flag zurückgesetzt und statt der Kugel "1000" angezeigt.

Zeile(n)	Funktion/Bemerkung
1890	Die Melodie bei einem Bonus wird mit Hilfe von CHR\$(X) in B\$ zwischengespeichert.
1900	Schleife für neun Töne. C wird aktueller Ton.
1910	Auf Kanal 3 mit Lautstärke 14 ausgeben.
1920	Warteschleife und nächster Ton.
1930	Die "1000" wieder löschen
1940	Sämtliche Zwischenspeicher für den 'Untergrund' des Jägers durch 32 (Leerzeichen) ersetzen. 100 zu den Punkten addieren (wird als 1000 angezeigt) und Zähler und Flag zurücksetzen.
1950	Zurück zur Hauptschleife.
1960	Das Loch, über dem der Verfolger, steht wieder schließen.
1970	Ton ausgeben.
1980	Zurück, kein Absturz.
1990	Der Jäger fällt eine Etage tiefer. Wenn die Hacke vorher in einer Leiter war, dann das Zeichen durch ein Leerzeichen ersetzen.
2000	Jäger auf alter Position mit den zwischengespeicherten Werten für den Untergrund löschen.
2010	Neue Position berechnen.
2020	Neue Hintergrundwerte aus dem Bildschirmspeicher lesen und in H, H1 und H2 ablegen.
2030	Aus der letzten X-Richtung die neuen Zeichen für den Jäger berechnen und in den Bildschirmspeicher POKE.
2040	Farben für den Jäger in den Farbspeicher POKE.
2050	Ton erzeugen und wieder abschalten.
2060	Zurück zur Hauptschleife (Tastaturabfrage).

Zeile(n)	Funktion/Bemerkung
2070...2110	Ausgabe des Titels.
2120,2130	CHR\$(202) Grafik-Zeichen durch CHR\$(160) ersetzen. Dieser wird dann anschließend undefiniert, sodaß die Mauer durch die Schrift zu laufen scheint.
2140	Darunter eine Zeile Boden ausgeben.
2150...2260	Restliche Texte ausgeben (In COLOUR 16 = weiß).
2270...2290	Diese Routine kopiert in CHR\$(160) die Mauer um jeweils ein Byte weitergeschoben.
2300	Zähler erhöhen.
2310	Warteschleife.
2320...2340	Tastaturabfrage auf "S" oder "R".
2350	Bei "R" wird die Rekordtabelle ausgegeben. Explosionsgeräusch in PSG programmieren.
2360	CRTC auf doppelte Zeichenhöhe umprogrammieren.
2370...2470	Rekordliste ausdrucken. Neun Plätze in verschiedenen Farben ausgeben. Hinter der Punktzahl eine zusätzliche Null anzeigen.
2480	Ist "S" gedrückt ? Nein, dann wieder abfragen.
2490	Zeichenhöhe wieder auf normal programmieren.
2500	Zurück vom Unterprogramm.
2510	Unterprogramm für Eingabe eines Namens mit 8 Buchstaben. Die RAM-Adresse des Buffers steht in der Variablen P. Zuerst den Buffer löschen.
2520...2570	Tastendrücke abspeichern bis RETURN gedrückt wird. Mehr als 8 Zeichen werden nicht angenommen und man kann mit Linkspfeil nicht vor die erste Bufferadresse P1 zurückgehen.
2580...2600	Schneller CLS, der Leerzeichen in den Bildschirmspeicher POKet. Die Cursorposition in 4020H und 4021H wird auf linke obere Ecke gePOKet.



## Bubble Sort

"Bubble Sort" ist ebenfalls ein Spielprogramm, daß Sie mit dem "Colour Compiler" compilieren müssen. Sie können das Programm aber auch mal im normalen Basic laufen lassen, um die immense Geschwindigkeitserhöhung, die der Compiler bringt, zu sehen. Auf eine Erklärung des Programmes sei verzichtet - die Spielregeln finden Sie im Programm selbst.

Programmlisting:

```
10 CLS:COLOUR1
20 PRINT$14,"Bubble Sort"
30 PRINT:PRINT"Bei diesem Spiel geht es darum, die"
40 PRINT"eine Sorte Kugeln (":COLOUR3:PRINTCHR$(230)::COLOUR1:PRINT")
   auf die eine und"
50 PRINT"die andere Sorte (":COLOUR2:PRINTCHR$(235)::COLOUR1:PRINT")
   auf die andere"
60 PRINT"Seite des Bildschirms zu bringen. Den"
70 PRINT"blauen Teil des Mittelstreifens koennen"
80 PRINT"Sie mit dem Hoch- und Runterpfeil be-"
90 PRINT"wegen und mit <RETURN> oeffnen.":PRINT:PRINT:PRINT"Bitte drue-
cken Sie <RETURN>"
100 A=PEEK(&HF840)AND1
110 IFA=0THEN100
120 SOUND7,248:SOUND8,16:SOUND9,0:SOUND10,0:SOUND1,0:SOUND0,100:SOUND1
2,4
130 COLOUR4:CLS
140 '***** UMRANDUNG ZEICHNEN
150 FORX=0TO39
160 POKEX-4096,3:POKE17408+X,202
170 POKEX-3216,3:POKE18288+X,202
180 NEXTX
190 FORY=40TO840STEP40
200 POKEY-4096,3:POKE17408+Y,202
210 POKEY-4057,3:POKE17447+Y,202
220 POKEY-4076,6:POKE17428+Y,202
230 NEXTY
240 '***** KUGELN ZUFAELLIG SETZEN
250 FORL=0TO9
260 R=2
270 X=AND(38):Y=AND(21):A=(LAND1)*3+R:X1=AND(2)OR1:Y1=AND(2)OR1
280 IFX=20GOTO270
290 POKE31000+L,X:POKE31010+L,Y:POKE31020+L,A:POKE31030+L,X1:POKE31040
+L,Y1
300 GOSUB760
310 NEXTL
320 Z=0
330 P1=40:P=40:D1=0
340 D1=D1+1:PRINT$920,D1;" Zeiteinheiten"; :A=PEEK(&HF840)
350 IF(AAND8)=8IFP>40LETP=P-40
```

```

360 IF(AAND16)=16IFP<800LETP=P+40
370 FORY=40TO840STEP40:POKEY=4076,6:POKE17428+Y,202
380 IFY=PPOKEY-4076,8ELSEIFY=P+40POKEY-4076,8
390 IF(AAND1)=1IFY=PPOKE17428+Y,32ELSEIFY=P+40POKE17428+Y,32
400 NEXTY
410 GOSUB430
420 GOTO340
430 C1=0:C2=0:FORL=0TO9
440 X=PEEK(31000+L):Y=PEEK(31010+L)
450 A=PEEK(31020+L)
460 X1=PEEK(31030+L)-2:Y1=PEEK(31040+L)-2
470 IFX<20IF(LAND1)=0LETC1=C1+1
480 IFX>20IF(LAND1)=1LETC1=C1+1
490 IFX<20IF(LAND1)=1LETC2=C2+1
500 IFX>20IF(LAND1)=0LETC2=C2+1
510 POKE 17408+X+Y*40,32
520 B=PEEK(17408+X+X1+Y*40)
530 C=PEEK(17408+X+(Y+Y1)*40)
540 D=PEEK(17408+X+X1+(Y+Y1)*40)
550 SOUND1,1:IFD<>32IFB=32LETY1=0-Y1:SOUND13,9:GOTO590 ELSE IFC=32LETX
1=0-X1:SOUND13,9:GOTO590
560 SOUND1,0
570 IFB<>32LETX1=0-X1:SOUND13,9
580 IFC<>32LETY1=0-Y1:SOUND13,9
590 X=X+X1:Y=Y+Y1:GOSUB760
600 POKE31000+L,X
610 POKE31010+L,Y
620 POKE31030+L,X1+2
630 POKE31040+L,Y1+2
640 NEXTL
650 IFZ=0IFC1=10GOTO670ELSEIFC2=10GOTO670
660 RETURN
670 RESTORE:POKEP+17428,202:POKEP+17468,202:Z=1:FORQ=1TO255STEP6:GOSUB
430
680 SOUND9,15:SOUND10,15
690 SOUND3,0:SOUND2,Q:SOUND5,0:SOUND4,255-Q
700 NEXTQ
710 SOUND7,255
720 CLS:PRINT"Sie haben es in";D1:PRINT"Zeiteinheiten geschafft."
730 A=PEEK(&HF840)AND1
740 IFA=0THEN730
750 GOTO10
760 POKE &HF000+X+Y*40,A
770 IFA=2THENR1=230ELSER1=235
780 POKE &H4400+X+Y*40,R1
790 RETURN

```

## Tilgungsplan

Das folgende Programm hilft Ihnen bei der Berechnung der Abzahlung einer Ratenschuld. Dabei kann man wählen, ob die Schuld mit konstanter Annuität (Gesamtbelastung = Zinsen + Tilgung) oder mit konstanter Tilgung erfolgen soll.

Programmlisting:

```
10 CLS
20 COLOUR3:PRINT" Tilgungsplan einer Ratenschuld"
30 PRINT
40 CLEAR
50 COLOUR4:INPUT"1 - Konstante Gesamtbelastung
2 - Konstante Tilgung";B
60 ONBGOTO70,220
70 INPUT"Annuitaet, Zins, Kapital eingeben
";A,P,K#
80 PRINT
90 COLOUR2:PRINT"Zeit      Zins      Tilgung Restschuld"
100 K1#=K#
110 X=X+1
120 ZI=INT(K#*P+.5)/100
130 T=A-ZI
140 IFT>=K#THENT=K#
150 R#=K#-T
160 D$=" ### #####.## #####.## #####.##"
170 PRINTUSINGD$;X;ZI;T;R#
180 IFT<=0THENCOLOR7:PRINT"Eine Tilgung ist nicht moeglich":PRINT"Die
Zinsen betragen";ZI;"DM und sind groesser":PRINT"als die Tilgung von"
;T;"DM":GOTO490
190 IFR#<=0THENPRINT"In";X;"Zeitperioden ist die Schuld von":PRINTK1#;
"DM getilgt":GOTO490
200 K#=R#
210 GOTO110
220 DIM Z(1000)
230 A$=" ### #####.## #####.## #####.##"
240 B$="K=#####.## N=### P=###.###"
250 PRINT
260 INPUT"Bitte Kapital, Zeit, Zinssatz eingeben
";K,N,P
270 K1=K
```

```

280 PRINT
290 PRINT
300 PRINT
310 PRINTUSINGB$;K;N;P
320 COLOUR2:PRINT"Zeit      Zinsen      Tilgung      Annuitaet"
330 PRINT
340 T=INT(100*K/N+.5)/100
350 FORI=1TON-1
360 IF(I=21)OR(I=41)OR(I=61)THENINPUT"Weiter";C$
370 Z=INT(K*P+.5)/100
380 PRINTUSINGA$;I;Z;T;T+Z
390 K=K-T
400 Z(I)=Z
410 NEXTI
420 U=INT(K*P+.5)/100
430 Z(N)=U
440 PRINTUSINGA$;N;U;K;U+K
450 FORI=1TON
460 Z1=Z1+Z(I)
470 NEXTI
480 PRINT"Gesamte Zahlungen =";Z1+K1
490 PRINT:COLOUR8:INPUT"Andere Berechnung (J/N)";C$
500 IFLEFT$(C$,1)="J"ORLEFT$(C$,1)="j"THEN10
510 END

```

### Gleichungen mit 3 Unbekannten

Dieses Mathematikprogramm berechnet die 3 Unbekannten von 3 Gleichungen der Form  $ax+by+cz=d$ .

Programmlisting:

```
10 CLS:COLOUR1
20 PRINT"Dieses Programm berechnet aus drei ":PRINT"Gleichungen der Fo
rm"
30 PRINT"ax+by+cz=d (a,b,c und d gegeben) die":PRINT"Unbekannten x,y u
nd z"
40 PRINT
50 COLOUR2:INPUT"a1, b1, c1 und d1";A1,B1,C1,D1
60 COLOUR3:INPUT"a2, b2, c2 und d2";A2,B2,C2,D2
70 COLOUR7:INPUT"a3, b3, c3 und d3";A3,B3,C3,D3
80 D=A1*B2*C3+B1*C2*A3+C1*A2*B3-A3*B2*C1-B3*C2*A1-C3*A2*B1
90 DX=D1*B2*C3+B1*C2*D3+C1*D2*B3-D3*B2*C2-B3*C2*D1-C3*D2*B1
100 DY=A1*D2*C3+D1*C2*A3+C1*A2*D3-A3*D2*C1-D3*C2*A1-C3*A2*D1
110 DZ=A1*B2*D3+B1*D2*A3+D1*A2*B3-A3*B2*D1-B3*D2*A1-D3*A2*B1
120 PRINT:PRINT:COLOUR4
130 IFD=0THENPRINT"Das System ist nicht loesbar.":END
140 PRINT"x =";DX/D
150 PRINT"y =";DY/D
160 PRINT"z =";DZ/D
170 END
```

### Pascalsches Dreieck

Für dieses Programm wird ein Drucker benötigt. Ausgedruckt wird das sogenannte Pascalsche Dreieck (siehe Ausdruck unten). Das Pascalsche Dreieck wird in der Mathematik dazu benutzt, um die Binominalkoeffizienten von Gleichungen n-ten Grades abzu-  
lesen.

Pascalsches Dreieck										
n= 0	-----1-----									
n= 1	-----1-----									
n= 2	-----1-----									
n= 3	-----1-----									
n= 4	-----1-----									
n= 5	-----1-----									
n= 6	-----1-----									
n= 7	-----1-----									
n= 8	-----1-----									
n= 9	-----1-----									
n=10	-----1-----									

#### Programmlisting:

```
10 LPRINTCHR$(15)
20 CLEAR100
30 DIMDR(17,17)
40 LPRINTCHR$(14)"          Pascalsches Dreieck"
50 LPRINTCHR$(15)
60 DR(0,0)=1
70 CLS
80 PRINT"Pascalsches Dreieck"
90 TAB=32
100 FORANZAHL=1TO11
110 LPRINT"n=";
120 LPRINTUSING"###";ANZAHL-1;
130 FORA=0TOTABB
140 LPRINT"--";
150 NEXT
160 FORS=1TOANZAHL
170 DR(ANZAHL,S)=DR(ANZAHL-1,S-1)+DR(ANZAHL-1,S)
180 IFS>1THENLPRINTUSING"#####";DR(ANZAHL,S);:NEXTELSELPRINTUSING"###";
DR(ANZAHL,S);:NEXT
190 FORA=1TOTAB
200 LPRINT"--";
210 NEXT
220 LPRINT
230 TAB=TAB-3
240 NEXT
250 LPRINTSTRING$(6,10)
```

## Garbage Collect

Das Programmlisting, das nun folgt, soll Ihnen einen Effekt verdeutlichen, dem Sie möglicherweise schon begegnet sind, ohne ihn sich erklären zu können.

Gemeint ist, daß sich der Computer in längeren Basic-Programmen für einige Sekunden "aufhängt", d.h. daß das Programm stehenbleibt und sogar mit der BREAK-Taste nichts mehr zu machen ist. Wenn Sie das folgende Programm eingeben und starten, wird das Problem sehr deutlich: Obwohl zwischen den einzelnen PRINTs nur ganz normale Variablenzuweisungen ausgeführt werden, dauert dies bis zu einigen Sekunden. Dies kann auch dann Probleme bereiten, wenn Sie Daten vom Band einlesen, da durch eine derartige Verzögerung Daten verlorengehen können.

Was ist nun der Grund für diese Verzögerung und wie kann man das Problem umgehen und seine Basicprogramme damit schneller machen?

Der Grund ist Zeile 50:

```
50 DIM A(MEM/3-500)
```

In dieser Zeile wird ein großes Feld dimensioniert (abhängig vom vorhandenen Speicher). Wird nun eine neue Variable eingeführt, wird das ganze Feld verschoben, um für die Variable Platz im Speicher zu schaffen. Dies verursacht besagte Verzögerungen. Fügt man nun eine Zeile ein, die alle später gebrauchten Variablen bereits vor dem DIM-Befehl schon einmal benutzt, werden diese schon vor dem Feld im Speicher angelegt, das dann später nicht mehr verschoben werden muß. Dies können Sie z.B. bei dem vorliegenden Programm einfach dadurch erreichen, indem Sie die ganz unten gelistete Zeile 5 mit eingeben, die allen benutzten Variablen den Wert 0 zuweist. Das Programm läuft dann erheblich schneller!

Programmlisting:

```
10 CLS:COLOUR1
20 DEFSTR A
30 R=0
40 Q=R
50 DIM A(MEM/3-500)
60 A(0)=" "
70 A(1)="<RETURN>"
80 CLS
90 PRINT"Druecken Sie <RETURN> zur Demonstration":PRINT"des Garbage Collect"
100 Q=1-Q
```

```

110 COLOUR=1:PRINT$13,A(Q);
120 FORR=1TO20
130 IFINKEY$(<>CHR$(13))THENNEXT:GOTO100
140 CLS:COLOUR9
150 PRINT"X =";
160 X=2
170 PRINTX
180 X1=X
190 PRINT"Zwei und zwei sind";
200 Y=X+X
210 PRINTY
220 Y1=Y
230 COLOUR1:PRINT"Wussten ";
240 A=""
250 COLOUR2:PRINT"Sie";
260 B=0
270 COLOUR3:PRINT", dass ";
280 C=0
290 COLOUR4:PRINT"Ihr ";
300 D=0
310 COLOUR5:PRINT"Computer ";
320 E=0
330 COLOUR6:PRINT"so "
340 F=0
350 COLOUR7:PRINT"lang";
360 G=0
370 PRINT"s";
380 H=0
390 PRINT"a";
400 I=J
410 J=K
420 PRINT"m ";
430 K=C
440 L=K
450 M=B
460 COLOUR8:PRINT"ist?"
470 COLOUR1

```

```

5 B=0:C=B:D=B:E=B:F=B:G=B:H=B:I=B:J=B:K=B:L=B:M=B:Q=B:R=B:X=B:X1=B:Y=B
:Y1=B:A$=""

```



## Ein Maschinensprache-Monitor in Basic

Da es im Laufe dieses Buches nötig sein wird, kleinere Programme in hexadezimal einzugeben oder sich Speicherinhalte anzusehen, geben Sie bitte das hier aufgelistete Basicprogramm ein und speichern Sie es auf Kassette:

```
10 CLS
20 CLEAR 1000
30 PRINT"Command";
40 INPUT A$
50 IF A$="" THEN 30
60 GOSUB 1460
70 IF B$="P" THEN PR=1: GOSUB 1460 ELSE PR=0
80 C$="HAETGRW"
90 FOR A=1 TO LEN(C$)
100 IF B$<>MID$(C$, A, 1) THEN NEXT A: GOTO 30
110 ON A GOTO 130, 390, 590, 740, 900, 1040, 1730
120 GOTO 30
130 GOSUB 1510
140 X=Z
150 Z=X
160 GOSUB 1600
170 PRINT Z$;" ";
180 IF PR=1 THEN LPRINT Z$;" ";
190 C=0
200 FOR Y=0 TO 7
210 Z=PEEK(X+Y)
220 C=C+Z
230 GOSUB 1600
240 PRINT RIGHT$(Z$, 2);" ";
250 IF PR=1 THEN LPRINT RIGHT$(Z$, 2);" ";
260 NEXT Y
270 PRINT" ";
280 IF PR=1 THEN LPRINT" ";
290 Z=C
300 GOSUB 1600
310 PRINT RIGHT$(Z$, 2)
320 IF PR=1 THEN LPRINT RIGHT$(Z$, 2)
330 X=X+8
340 KB=PEEK(&HF840)
350 IF KB=2 THEN 30
360 IF KB<>128 THEN 150
370 KB=PEEK(&HF840)
380 IF KB<>1 THEN 370 ELSE 150
390 GOSUB 1510
400 X=Z
410 Z=X
420 GOSUB 1600
430 PRINT Z$;" ";
440 IF PR=1 THEN LPRINT Z$;" ";
450 FOR Y=0 TO 23
460 Z=PEEK(X+Y)
470 IF Z<32 THEN Z=ASC(".")
480 PRINT CHR$(Z);
490 IF PR=1 THEN LPRINT CHR$(Z);
500 NEXT Y
510 PRINT
```

```

520 IF PR=1 THEN LPRINT
530 X=X+24
540 KB=PEEK(&HF840)
550 IF KB=2 THEN 30
560 IF KB<>128 THEN 410
570 KB=PEEK(&HF840)
580 IF KB<>1 THEN 570 ELSE 410
590 GOSUB 1510
600 X=Z
610 Z=X
620 GOSUB 1600
630 PRINT Z$;" - ";
640 Z=PEEK(X)
650 GOSUB 1600
660 PRINT RIGHT$(Z$, 2);" - ";
670 A$=""
680 INPUT A$
690 IF A$="" THEN 30
700 GOSUB 1510
710 POKE X, Z AND 255
720 X=X+1
730 GOTO 610
740 GOSUB 1510
750 X=Z
760 Z=X
770 GOSUB 1600
780 PRINT Z$;" - ";
790 Z=PEEK(X)
800 GOSUB 1600
810 PRINT RIGHT$(Z$, 2);" (";
820 IF Z<32 THEN Z=ASC(".")
830 PRINT CHR$(Z);") - ";
840 A$=""
850 INPUT A$
860 IF A$="" THEN 30
870 POKE X, ASC(A$)
880 X=X+1
890 GOTO 760
900 GOSUB 1510
910 Z1=INT(Z/256)AND 255
920 Z2=Z AND 255
930 POKE 16526, Z2
940 POKE 16527, Z1
950 GOSUB 1460
960 IF B$="" THEN 950
970 A$=B$+A$
980 GOSUB 1510
990 Z=Z AND 255
1000 Z=USR(Z)
1010 GOSUB 1600
1020 PRINT RIGHT$(Z$, 2)
1030 GOTO 30
1040 GOSUB 1510

```

```

1050 IF Z>=0 THEN X1=Z ELSE X1=Z+65536
1060 GOSUB 1460
1070 IF B$=" " THEN 1060
1080 A$=B$+A$
1090 GOSUB 1510
1100 IF Z>=0 THEN X2=Z ELSE X2=Z+65536
1110 PRINT"  A      B      A+B      A-B"
1120 PRINT" ";
1130 Z=X1
1140 GOSUB 1600
1150 PRINT Z$;" ";
1160 Z=X2
1170 GOSUB 1600
1180 PRINT Z$;" ";
1190 Z=X1+X2
1200 IF Z>65535 THEN Z=Z-65536
1210 GOSUB 1600
1220 PRINT Z$;" ";
1230 Z=X1-X2
1240 IF Z<0 THEN Z=Z+65536
1250 GOSUB 1600
1260 PRINT Z$
1270 Z=X1
1280 GOSUB 1420
1290 PRINT Z$;" ";
1300 Z=X2
1310 GOSUB 1420
1320 PRINT Z$;" ";
1330 Z=X1+X2
1340 IF Z>65535 THEN Z=Z-65536
1350 GOSUB 1420
1360 PRINT Z$;" ";
1370 Z=X1-X2
1380 IF Z<0 THEN Z=Z+65536
1390 GOSUB 1420
1400 PRINT Z$
1410 GOTO 30
1420 Z$=STR$(Z)
1430 Z$=RIGHT$(Z$, LEN(Z$)-1)
1440 IF LEN(Z$)<5 THEN Z$=STRING$(5-LEN(Z$), 32)+Z$
1450 RETURN
1460 B$=""
1470 IF LEN(A$)=0 THEN RETURN
1480 B$=LEFT$(A$, 1)
1490 A$=RIGHT$(A$, LEN(A$)-1)
1500 RETURN
1510 Z=0
1520 GOSUB 1460
1530 IF B$<"0" OR B$>"F" THEN 1580
1540 IF B$>"9" AND B$<"A" THEN 1580
1550 IF B$<="9" THEN Z1=ASC(B$)-48 ELSE Z1=ASC(B$)-55
1560 Z=Z*16+Z1
1570 GOTO 1520

```

```

1580 IF Z>32767 THEN Z=Z-65536
1590 RETURN
1600 IF Z>32767 THEN Z=Z-65536
1610 Z1=INT(Z/256)AND 255
1620 Z2=Z AND 255
1630 Z$=""
1640 Z3=INT(Z1/16)
1650 GOSUB 1710
1660 Z3=Z1 AND 15
1670 GOSUB 1710
1680 Z3=INT(Z2/16)
1690 GOSUB 1710
1700 Z3=Z2 AND 15
1710 IF Z3<10 THEN Z$=Z$+CHR$(Z3+48) ELSE Z$=Z$+CHR$(Z3+55)
1720 RETURN
1730 RESTORE
1740 WR$=""
1750 FOR X=1 TO 127
1760 READ A
1770 WR$=WR$+CHR$(A)
1780 NEXT
1790 X=VARPTR(WR$)
1800 X=PEEK(X+1)+256*PEEK(X+2)
1810 IF X>32767 THEN X=X-65536
1820 R=X+95
1830 R1=INT(R/256)AND 255
1840 R2=R AND 255
1850 POKE X+66, R2
1860 POKE X+67, R1
1870 POKE X+75, R2
1880 POKE X+76, R1
1890 GOSUB 2140
1900 GOSUB 2210
1910 GOSUB 2140
1920 GOSUB 2210
1930 GOSUB 2140
1940 PRINT"Filename":
1950 INPUT PN$
1960 FOR X1=X TO X+5
1970 POKE X1, 32
1980 NEXT
1990 FOR X1=1 TO LEN(PN$)
2000 POKE X-1+X1, ASC(MID$(PN$, X1, 1))
2010 NEXT
2020 X=X+6
2030 X1=INT(X/256)AND 255
2040 X2=X AND 255
2050 POKE 16526, X2
2060 POKE 16527, X1
2070 X1=USR(0)
2080 GOTO 30

```

```

2090 DATA 0, 0, 0, 0, 0, 0, 32, 32, 32, 32, 32, 32
2100 DATA 42, 142, 64, 17, 250, 255, 25, 205, 63, 2, 62, 85, 205, 31,
2, 6, 6, 126, 35, 205, 31, 2, 16, 249, 17, 244, 255, 25, 229, 221, 225
, 221, 94, 0, 221, 86, 1, 221, 110, 2, 221, 102, 3
2110 DATA 175, 237, 82, 35, 124, 183, 40, 8, 6, 0, 205, 0, 0, 37, 24,
244, 125, 69, 183, 196, 0, 0, 62, 120, 205, 31, 2
2120 DATA 221, 126, 4, 205, 31, 2, 221, 126, 5, 205, 31, 2, 201
2130 DATA 62, 60, 205, 31, 2, 120, 205, 31, 2, 123, 205, 31, 2, 79, 12
2, 205, 31, 2, 129, 79, 26, 19, 205, 31, 2, 129, 79, 16, 247, 195, 31,
2
2140 GOSUB 1510
2150 Z1=INT(Z/256)AND 255
2160 Z2=Z AND 255
2170 POKE X, Z2
2180 POKE X+1, Z1
2190 X=X+2
2200 RETURN
2210 GOSUB 1460
2220 IF B$=" " THEN 2210
2230 A$=B$+A$
2240 RETURN

```

Dieser Monitor unterstützt 7 Befehle, nämlich:

1. Hex-Dump

Eingabe: Adresse

Ab der eingegebenen Adresse wird der Speicherinhalt in hexadezimal gelistet. Jeweils nach Ausgabe einer Zeile wird die Tastatur auf folgende Tasten abgefragt:

<CLEAR> - Listing abbrechen.

<Leertaste> - Listing anhalten, bis <RETURN> gedrückt wird.

Die einzelnen Zeilen haben folgendes Format:

Die ersten 4 Zeichen geben die Adresse an, in der das erste der 8 Bytes, die in dieser Zeile gelistet werden, steht.

Dann folgen die 8 Bytes und zum Schluß, etwas abgesetzt, folgt die sogenannte Checksum. Dieses Byte ist das niederwertige Byte der Summe aller 8 Bytes und dient als Prüfmöglichkeit für die Richtigkeit einer Eingabe. Alle Maschinenprogramme in diesem Buch werden als Hex-Dump mit Prüfsumme gelistet, so daß Sie eine einfache Möglichkeit haben, die Richtigkeit Ihrer Eingabe zu prüfen.

2. ASCII-Dump

Eingabe: Adresse

Ab der eingegebenen Adresse wird der Speicherinhalt als ASCII-Zeichen gelistet. Jede Zeile beginnt mit der Adresse, in der das erste der 24 folgenden Zeichen steht, dann folgen die ASCII-Zeichen. Jedes Byte unter 32 dez. wird als '.' dargestellt, um die Bildschirmausgabe nicht durch Codes wie 'Backspace' etc. zu zerstören. Auch beim ASCII-Dump kann die Ausgabe durch <CLEAR> abgebrochen und durch <Leertaste> angehalten werden.

3. Edit hexadezimal

Eingabe: Adresse

Ab der eingegebenen Adresse kann der Speicherinhalt verändert werden. Es wird jeweils die Adresse und der bisherige Inhalt dieser Speicherstelle ausgegeben. Sie können dann ein neues Byte eingeben, das dann in dieser Speicherstelle abgelegt wird. Wenn Sie statt einer Eingabe nur <RETURN> drücken, wird der Editmodus abgebrochen.

4. Edit ASCII

Eingabe: Adresse

Ab der eingegebenen Adresse können Sie Texte direkt in den Speicher eingeben. Die Adresse wird angegeben, ihr Inhalt wird in hexadezimal und ASCII (Code kleiner als 32 = '.') ausgegeben. Sie können nun ein Zeichen eingeben, das an dieser Speicherstelle gespeichert wird. Auch hier wird die Eingabe abgebrochen, wenn Sie nur <RETURN> drücken. Achtung: Um ein Leerzeichen in den Speicher einzugeben, müssen Sie folgendes eingeben: " ". Dies liegt daran, daß der INPUT-Befehl des Colour-Basic alle führenden Leerzeichen ignoriert, eine Eingabe eines Leerzeichens allein also als Leerstring angesehen wird.

5. Go to memory

Eingabe: Gadresse byte

Es wird zu einem Programm gesprungen, das bei der eingegebenen Adresse beginnt. Wenn Ihr Programm mit einem CALL 0A7FH beginnt, so können Sie ein Byte in das L-Register übergeben. Wenn Sie Ihr Programm mit einem JP 0A9AH beenden, so wird der Inhalt des L-Registers wieder an das Basic übergeben und der Monitor gibt es in hexadezimal aus.

6. Hexadezimal Rechnung

Eingabe: Rzahl1 zahl2

Zahl1, Zahl2, ihre Summe und ihre Differenz werden in hexadezimal und ASCII ausgegeben.

7. SYSTEM-Band schreiben

Eingabe: Wstart ende einsprung

Zusätzlich wird noch ein Programmname abgefragt. Dieser Name darf maximal 6 Zeichen lang sein und das erste Zeichen muß ein Buchstabe sein. Dann wird der Bereich von start bis ende im SYSTEM-Band-Format auf Kassette geschrieben. Um dieses Band wieder einzulesen, benutzen Sie den SYSTEM-Befehl des Colour-Basic und geben Sie den von Ihnen gewählten Programmnamen an.

Anmerkung: Für das H und das A-Kommando gilt:

PHadresse bzw. PAadresse gibt das entsprechende Listing parallel auf Bildschirm und Drucker aus.

## Erklärung des Programms:

Zeile 10 - 20: Initialisierung: Bildschirm löschen und Stringspace vergrößern.  
Zeile 30 - 50: Kommandoeingabe. Wenn nur <RETURN> gedrückt wurde, Eingabe wiederholen.  
Zeile 60 - 70: Erstes Zeichen der Eingabe lesen. Wenn P, Druckerflag (PR) setzen und nächstes Zeichen lesen.  
Zeile 80 - 110: Kommando aus C\$ herausuchen und die entsprechende Routine anspringen:  
H = 130  
A = 390  
E = 590  
T = 740  
G = 900  
R = 1040  
W = 1730  
Zeile 120: Falls Zeichen kein zulässiges Kommando, wieder zur Eingabe springen.

### Zeile 130 - 380: Hex-Dump

Zeile 130 - 140: Adresse aus A\$ in Z lesen und in X umspeichern.  
Zeile 150 - 160: X in Hexadezimalzahl in Z\$ umwandeln.  
Zeile 170: Adresse des ersten der folgenden 8 Bytes ausgeben.  
Zeile 180: Falls nötig, Ausgabe zusätzlich auf Drucker.  
Zeile 190: Prüfsumme auf Null initialisieren.  
Zeile 200 - 260: 8 Bytes in hexadezimal auf Bildschirm (und auf Drucker wenn nötig) ausgeben und Prüfsumme berechnen.  
Zeile 270 - 320: Prüfsumme ausgeben.  
Zeile 330: Startadresse um 8 erhöhen.  
Zeile 340: KB fragt eine Tastenreihe ab.  
Zeile 350: Wenn <CLEAR> gedrückt, Sprung zur Kommandoeingabe.  
Zeile 360: Wenn nicht <Leertaste> gedrückt, nächste Zeile ausgeben.  
Zeile 370 - 380: Warten, bis <RETURN> gedrückt, dann nächste Zeile ausgeben.

### Zeile 390 - 580: ASCII-Dump

Zeile 390 - 400: Adresse aus A\$ in Z lesen und in X umspeichern.  
Zeile 410 - 440: Startadresse der Zeile ausgeben.  
Zeile 450 - 500: 24 ASCII-Zeichen ausgeben.  
Zeile 510 - 520: Cursor an den Anfang der nächsten Zeile bringen.  
Zeile 530: Startadresse um 24 erhöhen.



Zeile 540 - 580: Tastaturabfrage:  
<CLEAR> - Listing abbrechen  
<Leertaste> - Listing anhalten.

Zeile 590 - 730: Edit hexadezimal

Zeile 590 - 600: Adresse aus A\$ in Z lesen und in X umspeichern.  
Zeile 610 - 630: Adresse und Trennstrich ausgeben.  
Zeile 640 - 660: Bisherigen Inhalt der Speicherstelle X und zweiten Trennstrich ausgeben.  
Zeile 670 - 680: Eingabe in A\$ lesen.  
Zeile 690: Wenn nur <RETURN> gedrückt wurde, Eingabe abbrechen.  
Zeile 700 - 710: Das eingegebene Byte in Adresse X ablegen. Das Z AND 255 dient dazu, sicherzustellen, daß nicht aus Versehen eine Zahl größer als 255 dez. gePOKEt wird.  
Zeile 720 - 730: Adresse um 1 erhöhen und Eingabevorgang wiederholen.

Zeile 740 - 890: Edit ASCII

Zeile 740 - 750: Adresse aus A\$ in Z lesen und in X umspeichern.  
Zeile 760 - 780: Adresse und ersten Trennstrich ausgeben.  
Zeile 790 - 830: Inhalt der Speicherstelle X in hexadezimal und als ASCII-Zeichen ausgeben.  
Zeile 840 - 850: Eingabe in A\$ lesen.  
Zeile 860: Wenn nur <RETURN> gedrückt, Eingabe abbrechen.  
Zeile 870: ASCII-Wert des eingegebenen Zeichens in X ablegen.  
Zeile 880: Adresse um 1 erhöhen.  
Zeile 890: Eingabevorgang wiederholen.

Zeile 900 - 1030: Go to memory

Zeile 900: Sprungadresse aus A\$ in Z lesen.  
Zeile 910 - 920: Z1 = höherwertiges Byte von Z.  
Z2 = niederwertiges Byte von Z.  
Zeile 930 - 940: USR-Sprungadresse festlegen.  
Zeile 950 - 970: Alle Leerzeichen am Anfang von A\$ wegstreichen.  
Zeile 980 - 990: Zu übergebendes Byte aus A\$ in Z lesen.  
Zeile 1000: Programm starten. Das Argument in Z übergeben und das Ergebnis des Programms wieder in Z abspeichern.  
Zeile 1010 - 1020: Ergebnis in hexadezimal ausgeben.  
Zeile 1030: Sprung zur Kommandoeingabe.

Zeile 1040 - 1410: Hexadezimalrechnung

Zeile 1040: Zahl aus A\$ in Z lesen.  
Zeile 1050: Aus einer vorzeichenbehafteten Zahl in Z eine positive Zahl in X1 machen.  
( -32768 <= Z <= 32767; 0 <= X1 <= 65535 )  
Zeile 1060 - 1080: Alle Leerzeichen am Anfang von A\$ löschen.  
Zeile 1090: Zahl aus AS in Z lesen.  
Zeile 1100: Aus einer vorzeichenbehafteten Zahl in Z eine positive Zahl in X2 machen (siehe 1050).  
Zeile 1110: Kopfzeile der Tabelle ausgeben.  
Zeile 1120 - 1260: X1, X2, X1 + X2 und X1 - X2 in hexadezimal ausgeben.  
Zeile 1270 - 1400: X1, X2, X1 + X2 und X1 - X2 in dezimal ausgeben.  
Zeile 1410: Sprung zur Kommandoeingabe

Zeile 1420 - 1450: Zahl in Z in Dezimalstring umwandeln. Z\$ enthält Zahl rechtsbündig.

Zeile 1420: Z\$ = Zahl aus Z in Stringform.  
Zeile 1430: Vorzeichen aus Z\$ entfernen.  
Zeile 1440: Falls nötig, Z\$ mit Leerzeichen auf 5 Zeichen auffüllen.  
Zeile 1450: Rückkehr zum aufrufenden Programm.

Zeile 1460 - 1500: Das erste Zeichen aus A\$ in B\$ übertragen und A\$ um ein Zeichen kürzen.

Zeile 1460: B\$ löschen.  
Zeile 1470: Falls A\$ leer ist, sofort zurückspringen.  
Zeile 1480: Das erste Zeichen von A\$ in B\$ speichern.  
Zeile 1490: A\$ um ein Zeichen kürzen.  
Zeile 1500: Rücksprung.

Zeile 1510 - 1590: Hexadezimalzahl aus A\$ in Z lesen.

Zeile 1510: Z zunächst löschen.  
Zeile 1520: Ein Zeichen aus A\$ lesen.  
Zeile 1530 - 1540: Falls das Zeichen keine zulässige Hexziffer ist ( 0 - 9 oder A - F ), Ende der Hexadezimalzahl.  
Zeile 1550: Hexadezimalziffer in B\$ umwandeln in Dezimalzahl in Z1.  
Zeile 1560: Z entsprechend erhöhen.  
Zeile 1570: Schleife wiederholen.  
Zeile 1580: Falls Z größer als 32767 ist, muß es eine negative Zahl sein (damit PEEK und POKE funktioniert).  
Zeile 1590: Rücksprung.

Zeile 1600 - 1720: Zahl in Z in hexadezimal umwandeln und in Z\$ ablegen.

Zeile 1600: Falls Z größer als 32767 ist, muß es eine negative Zahl sein.  
Zeile 1610: Z1 = höherwertiges Byte von Z.  
Zeile 1620: Z2 = niederwertiges Byte von Z.  
Zeile 1630: Z\$ als leer initialisieren.  
Zeile 1640 - 1650: Höchstwertige Ziffer erzeugen.  
Zeile 1660 - 1670: Zweite Ziffer erzeugen.  
Zeile 1680 - 1690: Dritte Ziffer erzeugen.  
Zeile 1700: Letzte Ziffer erzeugen.  
Zeile 1710: Zahl in Z3 in hexadezimale Ziffer umwandeln und an Z\$ anhängen.  
Zeile 1720: Rücksprung.

Zeile 1730 - 2080: Write SYSTEM-Tape

Zeile 1730 - 1780: Maschinenprogramm in WR\$ einlesen.  
Zeile 1790: X zeigt auf Eintrag der Variable WR\$.  
In X steht die Länge des Strings.  
In X+1 steht das niederwertige Byte der Adresse, bei der der String im Speicher beginnt.  
In X+2 steht das höherwertige Byte der Adresse, bei der der String im Speicher beginnt.  
Zeile 1800: X = Adresse, bei der der Stringtext im Speicher beginnt.  
Zeile 1810: Adresse in X für PEEK und POKE anpassen.  
Zeile 1820: R = Adresse eines Unterprogramms innerhalb des Maschinenprogramms. (Genauere Erläuterungen zu diesem Programm siehe nächstes Kapitel)  
Zeile 1830: R1 = höherwertiges Byte von R.  
Zeile 1840: R2 = niederwertiges Byte von R.  
Zeile 1850 - 1880: Unterprogrammadresse in Programm POKEn.  
Zeile 1890 - 1930: Start-, End- und Einsprungadresse einlesen und in die ersten sechs Bytes des Maschinenprogramms POKEn.  
Zeile 1940 - 1950: Programmnamen in PN\$ einlesen.  
Zeile 1960 - 1980: Byte Nr. 6 bis Byte Nr. 11 des Maschinenprogramms mit Leerzeichen füllen.  
Zeile 1990 - 2010: Den Programmnamen in die Bytes 6 bis maximal 11 des Maschinenprogramms POKEn.  
Zeile 2020 - 2060: Startadresse des Programms für USR-Aufruf POKEn.  
Zeile 2070: Maschinenprogramm aufrufen.  
Zeile 2080: Zur Kommandoeingabe springen.

Zeile 2090 - 2130: Das Maschinenprogramm für Write Tape.  
Zeile 2090 - 2120: Das eigentliche Programm.  
Zeile 2130: Das Unterprogramm zum Schreiben eines Datenblocks.

Zeile 2140 - 2200: Eine Zahl aus A\$ lesen und in X und X+1 POKEn.

Zeile 2140: Eine Zahl aus A\$ in Z lesen.  
Zeile 2150: Z1 = höherwertiges Byte von Z.  
Zeile 2160: Z2 = niederwertiges Byte von Z.  
Zeile 2170: Das niederwertige Byte wird in X gePOKEd.  
Zeile 2180: Das höherwertige Byte wird in X+1 gePOKEd.  
Zeile 2190: X wird um 2 erhöht.  
Zeile 2200: Rücksprung.

Zeile 2210 - 2240: Alle führenden Leerzeichen aus A\$ löschen.

Zeile 2210: Ein Zeichen aus A\$ in B\$ lesen.  
Zeile 2220: Wenn BS ein Leerzeichen enthält, Prozedur wiederholen.  
Zeile 2230: A\$ wieder korrigieren.  
Zeile 2240: Rücksprung.

Nach der Erläuterung des Basic-Teils des Monitorprogramms hier nun noch ein Listing und die Erläuterungen zu dem Maschinenprogramm, das die Bandaufzeichnung macht.

8404	2A	8E	40	LD	HL,(408EH)	*.8
8407	11	FA	FF	LD	DE,FFFAH	...
840A	19			ADD	HL,DE	.
840B	CD	3F	02	CALL	023FH	.?.
840E	3E	55		LD	A,55H	>U
8410	CD	1F	02	CALL	021FH	...
8413	06	06		LD	B,06H	..
8415	7E			LD	A,(HL)	.
8416	23			INC	HL	#
8417	CD	1F	02	CALL	021FH	...
841A	10	F9		DJNZ	8415H	..
841C	11	F4	FF	LD	DE,FFF4H	...
841F	19			ADD	HL,DE	.
8420	E5			PUSH	HL	.
8421	DD	E1		POP	IX	..
8423	DD	5E	00	LD	E,(IX+00H)	..^.
8426	DD	56	01	LD	D,(IX+01H)	.V.
8429	DD	6E	02	LD	L,(IX+02H)	.n.
842C	DD	66	03	LD	H,(IX+03H)	.f.
842F	AF			XOR	A	.
8430	ED	52		SBC	HL,DE	.R
8432	23			INC	HL	#
8433	7C			LD	A,H	.
8434	B7			OR	A	.
8435	28	08		JR	Z,843FH	(.
8437	06	00		LD	B,00H	..
8439	CD	57	84	CALL	8457H	.W.
843C	25			DEC	H	%
843D	18	F4		JR	8433H	..
843F	7D			LD	A,L	.
8440	45			LD	B,L	E
8441	B7			OR	A	.
8442	C4	57	84	CALL	NZ,8457H	.W.
8445	3E	78		LD	A,78H	>x
8447	CD	1F	02	CALL	021FH	...
844A	DD	7E	04	LD	A,(IX+04H)	...
844D	CD	1F	02	CALL	021FH	...
8450	DD	7E	05	LD	A,(IX+05H)	...
8453	CD	1F	02	CALL	021FH	...
8456	C9			RET		.
8457	3E	3C		LD	A,3CH	><
8459	CD	1F	02	CALL	021FH	...
845C	78			LD	A,B	x
845D	CD	1F	02	CALL	021FH	...
8460	7B			LD	A,E	ä
8461	CD	1F	02	CALL	021FH	...
8464	4F			LD	C,A	O
8465	7A			LD	A,D	z
8466	CD	1F	02	CALL	021FH	...
8469	81			ADD	A,C	.
846A	4F			LD	C,A	O
846B	1A			LD	A,(DE)	.
846C	13			INC	DE	.
846D	CD	1F	02	CALL	021FH	...
8470	81			ADD	A,C	.
8471	4F			LD	C,A	O
8472	10	F7		DJNZ	846BH	..
8474	C3	1F	02	JP	021FH	...

In dem Listing auf der vorherigen Seite beginnt das Programm bei Adresse 8404H. Dies kann jedoch nicht immer so sein, da die Strings immer an anderen Stellen abgelegt werden. Dies hängt zum Beispiel davon ab, ob Ihr Rechner über 16 oder 32 K RAM verfügt, ob obere RAM-Bereich durch Eingabe einer MEM SIZE geschützt wurden etc. Hierdurch taucht ein großes Problem auf: Das Maschinenprogramm muß zum einen auf Adressen zugreifen, in die das Basic-Programm die Start-, End- und Einsprungadresse des auf Band zu speichernden Speicherbereichs sowie den Programmnamen abgelegt hat. Zum anderen wird in dem Programm zweimal ein Unterprogramm aufgerufen. Auch diese Adresse muß korrigiert werden. Dieses Problem löst schon der Basic-Teil der Write SYSTEM Tape Routine. In den Zeilen 1820 bis 1880 wird aus der Anfangsadresse des Programms der Anfang des Unterprogramms berechnet und an den entsprechenden Stellen im eigentlichen Programm abgelegt (in diesem Listing wären dies die Adressen 843AH und 8443H). Um die Stellen zu finden, an denen das Basic-Programm die nötigen Adressen und den Programmnamen abgelegt hat, wird derselbe Trick benutzt, diesmal jedoch vom Maschinenprogramm aus.

In Adresse 8404 wird das HL-Register mit dem Inhalt der Speicherzellen 408EH/408FH geladen. Diese Speicherzellen enthalten die Startadresse der Routine, die mit dem USR-Befehl aufgerufen werden soll (kennen Sie sicher: 16526/16527 dez.). In diesem Fall würde HL also 8404H enthalten, denn dies ist ja die Startadresse des Programms!

Die nächsten zwei Befehl erniedrigen HL um 6, sodaß es nun auf den Beginn des Programmnamens im Speicher zeigt. Nun wird zunächst eine ROM-Routine aufgerufen, die einen sogenannten Leader auf das Band schreibt. Dieser Leader sorgt dafür, daß beim Lesen alle Bytes genauso gefunden werden, wie sie geschrieben wurden (siehe nächstes Kapitel: Format von SYSTEM- und CLOAD-Bändern). Die Befehlsfolge LD A,55H; CALL 021FH zeichnet das Byte 55H auf Kassette auf. Die Befehle in den Adressen 8413H bis 841BH stellen eine Schleife dar, die den Programmnamen auf Kassette aufzeichnet. Die Befehle in Adresse 841CH bis 842EH laden die Startadresse des Programms in DE und die Endadresse in HL. Nun wird die Differenz berechnet und in HL abgelegt. Adresse 8433H bis 843EH stellt wieder eine Schleife dar, in der sovieler 256 Byte lange Blöcke vom Speicher auf Kassette übertragen werden, wie durch die Programmlänge angegeben wurden. Dann wird noch (843FH bis 8444H), falls nötig, ein kürzerer Block ausgegeben, der die restlichen Bytes enthält. Zum Abschluß wird die Einsprungadresse aufgezeichnet und in den Basic-Teil des Monitorprogramms zurückgesprungen.

Das Unterprogramm, das einen Block aufzeichnet, hat folgende Form:

Zunächst wird ein Byte 3CH aufgezeichnet, dann die Blocklänge aus dem B-Register. Dann folgen das niederwertige und höherwertige Byte der Startadresse des Blocks, dann alle Datenbytes. Zum Schluß wird noch die Prüfsumme ausgegeben (Siehe nächstes Kapitel).

## Das Format von SYSTEM-Bändern:

Ein Band, das mit dem SYSTEM-Befehl eingelesen werden soll, muß folgendes Format haben:

1. Leader                   Dieser Leader besteht aus 256 Bytes AAH und einem Byte 66H. Der Sinn dieses Leaders ist folgender:  
Jedes Byte wird bitweise auf Kassette aufgezeichnet. Es ist also wichtig, das der Rechner an der richtigen Stelle anfängt, zu lesen, um die richtigen Bits zu einem Byte zusammenzufassen. Dabei hilft der Leader. Der Rechner liest immer ein Bit vom Band und schiebt es durch das A-Register. Solange die Bytes AAH gelesen werden, enthält das A-Register entweder AAH oder 55H, aber nie 66H. In dem Moment, in dem das A-Register 66H enthält, weiß der Rechner, daß er die Bits nun richtig in Gruppen zusammenfassen kann.
2. Ein Byte 55H:       Dieses Byte dient als Kennung für SYSTEM-Bänder.
3. Der Programm-       Die nächsten 6 Bytes geben den Programmnamen  
name                   an. Falls dieser kürzer als 6 Zeichen ist, so sind die letzten Zeichen Leerzeichen.
4. Blockkennung:       Als nächstes Byte folgt die sogenannte Blockkennung. Ein Byte 3CH gibt an, daß nun ein Datenblock folgt, ein Byte 78H gibt an, daß nun die Einsprungsadresse des Programms folgt. Alle anderen Bytes werden ignoriert und es wird weiter nach 3CH oder 78H gesucht.
5. Datenblocks:       Das erste Byte in einem Block gibt die Anzahl von Datenbytes an, die dieser Block enthält. 00H entspricht dabei 256 dez.  
Das zweite und dritte Byte bilden die Adresse, ab der die Datenbytes im Speicher abgelegt werden sollen.  
Dann folgen die Datenbytes und zum Schluß noch ein Byte, die Checksum.  
Diese Checksum ist das niederwertige Byte der Summe aller Datenbytes plus das höher- und niederwertige Byte der Ladeadresse.
6. Einsprung:           Nach dem Byte 78H folgen das niederwertige und das höherwertige Byte der Einsprungsadresse des Programms.

### Das Format von CLOAD-Bändern.

Bändern, die mit dem CLOAD-Befehl geladen werden sollen, müssen folgendes Format haben:

1. Leader
2. Drei Bytes D3H
3. 1 Byte für den Programmnamen
4. Der Programmtext, wie er im Speicher steht. Ende, wenn Link 0000H erreicht.

Wichtig ist, daß der CLOAD-Befehl kein Prüfsummen akzeptiert und also das Gelesene in keiner Weise auf Richtigkeit überprüft.



### Die Begriffe "LSB" und "MSB"

Wir gehen davon aus, daß Ihnen die Begriffe Bit, Byte und Hexadezimalsystem vertraut sind. Sollte dies jedoch nicht der Fall sein, lesen Sie vorab Kapitel 31 des Handbuches "COLOUR BASIC - leicht gelernt".

Was bedeuten nun die Abkürzungen LSB und MSB ?

LSB ist die englische Abkürzung für "Lower Significant Byte", was soviel bedeutet wie niederwertiges Byte. Entsprechend ist MSB die Abkürzung von "More Significant Byte", und bedeutet höherwertiges Byte.

LSB und MSB bilden zusammen eine 16-Bit Adresse mit der alle Speicheradressen angesprochen werden können.

64 K-Byte sind genau 65536 Byte die mit Adressen zwischen 0000H und FFFFH (0 bis 65535) angesprochen werden.

Wenn man z.B. Adresse 44E2H ansprechen will, so ist das MSB=44H und das LSB=E2H.

Die dezimale Schreibweise ergibt sich aus:

$44H = 68 \text{ dez.}$  und  $E2H = 226 \text{ dez.}$ , also  $68 * 256 + 226 = 17634$ .

Man multipliziert also das MSB mit 256 und addiert das LSB hinzu.

In der Z-80 Maschinensprache werden die beiden Adreßbytes in der Reihenfolge LSB/MSB abgearbeitet. Also z.B. JP 0066H muß als C3 66 00 eingegeben werden.

Der Basic-Interpreter verhält sich beim Gebrauch von Adressen und Variablen genauso (siehe auch "Links" und "Zeilennummern" im folgenden Kapitel).

## Wie werden Basicprogramme abgespeichert ?

Sie kennen natürlich den Basic-Befehl "LIST". Dieser Befehl zeigt Ihnen in klar lesbarer Form das Basicprogramm an, das im Speicher steht.

Wie aber "merkt" sich der Computer das Programm intern - so wie Sie es auf dem Bildschirm sehen oder etwa anders? Und woher weiß das Colour-Genie, wo es das Programm abspeichern soll?

Diesen Fragenkomplex wollen wir nun klären. Am Ende finden Sie dann eine interessante Anwendung.

### 1. Die Basicprogramm-Anfangsadresse:

In den beiden Speicherzellen 40A4H und 40A5H (16548 und 16549 dez.) steht die Adresse, bei der das Basicprogramm im Speicher beginnt.

Haben Sie beim Einschalten nicht <MOD SEL> niedergehalten, beginnt das Basicprogramm bei 5801H (22529 dez.).

Dies läßt sich einfach überprüfen:

```
READY
>PRINT PEEK(&H40A4)+256*PEEK(&H40A5)
22529
READY
>PRINT &H5801
22529
READY
>
```

Wenn Sie beim Einschalten die <MOD SEL>-Taste gedrückt haben, d.h. wenn der Speicher von 4800H (18432 dez.) bis 57FFH (22527 dez.) nicht für die hochauflösende Grafik verwendet werden soll, beginnt das Basicprogramm bei Adresse 4801H (18433 dez.).

Auch dies läßt sich leicht überprüfen:

```
READY
>PRINT (PEEK(&H40A4)+256*PEEK(&H40A5))=&H4801
-1
```

(Die -1 bedeutet, daß obige logische Aussage wahr ist.)

### 2. Die Speicherung eines Basicprogramms:

Wir wissen nun, ab welcher Adresse ein Basicprogramm im Speicher beginnt.

Die Frage ist nun: Wie wird das Programm dort gespeichert. Um dies zu untersuchen, laden Sie den "Basic-Monitor", den weiter vorne aufgelistet finden, und den Sie auf ein Band gespeichert haben sollten. Wenn Sie dies noch nicht getan haben, holen Sie dies jetzt nach.

Wir wollen nun mit dem "Basic-Monitor" untersuchen, wie der Monitor selbst abgespeichert wird.

Wenn Sie beim Einschalten nicht <MOD SEL> gedrückt hatten, beginnt das Programm im Speicher bei Adresse 5801H. Starten Sie nun den "Basic-Monitor" mit "RUN". Geben Sie dann "H5801" ein - der Monitor macht nun ein hexadezimal Listing. Brechen Sie dieses nach der 7. Zeile mit <CLEAR> ab. Anschließend können Sie noch ein ASCII-Listing dieses Speicherbereiches machen. Dazu geben Sie das Kommando "A5801" ein. Dieses Listing können Sie nach der 3. Zeile ebenfalls mit <CLEAR> abbrechen. Ihr Bildschirm sieht dann etwa so aus (Die Linien sind nachträglich zur Orientierung eingezeichnet; darunter sind hier noch die ersten vier Zeilen des "Basic-Monitors" zum Vergleich aufgelistet) :

```
Command? H5801
5801 07 58 0A 00 84 00 12 58 57
5809 14 00 B8 20 31 30 30 30 AD
5811 00 22 58 1E 00 B2 22 43 AF
5819 6F 6D 6D 61 6E 64 22 38 D9
5821 00 28 58 28 00 89 20 41 95
5829 24 00 39 58 32 00 8F 41 B7
5831 24 05 22 22 CA 33 30 00 6A
Command? A5801
5801 .X...X... 1000."X... "C
5819 ommand";.+X( . A$.9X2. A
5831 $#"30.BX<. 690.^XF. B$
Command?
```

```
10 CLS
20 CLEAR 1000
30 PRINT"Command";
40 INPUT A$
```

Jede Basiczeile hat im Speicher nun folgendes Format:

1. 2 Bytes, die die Anfangsadresse der nächsten Zeile angeben. ("Link")
2. 2 Bytes, die die Zeilennummer in hexadezimal enthalten.
3. Der Text der Zeile, wobei alle Befehle in sogenannte "Tokens" verschlüsselt werden. D.h., daß ein Befehl nicht als Zeichenfolge gespeichert wird, sondern als ein oder zwei Bytes. Eine Liste dieser Tokens mit den zugehörigen Befehlen finden Sie in Anhang A. Alles, was kein Befehl ist, wird in normaler Textform abgespeichert.
4. Ein Byte 00H als Zeilen-End-Kennung.

Das Programmende ist durch einen Link markiert, der 00 00 ist.

Dies klingt alles recht kompliziert, aber anhand des oben ausgedruckten Beispiels wollen wir Ihnen das Prinzip nun verdeutlichen.

Sehen wir uns zuerst einmal die erste Basic Zeile an:

10 CLS

Im Speicher steht:

5801: 07 58 Dies ist der Link, d.h. die nächste Zeile  
beginnt bei Adresse 5807H.  
Beachten Sie, daß die Adresse 5807H, wie  
üblich, als 07 58 gespeichert (siehe voriges  
Kapitel LSB/MSB).  
5803: 0A 00 Dies ist die Zeilennummer: 000AH = 10 dez.  
5805: 84 Dies ist das Token für den Befehl CLS,  
wie Sie anhand Anhang A und Anhang B leicht  
überprüfen können. (84H = 132 dez.)  
5806: 00 Ende der Zeile

Genauso läßt sich die zweite Programmzeile untersuchen:

20 CLEAR 1000

Im Speicher steht:

5807: 12 58 Die nächste Zeile beginnt bei Adresse 5812H.  
5809: 14 00 Zeilennummer: 0014H = 20 dez.  
580B: B8 Token für CLEAR (B8H = 184 dez.).  
580C: 20 31 30 30 30 ASCII-Text ' 1000'.  
5811: 00 Zeilenende.

Analog dazu können Sie auch die zwei nächsten Zeilen  
erkennen, sowie natürlich jede andere Basic-Zeile auch.

Wie Sie Anhang A entnehmen können, sind die Dezimalwerte  
aller Tokens größer als 127, d.h. ein Token ist durch ein  
gesetztes Bit Nummer 7 gekennzeichnet (Im Gegensatz zu den  
ASCII-Zeichen, die alle einen Wert kleiner als 128 haben -  
vergleiche dazu Anhang B).

Daraus ergibt sich, daß mit einem Byte nur maximal 128  
Befehle codiert werden können, da das 7. Bit ja immer  
gesetzt sein muß. Beim großen Befehlssatz des Colour-Genies  
reichte dies jedoch nicht aus. Aus diesem Grunde wurde für  
eine Reihe von Befehlen eine Codierung mit zwei Bytes  
gewählt, wobei das erste Byte immer FFH (255 dez.) ist.  
Der Befehl PLOT hat z.B. das Doppeltoken FFH 84H (255 dez.  
132 dez.).

In diesem Zusammenhang seien noch drei Besonderheiten  
der Abspeicherung erwähnt:

1. Beim Befehl ELSE wird intern vor dem ELSE noch ein  
Doppelpunkt abgespeichert, der beim LISTen aber nicht  
ausgegeben wird.
2. Der Befehl FCOLOUR wird durch 3 Bytes ausgedrückt:  
FFH 81H 52H (255 dez. 129 dez. 82 dez.)
3. Die Kurzform des REM-Befehls (') wird auf folgende Weise  
abgespeichert:  
3AH 93H FBH (58 dez. 147 dez. 251 dez.)

Als letztes wollen wir nun eine interessante Anwendung für das in diesem Kapitel Erklärte bringen. Es kommt öfters vor, daß man in einem Basicprogramm Teile des Programms selbst ändern möchte. Dies ist nicht ohne weiteres möglich - normalerweise muß man die entsprechende Zeile erst eingeben bzw. editieren und muß dann das Programm neu starten.

Wenn man allerdings weiß, wie Basicprogramme abgespeichert werden, kann man das Programm von sich aus ändern. Ein wesentlicher Vorteil neben dem Eingabe-Komfort ist, daß bei dieser Methode keine Variablenwerte gelöscht werden. Folgendes Programm ist nun ein Beispiel für dieses Verfahren. Das Programm fragt nach einer mathematischen Funktion. Diese geben Sie ganz normal ein und sie wird dann in das Programm "gePOKEd". Anschließend werden noch Unter- u. Obergrenze, sowie die Schrittweite abgefragt. Das Programm berechnet dann eine Wertetabelle für die Funktion. Natürlich gibt es eine Menge anderer Anwendungen - lassen Sie mal Ihre Phantasie spielen ! Wichtig kann die Kenntnis des Abspeicherverfahrens auch sein, wenn man ein kaputtes Basicprogramm, in dem z.B. falsche Zeilennummern vorkommen, reparieren will.

#### Programmlisting:

```

10 CLEAR500:DEFINT A-V:CLS:GOTO30
20 Y=SIN(X):RETURN:REM::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
30 INPUT"Funktion: Y = ";F$
40 F$="Y="+F$
50 T$=""
60 PRINT$40,F$;CHR$(30):X=6040:T=205
70 IFX=6095THEN130ELSER$=CHR$(PEEK(X)AND127):X=X+1
80 IFPEEK(X)>127THEN100
90 R$=R$+CHR$(PEEK(X)):X=X+1:GOTO80
100 IFLEN(F$)<LEN(R$)THENT=T+1:GOTO70
110 IFLEFT$(F$,LEN(R$))<>R$THENT=T+1:GOTO70
120 T$=T$+CHR$(T):F$=RIGHT$(F$,LEN(F$)-LEN(R$)):IFF$<>""THEN60ELSE140
130 T$=T$+LEFT$(F$,1):F$=RIGHT$(F$,LEN(F$)-1):IFF$<>""THEN60
140 PRINT$40,,:T$=T$+"":+CHR$(146)+"":+CHR$(147)
150 X=PEEK(&H40A4)+256*PEEK(&H40A5)
160 X1=PEEK(X)+256*PEEK(X+1)
170 IFPEEK(X1+2)+256*PEEK(X1+3)<>20THENX=X1:GOTO160
180 X1=X1+4
190 FORX=1TOLEN(T$):POKE X1,ASC(MID$(T$,X,1)):X1=X1+1:NEXTX
200 ONERRORGOTO210:GOSUB20:ONERRORGOTO0:GOTO230
210 IFERR=50RERR=11THEN230ELSECOLOUR3:PRINT"Fehlerhafte Funktion":CALL
357C:COLOUR2
220 IFINKEY$=""THEN220ELSERUN
230 INPUT"Untergrenze";XU
240 INPUT"Obergrenze";XO
250 IFXU>XOTHEN230
260 INPUT"Schrittweite";XS
270 IFXS<=0THEN230
280 FORX=XUTOXOSTEPXS
290 GOSUB20:PRINT"F("X;") =";Y
300 IFINKEY$=""THEN300ELSENEXTX
310 IFINKEY$=""THEN310ELSERUN

```

## Erklärung des Programms:

Zeile 10 : Genug Platz für Strings schaffen, zur Beschleunigung des Programmablaufs alle nicht benutzten Variablen als Integer erklären, den Bildschirm löschen und das eigentliche Programm starten.

Zeile 20 : In dieser Zeile soll die Funktion als Unterprogramm abgelegt werden. Die ':' halten den Platz für längere Funktionen frei. Das Unterprogramm besteht immer aus der Berechnung eines Wertes Y aus einer Funktion von X. Das REM verhindert, daß etwaige Reste längerer Funktionen zu einem SYNTAX Error führen.

Zeile 30 : Die Funktion wird in F\$ eingelesen.

Zeile 40 : Der erste Teil des Unterprogramms ist  $Y=f(X)$ .

Zeile 50 : In T\$ wird die codierte Zeile generiert.

Zeile 60 : Den noch zu codierenden Rest der Funktion ausgeben. X zeigt auf den Anfang einer Tabelle im ROM, in der sämtliche BASIC-Schlüsselworte stehen (diese Tabelle beginnt bei 1650H, in diesem Fall wurde jedoch erst bei den arithmetischen Schlüsselworten begonnen). T entspricht dem Token des momentanen Schlüsselwortes.

Zeile 70 : Wenn  $X = 6095$ , dann ist das Ende der arithmetischen Schlüsselworte erreicht, also wurde kein passendes gefunden. Weiter Verarbeitung in Zeile 130.

Zeile 70 (Rest) - Zeile 90 :

Das momentane Schlüsselwort wird in R\$ eingelesen. Der erste Buchstabe des nächsten Schlüsselwortes ist durch ein gesetztes Bit 7 gekennzeichnet, wenn ein solches Zeichen gefunden wird, enthält R\$ das komplette Schlüsselwort.

Zeile 100: Wenn der noch zu codierende Rest der Funktion kürzer als das gefundene Schlüsselwort ist, kann es nicht passen. Nächstes Schlüsselwort versuchen!

Zeile 110: Falls der Rest der Funktion das Schlüsselwort nicht enthält, nächstes Wort versuchen!

Zeile 120: Passendes Wort gefunden. Token an T\$ anhängen. F\$ entsprechend kürzen. Wenn FS noch nicht leer ist, weitermachen, sonst ist die Codierung beendet.

Zeile 130: Diese Zeile wird angesprungen, wenn kein Token passt. Ein Zeichen aus F\$ wird in T\$ übertragen. F\$ wird gekürzt. Wenn F\$ noch nicht leer ist, weitermachen, sonst ist die Codierung beendet.

Zeile 140: An T\$ wird noch der Code für ':RETURN:REM' angehängt, um die Basic-Zeile zu komplettieren.

Zeile 150: X zeigt auf die erste Programmzeile.

Zeile 160 - Zeile 170:

Zeile 20 wird im Programm gesucht. Die Anfangsadresse steht dann in X1.

Zeile 180: Link und Zeilennummer werden übersprungen.

Zeile 190: T\$ wird in Zeile 20 kopiert.

Zeile 200 - Zeile 220 :

Zeile 20 wird auf Fehler getestet. Die Zeile wird ausgeführt (GOSUB 20). Wenn kein Fehler auftaucht, wird zu Zeile 230 gesprungen, vorher jedoch noch die Fehlerabfrage abgestellt (ON ERROR GOTO 0:GOTO 230). Wenn ein Fehler auftaucht, so darf dies nur der FC-Error (Fehlercode 5) oder der /O-Error (Fehlercode 11) sein, sonst ist die Funktion fehlerhaft.

Zeile 230: In XU wird die Untergrenze eingelesen.

Zeile 240: In XO wird die Obergrenze eingelesen.

Zeile 250: Falls die Untergrenze größer oder gleich der Obergrenze ist, wird die Eingabe wiederholt.

Zeile 260: In XS wird die Schrittweite eingelesen.

Zeile 270: Falls die Schrittweite kleiner oder gleich Null ist, wird die Eingabe wiederholt.

Zeile 280: Schleife von XU bis XO mit der Schrittweite XS.

Zeile 290: Aus X einen Y Wert berechnen und ausgeben.

Zeile 300: Auf einen Tastendruck warten, dann nächsten Wert berechnen und ausgeben.

Zeile 310: Nach Beendigung der Schleife auf Tastendruck warten, dann Programm neu starten.

## Wie werden Basic-Variablen abgespeichert?

Die Basic-Variablen werden in zwei Untergruppen aufgeteilt, nämlich:

- die normalen Variablen ( z.B. A = 0 ) und die
- Feldvariablen ( z.B. DIM A(100) )

Die normalen Variablen werden direkt hinter dem Basic-Programm abgelegt (dies ist auch der Grund, weshalb nach jeder Veränderung des Programms alle Variablen gelöscht sind, denn das Programm kann ja dadurch seine Länge verändern). Die Anfangsadresse dieses Speicherbereichs (und damit natürlich auch die Endadresse Ihres Basicprogramms) steht in den Speicherzellen 40F9H/40FAH (16633/16634 dez.). In diesem Speicherbereich stehen die einzelnen Variablen nun einfach hintereinander, und zwar in folgendem Format:

- |                       |  |
|-----------------------|--|
| 1 Byte Typcode        | Dieses Byte ist so gewählt, daß es auch die Länge des Datenfeldes der Variable angibt:<br>02 = Integer<br>03 = String<br>04 = einfache Genauigkeit<br>08 = doppelte Genauigkeit  |
| 2 Bytes Name          | Der Name wird umgekehrt abgespeichert, das heißt, die Variable A1 heißt im Speicher 1A. Einbuchstabile Variablen wie A, B u.ä. werden mit einem Byte 00H an erster Stelle abgespeichert (z.B. 00H 41H = Variable A). Der Sinn dieser Methode ist, daß in jedem Fall ein Kennbuchstabe direkt vor dem Datenfeld steht, dies erleichtert den Vorgang des Suchens nach einer bestimmten Variable. |
| 2 - 8 Bytes Datenfeld | Je nach Typ folgen nun die der Variable zugeordneten Daten:  |

### Integer

Der Wert in LSB/MSB Format.

### Strings

Das erste Byte gibt die Länge des Strings an, die beiden anderen, wo der String im Speicher beginnt.



### Einfache Genauigkeit

Die ersten drei Bytes enthalten die Mantisse, das vierte Byte den Exponenten. Die Mantisse ist folgendermaßen strukturiert:

Das 3. Byte ist das höchstwertigste, dann folgt das zweite und das erste Byte ist das niederwertigste. Diese drei Bytes zusammen ergeben eine 24-bit Zahl, wobei das höchste Bit das Vorzeichen der Mantisse angibt ( 0 = positiv ). Das höchste Bit des Exponenten gibt an, ob der Binärpunkt nach rechts oder nach links verschoben wird ( 0 = rechts ), die restlichen sieben Bits geben an, um wieviele Stellen der Punkt verschoben wird.

### Doppelte Genauigkeit

Die Struktur entspricht genau der oben geschilderten, doch die Mantisse ist insgesamt 56 Bits lang.

Nach der Variablentabelle folgt der Speicherbereich, in dem die Felder abgelegt werden. Der Beginn dieses Bereiches steht in den Adressen 40FBH/40FCH (16635/16636 dez.). Der Eintrag für ein Feld sieht nun folgendermaßen aus:

1 Byte Typcode	Wie bei Variablen: 02 = Integer 03 = String 04 = Einfache Genauigkeit 08 = Doppelte Genauigkeit
2 Bytes Name	Die Abspeicherung des Namens entspricht der bei normalen Variablen.
2 Bytes Länge	Die folgenden zwei Bytes geben an, wieviel Speicherplatz das Feld benötigt. Wenn man zu der Adresse, in der das erste dieser beiden Bytes steht, diese Länge + 2 addiert, erhält man die Adresse, bei der das nächste Feld im Speicher beginnt.
1 Byte Dimension	Dieses Byte gibt an, wieviele Dimension das Feld hat.

n \* 2 Bytes  
Dimension

Nun folgt für jede Dimension deren Länge. Da jede Dimension von 0 bis zum maximalen Index angesprochen werden kann, steht hier der maximale Index + 1.

Datenfeld

Schließlich folgen die Daten, die das Feld enthält. Ihr Format entspricht dem der Daten von normalen Variablen.

Die Daten sind folgendermaßen sortiert:  
Gehen wir von dem Feld A(2,1) aus. Dann steht im Speicher:

A(0,0); A(1,0); A(2,0); A(0,1); A(1,1);  
A(2,1)

Das Ende des Speicherbereichs, in dem die Felder abgelegt werden und somit der Beginn des freien Speichers steht in den Adressen 40FDH/40FEH (16637/16638 dez.)

### Zusammenladen von mehreren Basicprogrammen

Sie kennen sicher das Problem, daß man zwei Basicprogramme aneinanderhängen möchte, ohne einen der beiden Teile neu eintippen zu müssen.

Abhilfe schafft folgendes Vorgehen:

1. Das Basicprogramm mit den kleinsten Zeilennummern mit CLOAD laden.
2. Eventuell Programm ändern...
3. Folgende Zeile eingeben (ohne Zeilennummer!):  
A=PEEK(&H40F9)+256\*PEEK(&H40FA):A=A-2:POKE&H40A4,A-INT(A/256)  
) \*256:POKE&H40A5,INT(A/256)  
(Zur Erklärung der Adressen 40F9H/40FAH siehe Kapitel:  
"Wie werden Variablen abgespeichert?")
4. Nächstes Basicprogramm mit CLOAD laden. Wichtig ist, daß dieses Programm keine Zeilennummer enthält, die kleiner oder gleich der höchsten Zeilennummer ist, das als letztes geladen wurde.
5. Schritte 2 bis 4 sooft wiederholen, bis alle Teilprogramme einmal geladen wurden.
6. Folgende Zeile eingeben (ohne Zeilennummer!):  
POKE&H40A4,1:POKE&H40A5,88
7. Nun sind alle Programme vorhanden und Sie können mit Ihnen arbeiten.

Bitte beachten Sie:

Schritt Nr. 6 gilt nur, wenn Sie beim Einschalten nicht die MOD SEL-Taste gedrückt hatten und keine Programme wie den Colour-Compiler oder den Colour-Assembler geladen und gestartet haben. Dies alles verändert die Startadresse des Basicprogramms (siehe Kapitel "Wie werden Basicprogramme abgespeichert?"). In einem solchen Fall geben Sie vor Schritt 1 folgende Zeile ein (ohne Zeilennummer):

```
PRINTPEEK(&H40A4);PEEK(&H40A5)
```

und merken sich die beiden Zahlen, die nun ausgegeben werden. In Schritt Nr. 6 tauschen Sie nun die 1 gegen die erste und die 88 gegen die zweite der beiden Zahlen aus.

### Reserviert

Sie hassen doch sicher auch die Situation, daß ein Programm Ihnen anzeigt: "Bitte geben Sie bei MEM SIZE? 32000 ein." Mit den folgenden Zeilen können Sie in Ihren Basicprogrammen eine Speicherobergrenze für das Basic neu festlegen. Sinnvoll ist dies, wenn man, kombiniert mit dem Basic, Maschinenspracheprogramme verwendet.

```
10 CLEAR 50
20 HA=PEEK(&H40B1)+256*PEEK(&H40B2)
30 HA=32000:'Je nach Bedarf festlegen
40 H2=INT(HA/256):H1=HA-H2*256
50 POKE&H40B1,H1:POKE&H40B2,H2
60 POKE&H40D6,H1:POKE&H40D7,H2
70 CLEAR50
80 REM Hier folgt Ihr Basicprogramm
```

In Zeile 20 wird die alte Speicherobergrenze gelesen.  
In Zeile 30 wird die neue Speicherobergrenze festgelegt.  
(Hier 32000: Ihren Wünschen entsprechend ändern)  
In Zeile 40 wird die 2-Byte-Integerzahl für die neue Obergrenze in zwei einzelne Bytes zerlegt.  
In Zeile 50 und 60 werden diese neuen Werte in die Basicspeicheradresse und Stringvariablenadresse geschrieben.  
Die Zeilen 10 und 70 sind notwendig und dürfen nicht entfernt werden.

### Ausgabe der Bytes 0,11 oder 12 an den Drucker

Das Basic-ROM des Colour-Genie wandelt die Bytes 11 dez. (Top of form) und 12 dez. (Formfeed) in eine Folge von 10er (dez.) Bytes (Linefeeds) um.

Das Byte 0 wird überhaupt nicht ausgegeben.

(Zum Vergleich können Sie die entsprechende ROM-Routine mit einem Disassembler listen: Sie beginnt bei Adresse 04E7H und endet bei Adresse 0563H.)

Das heißt also, daß die Befehle LPRINT CHR\$(0), LPRINT CHR\$(11) und LPRINT CHR\$(12) nicht die Werte 0,11 und 12 an den Drucker ausgeben!

Dies kann, je nach angeschlossenem Drucker, zu einigen Problemen führen. So kommt es bei der Programmierung von hochauflösenden Grafiken über Einzelnadelsteuerung (Bit-Image-Grafik) oft zu falschen Ausdrucken. Ferner ist die Tabulatorprogrammierung des Druckers oft beschränkt, da Byte 11 (dez.) das Steuerzeichen für einen vertikalen Tabulator ist. Eine anderes Problem ist z.B., daß man beim Star-Drucker DP 510 die Unterstreichung hilfs eines Bytes 00 abschaltet. Alle diese Probleme kann man mit folgenden Basic-Zeilen lösen, die mit GOSUB 100 aufgerufen werden. Der ASCII-Wert des auszugebenden Zeichens steht dabei in Variable X.

```
100 SOUND 7,127 : OUT248,15 : IF (INP(249) AND 239) = 47 THEN
    110 ELSE 100
110 SOUND 7,127 : SOUND 14,X : SOUND 7,255 : SOUND 15,0 : SOUND
    15,1 : RETURN
```

Zeile 100 prüft, ob der Drucker druckbereit ist. In Zeile 110 wird der Wert X an den Drucker ausgegeben. (Zur Ausgabe von Buchstaben vergleiche Anhang B.)

Natürlich können Sie die Zeilennummern so ändern, daß Sie in Ihr Programm passen. Als Anwendungsbeispiel für obige Zeilen folgt nun ein kleines Programm, daß auf dem STAR-Drucker DP 510/DP 515 ein kleines Pferd ausdruckt:



Das Programm finden Sie auf der nächsten Seite aufgelistet. Beachten Sie, daß das Programm nur auf dem STAR-Drucker DP 510/515 läuft. In den Programmzeilen 260 und 270 finden Sie übrigens die oben aufgeführten Ausgabezeilen.



### Ausgabe von Tabulatoren größer als 40 auf den Drucker

Der Basic-Interpreter Ihres Colour-Genie ist so aufgebaut, daß Sie an jeder Stelle einer Bildschirmzeile eine Tabulatormarke setzen können.

Analog dazu behandelt der Interpreter den Drucker.

Beachten Sie aber:

Eine Bildschirmzeile hat 40 Zeichen. Daher ist das Setzen einer Tabulatormarke auf eine Position > 39 mit dem TAB-Befehl auch auf dem Drucker nicht möglich. Benutzen Sie in solchen Fällen

statt:           10 LPRINT TAB(T);"Text"

folgendes:       10 LPRINT STRING\$(T-PEEK(&H409B),32);"Text"

Wie beim TAB-Befehl gilt natürlich auch hier:

$0 \leq T \leq \text{maximale Zeilenlänge} - 1$ .

Beachten Sie, daß am Anfang eines Programms, in dem diese Methode benutzt wird, ein CLEAR Z stehen muß, wobei Z  $\geq$  Zeilenlänge sein muß. Sonst könnte ein OS (Out of Stringspace) Error auftreten.

### Interessante ROM-CALL's

Im folgenden sind einige nützliche ROM-Routinen aufgelistet und kurz erklärt, die dem Programmierer, der Erfahrung in Maschinenspracheprogrammierung hat, viel Arbeit abnehmen können. Am besten schaut man sich die Routinen, die beschrieben sind, mit dem Disassembler eines Maschinensprache-Monitors an. Im übrigen sei erwähnt, daß nur die wichtigsten Routinen aufgeführt sind, da in nächster Zeit ein komplettes, dokumentiertes ROM-Listing zur Verfügung stehen wird.

- 0000H    System-Kaltstart
- 0008H    Basic Syntax-Check. Wird über RST 08 aufgerufen  
Das Byte hinter dem RST 08 Code wird mit (HL)  
verglichen. Wenn ungleich dann "?SN Error".
- 0018H    CP HL,DE . Vergleicht das Doppelregister DE mit HL.  
HL > DE    C=0    Z=0  
HL = DE    C=0    Z=1  
HL < DE    C=1    Z=0
- 002BH    INCH. Liest einen Tastendruck in Akku. A=0 keine Taste.
- 0033H    OUTCH. Gibt den Akkuinhalt als Zeichen auf die aktuelle  
Bildschirmposition in 4020H/4021H aus. Zeichen kleiner  
20H werden als Control-Zeichen akzeptiert (Cursor Home,  
Clear to End of Line usw.) und bei Bedarf wird der  
Bildschirminhalt gescrollt.
- 003BH    PRTCHR. Gibt das Zeichen in A auf den Drucker aus.
- 0040H    INLINE. Liest ab HL als Bufferadresse (norm. 41E8H)  
maximal B Zeichen von der Tastatur ein. Das letzte  
Zeichen ist 0DH (RETURN). Bei BREAK ist das Carryflag  
gesetzt. Bei RET steht HL wieder auf Buffer und B ent-  
hält die Zahl der eingegebenen Zeichen.
- 0049H    Wie INCH, jedoch wird auf Tastendruck gewartet.
- 0060H    Wartet BC-Registerinhalt \* 14.66 µs.
- 0066H    Basic-Warmstart. Legt den Basic-Stackpointer neu an und  
springt anschließend wieder zum Basic ">READY".
- 01C9H    CLS. Löscht Bildschirm über 1CH = Cursor Home und  
1FH = Clear to End of Frame Codes.
- 01D3H    RANDOM. Erzeugt neue RND Werte.



01E4H Läßt das rechte '\*' blinken durch XOR 0AH .  
 01EDH Read Byte. Ein Byte über Kassettenport in A lesen.  
 021FH Write Byte. Akkuinhalt auf Kassettenport schreiben.  
 023FH Write Sync. 256 \* AAH und einmal 66H auf Kassettenport.  
 024CH Read Sync. Wartet auf Synchronisation.  
 0314H Liest ein Doppelbyte LSB/MSB von Kassette in HL.  
 0A7FH CINT. Überträgt die Basicvariable X (4121H/4122H) in das HL Registerpaar.  
 0A9AH Überträgt HL in die Variable X. (Anm.: X ist eine Variable, die vom Basic-Interpreter in den Adressen 4121H ff. angelegt wird und keine Variable des normalen Basic's !!!)  
 0FAFH LINENR. Gibt HL als dez. Zahl ohne Vorzeichen aus (z.B. Zeilennummer).  
 1650H ASCII Tabelle der Basic-Tokens. Erstes Byte jeweils mit gesetztem Bit 7. Sortiert nach aufsteigenden Tokens.  
 1822H Sprungtabelle zu den Tokens. Enthält die Einsprungsadressen der entsprechenden Routinen als Doppelbytes.  
 2B01H Verwandelt Ausdruck ab (HL) in eine Integerzahl in DE. Nachher steht HL auf dem ersten Byte nach dem Ausdruck. Dient zur Übernahme von Zahlen oder Ausdrücken in Maschinensprache.  
 2B1BH Verwandelt Ausdruck ab (HL) in eine Zahl < 256 dez. Wert steht hinterher im Akku, sonst wie 2B01H.  
 2B75H Print String ab (HL). Bei (HL)=0 RET.  
 38A9H FGR. Schaltet Bildschirm auf FGR um.  
 38B0H LGR. Schaltet auf LGR um.  
 38CCH COLOUR n. Wobei n-1 im Akku stehen muß (0...F).  
 38DAH FCOLOUR n. Wobei n-1 im Akku stehen muß (0...3).  
 3FBOH CHAR n. Schaltet auf CHAR n. Im Akku muß n-1 stehen.

*Änderungen der Break-Port-Tabelle siehe Anlage*

### \*\*\* Neuer Zeichensatz \*\*\*

Hat Sie der eingebaute Zeichensatz Ihres Colour-Genie auch schon einmal gelangweilt? Dann geben Sie mit dem "Basic-Monitor" oder einem anderen Monitor das nachfolgende Hex-Listing ein (Die Prüfsummen am Ende jeder Zeile werden nicht mit eingegeben). Bevor Sie jedoch den Monitor laden, müssen Sie bei MEM SIZE den Wert 31936 eingeben, um das Programm zu schützen. Nachdem Sie das Programm eingegeben und alle Prüfsummen verglichen haben, schreiben Sie alles mit dem W-Befehl auf Kasette: W7CCB 7FFF 7CD2. Dann können Sie das Zeichensatzprogramm mit G7CD2 starten. Nun werden alle ASCII-Zeichen in der neuen Schrift ausgegeben. Da die neue Schrift mit den definierbaren Zeichen erzeugt wird, dürfen Sie nur die Zeichen 128 bis 159 undefinieren und müssen immer auf CHAR 1 bleiben. Den normalen Zeichensatz können Sie weiterhin darstellen, indem Sie den Wert des Zeichens in den Bildschirm POKEn.

#### Erklärung der Funktionsweise des Programms

Nach dem im folgenden abgedruckten Hex-Listing des Programmes, finden Sie ein disassembliertes Listing der Routine, die den neuen Zeichensatz initialisiert.

Bei 7CD2H werden HL, DE und BC für einen Transfer eines kleinen Programms von 7CC0H bis 7CD1H nach 43A0H vorbereitet. Dieses kleine Programm wird über den PRINT-DCB (Device-Control-Block) angesprungen, dessen Zeiger nach 43A0H 'verbogen' wird.

Wenn das Programm angesprungen wird, steht im C-Register des Z-80 der Wert des auszugebenden ASCII-Zeichens.

Wenn er kleiner als 20H oder größer als 7FH ist, bleibt er erhalten. Andernfalls wird 80H dazuaddiert, d.h. aus dem Zeichen '!' daß den ASCII-Code 21H hat wird also in A1H umgewandelt, während z.B. ein Cursor-Home-Code 1CH als 1CH erhalten bleibt, damit der Basic-Interpreter ihn noch erkennt.

Bei 7CDDH wird der PRINT-DCB nach Adresse 43A0H verbogen.

Der DCB ist eine Doppelbyte-Adresse, die vom Basic-Interpreter im RAM abgelegt wird. Normalerweise zeigt sie auf eine ROM-Adresse (hier 30E4H), kann aber bei Bedarf geändert werden.

Man kann also den Basic-Interpreter dazu veranlassen, bei einer Zeichenausgabe in ein eigenes Programm zu verzweigen.

Die Adressen der DCB's sind folgende:

401EH = Display                      4016H = Keyboard                      4026H = Printer

Anschließend werden bei Adresse 7CE3H HL, DE und BC für den Transfer des Zeichensatzes von 7D00H bis 7FFFFH nach F500H vorbelegt. Nach dem Transfer mit LDIR wird noch der Zeichensatz (CHARn) über die ROM-Routine bei 3FB0H geändert. Im A-Register steht dabei der CHAR-Wert minus eins. Für CHAR 1 also 0, was hier über XOR A erreicht wird. Zuletzt folgt ein Sprung zum BASIC-Warmstart bei Adresse 0066H. Von nun an werden alle Zeichen im Bereich zwischen 20H und 7FH auf A0H bis FFH erhöht. Wenn Sie wollen können Sie die Zeichen natürlich auch noch abändern, z.B. mit Hilfe eines Zeicheneditors.

7CC0	F5	79	FE	20	38	07	FE	80	49
7CC8	30	03	C6	80	4F	F1	C3	E4	60
7CD0	30	00	21	C0	7C	11	A0	43	81
7CD8	01	11	00	ED	B0	21	A0	43	B3
7CE0	22	1E	40	21	00	7D	11	00	2F
7CE8	F5	01	00	03	ED	B0	AF	CD	12
7CF0	B0	3F	C3	66	00	00	00	00	18
7CF8	00	00	00	00	00	00	00	00	00
7D00	00	00	00	00	00	00	00	00	00
7D08	18	18	18	18	18	00	18	00	90
7D10	24	24	24	00	00	00	00	00	6C
7D18	24	24	7E	24	7E	24	24	00	B0
7D20	18	7E	40	7E	02	7E	18	00	EC
7D28	60	66	0C	18	30	66	06	00	86
7D30	30	48	48	30	6A	64	3A	00	F8
7D38	18	18	18	00	00	00	00	00	48
7D40	0C	18	30	30	30	18	0C	00	D8
7D48	30	18	0C	0C	0C	18	30	00	B4
7D50	18	5A	3C	18	3C	5A	18	00	74
7D58	00	18	18	7E	18	18	00	00	DE
7D60	00	00	00	00	18	18	30	00	60
7D68	00	00	00	7E	00	00	00	00	7E
7D70	00	00	00	00	00	18	18	00	30
7D78	00	04	08	10	30	60	00	00	AC
7D80	7C	44	4C	54	62	62	7E	00	A2
7D88	10	10	10	10	18	18	18	00	88
7D90	7C	04	04	7C	60	60	7E	00	3E
7D98	7C	04	04	1E	06	06	7E	00	2C
7DA0	42	42	42	7E	06	06	06	00	56
7DA8	7E	40	40	7E	06	06	7E	00	06
7DB0	40	40	40	7E	62	62	7E	00	80
7DB8	7E	02	02	02	06	06	06	00	96
7DC0	7C	44	44	3C	62	62	7E	00	82
7DC8	7E	42	42	7E	06	06	06	00	92
7DD0	00	00	18	00	18	00	00	00	30
7DD8	00	00	18	00	18	18	30	00	78
7DE0	0C	18	30	60	30	18	0C	00	08
7DE8	00	00	7E	00	7E	00	00	00	FC
7DF0	60	30	18	0C	18	30	60	00	5C
7DF8	3C	66	06	0C	18	00	18	00	E4
7E00	7E	42	5E	4C	60	62	7E	00	AA
7E08	7E	42	42	7E	62	62	62	00	A6
7E10	7E	42	42	7C	62	62	7E	00	C0
7E18	7E	42	40	40	60	62	7E	00	80
7E20	7C	46	42	42	62	66	7C	00	8A
7E28	7E	40	40	78	60	60	7E	00	B4
7E30	7E	40	40	78	60	60	60	00	96
7E38	7E	42	40	4E	62	62	7E	00	90
7E40	42	42	42	7E	62	62	62	00	6A
7E48	10	10	10	10	18	18	18	00	88
7E50	7E	42	02	02	06	46	7E	00	8E
7E58	42	44	48	78	64	62	62	00	6E
7E60	40	40	40	40	60	60	7E	00	3E
7E68	76	4A	4A	42	62	62	62	00	72
7E70	72	4A	4A	46	62	62	62	00	72
7E78	7E	42	42	42	62	62	7E	00	86
7E80	7E	42	42	7E	60	60	60	00	A0
7E88	7E	42	42	42	6A	64	7A	00	8C
7E90	7E	42	42	7E	68	64	62	00	AE
7E98	7E	42	40	7E	06	46	7E	00	48
7EA0	7E	52	10	10	18	18	18	00	38
7EA8	42	42	42	42	62	62	7E	00	4A

7EB0	42	42	42	42	64	68	30	00	04
7EB8	42	42	42	42	6A	6A	34	00	10
7EC0	42	42	2C	10	2C	62	62	00	B0
7EC8	42	42	2C	10	18	18	18	00	08
7ED0	7E	42	02	7E	60	62	7E	00	80
7ED8	3C	30	30	30	30	30	3C	00	68
7EE0	00	60	30	18	0C	06	00	00	BA
7EE8	3C	0C	0C	0C	0C	0C	3C	00	B4
7EF0	00	32	4C	00	00	00	00	00	7E
7EF8	00	00	00	00	00	00	7E	7E	FC
7F00	08	18	30	00	00	00	00	00	50
7F08	00	00	7A	46	62	62	7E	00	02
7F10	40	40	7E	42	62	62	7E	00	82
7F18	00	00	7E	40	60	60	7E	00	FC
7F20	02	02	7E	42	62	62	7E	00	06
7F28	00	00	7E	42	7E	60	7E	00	1C
7F30	0C	10	38	10	18	18	18	00	AC
7F38	00	00	7A	46	62	7E	02	7E	20
7F40	40	40	7C	42	62	62	62	00	64
7F48	10	00	10	10	18	18	18	00	78
7F50	08	00	08	08	18	18	18	70	D0
7F58	40	40	44	48	70	68	64	00	48
7F60	10	10	10	10	18	18	18	00	88
7F68	00	00	7C	4A	6A	6A	6A	00	04
7F70	00	00	7C	42	62	62	62	00	E4
7F78	00	00	7E	42	62	62	7E	00	02
7F80	00	00	7E	42	62	7E	40	40	20
7F88	00	00	7E	42	62	7E	02	02	A4
7F90	00	00	7C	42	60	60	60	00	DE
7F98	00	00	7E	40	7E	06	7E	00	C0
7FA0	10	10	38	10	18	18	18	0E	BE
7FA8	00	00	42	42	62	62	7E	00	C6
7FB0	00	00	42	42	64	68	30	00	80
7FB8	00	00	42	42	6A	6A	34	00	8C
7FC0	00	00	42	4C	30	6C	62	00	8C
7FC8	00	00	42	42	62	7E	02	7E	E4
7FD0	00	00	7E	02	7E	60	7E	00	DC
7FD8	0C	18	18	30	18	18	0C	00	A8
7FE0	18	18	18	00	18	18	18	00	90
7FE8	30	18	18	0C	18	18	30	00	CC
7FF0	00	18	3C	66	00	00	00	00	BA
7FF8	FF	FF	FF	FF	FF	FF	FF	FF	F8

7CC0	F5	u	PUSH	AF
7CC1	79	y	LD	A,C
7CC2	FE20	β	CP	20H
7CC4	3807	8	JR	C,\$+09H
7CC6	FE80	β	CP	80H
7CC8	3003	0	JR	NC,\$+05H
7CCA	C680	F	ADD	A,80H
7CCC	4F	0	LD	C,A
7CCD	F1	q	POP	AF
7CCE	C3E430	Cd0	JP	30E4H
7CD1	00		NOP	
7CD2	21C07C	! 8ö	LD	HL,7CC0H
7CD5	11A043	C	LD	DE,43A0H
7CD8	011100		LD	BC,0011H
7CDB	EDB0	m0	LDIR	
7CDD	21A043	! C	LD	HL,43A0H
7CE0	221E40	" 8	LD	(401EH),HL
7CE3	21007D	! ü	LD	HL,7D00H
7CE6	1100F5	u	LD	DE,0F500H
7CE9	010003		LD	BC,0300H
7CEC	EDB0	m0	LDIR	
7CEE	AF	/	XOR	A
7CEF	CDB03F	M0?	CALL	3FB0H
7CF2	C36600	Cf	JP	0066H

## Ein Screen-Printer

Sicher haben Sie sich gefragt, wie die vielen Bildschirmausdrucke, die in diesem Buch zu finden sind, erzeugt wurden.

Das folgende Maschinenspracheprogramm erzeugt einen solchen Bildschirmausdruck auf dem STAR DP 510/515. Dabei sind sowohl Ausdrücke von Texten, als auch von FGR-Grafiken möglich. Außerdem erkennt das Programm, welcher Zeichensatz für die Zeichen von 128 bis 255 angewählt wurde.

Nach dem Einschalten Ihres Rechners geben Sie als MEM SIZE 46330 ein\*. Dann laden Sie den "Basic-Monitor" (oder jeden anderen Monitor, sofern dieser nicht die Adressen über B500H belegt) und geben das Programm mit dem E-Befehl ein. Wenn Sie sicher sind, daß Sie alles fehlerfrei eingegeben haben, schreiben Sie den Screen-Printer mit dem Befehl WB500 BFFF BE40 auf Kassette. Zum Schluß geben Sie ein:  
GBE40.

Der Screen-Printer meldet sich mit:

<SCREEN PRINTER>

(c) 1983 von TCS

COLOUR BASIC

READY

>

Nun können Sie jederzeit durch Druck auf die Tasten <CTRL> und <P> einen Ausdruck des Bildschirms machen. Da dieses Programm den Tastatur-DCB anzapft, funktioniert das Programm immer, solange es nicht überschrieben wird oder die Tastatur-Treiber-Adresse verändert wird.

Hier nun ein Hex-Dump (mit Prüfsummen) des Screen-Printer:

\* (Für dieses Programm muß Ihr Colour-Genie auf 32 K RAM erweitert sein.)

B500	30	FF	FF	FF	FF	FF	FF	FF	29
B508	FF	FF	FF	FF	FF	FF	FF	FF	F8
B510	FF	FF	FF	FF	FF	FF	FF	FF	F8
B518	FF	FF	FF	FF	FF	FF	FF	FF	F8
B520	FF	FF	FF	FF	FF	FF	FF	FF	F8
B528	FF	FF	FF	FF	FF	FF	FF	FF	F8
B530	FF	FF	FF	FF	FF	FF	FF	FF	F8
B538	FF	FF	FF	FF	FF	FF	FF	FF	F8
B540	FF	FF	FF	FF	FF	FF	FF	FF	F8
B548	FF	FF	FF	FF	FF	FF	FF	FF	F8
B550	FF	FF	FF	FF	FF	FF	FF	FF	F8
B558	FF	FF	FF	FF	FF	FF	FF	FF	F8
B560	FF	FF	FF	FF	FF	FF	FF	FF	F8
B568	FF	FF	FF	FF	FF	FF	FF	FF	F8
B570	FF	FF	FF	FF	FF	FF	FF	FF	F8
B578	FF	FF	FF	FF	FF	FF	FF	FF	F8
B580	FF	FF	FF	FF	FF	FF	FF	FF	F8
B588	FF	FF	FF	FF	FF	FF	FF	FF	F8
B590	FF	FF	FF	FF	FF	FF	FF	FF	F8
B598	FF	FF	FF	FF	FF	FF	FF	FF	F8
B5A0	FF	FF	FF	FF	FF	FF	FF	FF	F8
B5A8	FF	FF	FF	FF	FF	FF	FF	FF	F8
B5B0	FF	FF	FF	FF	FF	FF	FF	FF	F8
B5B8	FF	FF	FF	FF	FF	FF	FF	FF	F8
B5C0	FF	FF	FF	FF	FF	FF	FF	FF	F8
B5C8	FF	FF	FF	FF	FF	FF	FF	FF	F8
B5D0	FF	FF	FF	FF	FF	FF	FF	FF	F8
B5D8	FF	FF	FF	FF	FF	FF	FF	FF	F8
B5E0	FF	FF	FF	FF	FF	FF	FF	FF	F8
B5E8	FF	FF	FF	FF	FF	FF	FF	FF	F8
B5F0	FF	FF	FF	FF	FF	FF	FF	FF	F8
B5F8	FF	FF	FF	FF	FF	FF	FF	FF	F8
B600	00	00	00	00	00	00	00	00	00
B608	10	10	10	10	10	00	10	00	60
B610	28	28	28	00	00	00	00	00	78
B618	28	28	7C	28	7C	28	28	00	C0
B620	10	3C	50	38	14	78	10	00	70
B628	60	64	08	10	20	4C	0C	00	54
B630	20	50	50	20	54	48	34	00	B0
B638	10	10	10	00	00	00	00	00	30
B640	08	10	20	20	20	10	08	00	90
B648	20	10	08	08	08	10	20	00	78
B650	10	54	38	10	38	54	10	00	48
B658	00	10	10	7C	10	10	00	00	BC
B660	00	00	00	00	10	10	20	00	40
B668	00	00	00	7C	00	00	00	00	7C
B670	00	00	00	00	00	00	10	00	10
B678	00	04	08	10	20	40	00	00	7C
B680	38	44	4C	54	64	44	38	00	FC
B688	10	30	10	10	10	10	10	00	90
B690	38	44	04	38	40	40	7C	00	B4
B698	7C	04	08	18	04	44	38	00	20
B6A0	18	28	48	7C	08	08	08	00	1C
B6A8	7C	40	78	04	04	44	38	00	B8
B6B0	1C	20	40	78	44	44	38	00	B4
B6B8	7C	04	04	08	10	20	40	00	FC
B6C0	38	44	44	38	44	44	38	00	B8
B6C8	38	44	44	3C	04	08	70	00	78
B6D0	00	00	10	00	10	00	00	00	20
B6D8	00	00	10	00	10	10	20	00	50

B6E0	08	10	20	40	20	10	08	00	B0
B6E8	00	00	7C	00	7C	00	00	00	F8
B6F0	20	10	08	04	08	10	20	00	74
B6F8	38	44	04	08	10	00	10	00	A8
B700	38	44	54	5C	58	40	3C	00	00
B708	10	28	44	44	7C	44	44	00	C4
B710	78	44	44	78	44	44	78	00	78
B718	38	44	40	40	40	44	38	00	B8
B720	78	44	44	44	44	44	78	00	44
B728	7C	40	40	70	40	40	7C	00	68
B730	7C	40	40	70	40	40	40	00	2C
B738	38	44	40	5C	44	44	38	00	D8
B740	44	44	44	7C	44	44	44	00	14
B748	38	10	10	10	10	10	38	00	C0
B750	7C	04	04	04	04	44	38	00	08
B758	44	48	50	60	50	48	44	00	18
B760	40	40	40	40	40	40	7C	00	FC
B768	44	6C	54	54	44	44	44	00	24
B770	44	64	54	4C	44	44	44	00	14
B778	38	44	44	44	44	44	38	00	C4
B780	78	44	44	78	40	40	40	00	38
B788	38	44	44	44	54	48	34	00	D4
B790	78	44	44	78	50	48	44	00	54
B798	38	44	40	38	04	44	38	00	74
B7A0	7C	54	10	10	10	10	10	00	20
B7A8	44	44	44	44	44	44	38	00	D0
B7B0	44	44	44	28	28	10	10	00	3C
B7B8	44	44	44	54	54	54	28	00	F0
B7C0	44	44	28	10	28	44	44	00	70
B7C8	44	44	28	10	10	10	10	00	F0
B7D0	7C	44	08	10	20	44	7C	00	B8
B7D8	78	60	60	60	60	60	78	00	D0
B7E0	00	40	20	10	08	04	00	00	7C
B7E8	3C	0C	0C	0C	0C	0C	3C	00	B4
B7F0	10	28	44	00	00	00	00	00	7C
B7F8	00	00	00	00	00	00	00	7C	7C
B800	20	10	08	00	00	00	00	00	38
B808	00	00	38	04	3C	44	3C	00	F8
B810	40	40	78	44	44	44	78	00	3C
B818	00	00	38	44	40	44	38	00	38
B820	04	04	3C	44	44	44	3C	00	4C
B828	00	00	38	44	7C	40	38	00	70
B830	18	20	70	20	20	20	20	20	48
B838	00	00	3C	44	44	3C	04	38	3C
B840	40	40	78	44	44	44	44	00	08
B848	10	00	30	10	10	10	38	00	A8
B850	08	00	08	08	08	08	48	30	A0
B858	40	40	48	50	60	50	48	00	10
B860	30	10	10	10	10	10	38	00	B8
B868	00	00	68	54	54	54	54	00	B8
B870	00	00	58	64	44	44	44	00	88
B878	00	00	38	44	44	44	38	00	3C
B880	00	00	78	44	44	78	40	40	F8
B888	00	00	3C	44	44	3C	04	04	08
B890	00	00	58	60	40	40	40	00	78
B898	00	00	3C	40	38	04	78	00	30



B8A0	10	10	38	10	10	10	10	08	A0
B8A8	00	00	44	44	44	4C	34	00	4C
B8B0	00	00	44	44	44	28	10	00	04
B8B8	00	00	44	44	54	54	28	00	58
B8C0	00	00	44	28	10	28	44	00	E8
B8C8	00	00	44	44	44	3C	04	38	44
B8D0	00	00	7C	08	10	20	7C	00	30
B8D8	18	20	20	40	20	20	18	00	F0
B8E0	10	10	10	00	10	10	10	00	60
B8E8	30	08	08	04	08	08	30	00	84
B8F0	00	00	32	4C	00	00	00	00	7E
B8F8	FF	FF	FF	FF	FF	FF	FF	FF	F8
B900	FF	FF	FF	07	07	07	07	07	20
B908	FF	FF	FF	E0	E0	E0	E0	E0	5D
B910	07	07	07	07	07	FF	FF	FF	20
B918	E0	E0	E0	E0	E0	FF	FF	FF	5D
B920	1C	1C	FC	FC	FC	FC	1C	1C	60
B928	38	38	3F	3F	3F	3F	38	38	DC
B930	3C	3C	3C	FF	FF	FF	00	00	B1
B938	00	00	FF	FF	FF	3C	3C	3C	B1
B940	00	00	FF	FF	FF	00	00	00	FD
B948	80	80	80	80	80	80	80	80	00
B950	01	01	01	01	01	01	01	01	08
B958	FF	00	00	00	00	00	00	00	FF
B960	00	00	00	00	00	00	00	FF	FF
B968	FF	81	81	81	81	81	81	FF	04
B970	38	38	38	38	38	38	38	38	C0
B978	00	00	00	18	18	00	00	00	30
B980	3C	3C	3C	3C	3C	3C	3C	3C	E0
B988	00	00	FF	FF	FF	FF	00	00	FC
B990	FC	FC	C0	C0	C0	C0	C0	C0	78
B998	3F	3F	3F	3F	03	03	03	03	08
B9A0	C0	C0	30	3C	3C	30	C0	C0	D8
B9A8	03	03	0C	3C	3C	0C	03	03	9C
B9B0	C0	C0	F0	F0	F0	F0	C0	C0	C0
B9B8	03	03	0F	0F	0F	0F	03	03	48
B9C0	00	00	3C	3C	3C	3C	00	00	F0
B9C8	0C	0C	3C	3C	3C	3C	30	30	68
B9D0	00	00	FC	FC	3F	3F	00	00	76
B9D8	C0	C0	C0	C0	C0	FC	FC	FC	B4
B9E0	03	03	03	03	3F	3F	3F	3F	08
B9E8	00	00	18	18	3C	3C	C3	C3	2E
B9F0	C3	C3	3C	3C	18	18	00	00	2E
B9F8	00	00	00	00	3C	3C	FF	FF	76
BA00	FF	FF	3C	3C	00	00	00	00	76
BA08	13	7E	7E	7E	7E	7E	7E	00	07
BA10	FF	FF	C0	C0	C0	C0	FF	FF	FC
BA18	3C	3C	3C	3C	3C	3C	3C	3C	E0
BA20	FF	FF	C3	C3	C3	C3	C3	C3	90
BA28	C3	C3	C3	C3	C3	C3	FF	FF	90
BA30	3C	3C	FF	FF	3C	3C	3C	3C	66
BA38	24	24	24	E7	E7	24	24	24	A6
BA40	C3	C3	C3	C3	C3	C3	C3	C3	18
BA48	FF	FF	FF	FF	FF	FF	18	18	2A
BA50	18	18	FF	FF	FF	FF	FF	FF	2A
BA58	C0	C0	C0	C0	C0	C0	C0	C0	00

BA60	FF	FF	03	03	03	03	FF	FF	08
BA68	FF	FF	F0	F0	F0	F0	0F	0F	DC
BA70	0F	0F	F0	F0	F0	F0	F0	FF	CD
BA78	FC	FC	FC	FC	C3	C3	C3	C3	FC
BA80	C3	C3	C3	C3	FC	FC	FC	FC	FC
BA88	FC	FC	FC	FC	FC	FC	FC	FC	E0
BA90	C3	C3	C3	C3	3F	3F	3F	3F	08
BA98	1F	1F	1F	1F	63	63	63	63	08
BAA0	00	00	00	00	00	00	FF	FF	FE
BAA8	FF	FF	00	00	00	00	00	00	FE
BAB0	03	03	03	03	03	03	03	03	18
BAB8	FF	81	BD	A5	A5	BD	81	FF	C4
BAC0	E7	66	66	66	FF	FF	FF	FF	15
BAC8	C3	C3	3C	3C	3C	C3	C3	00	C0
BAD0	18	18	18	18	18	18	18	18	C0
BAD8	00	00	00	00	00	00	F8	F8	F0
BAE0	00	00	00	00	00	00	1F	1F	3E
BAE8	0F	0F	0F	0F	00	00	00	00	3C
BAF0	00	00	00	00	F0	F0	F0	F0	C0
BAF8	F0	F0	00	00	00	00	00	00	E0
BB00	18	66	FF	18	24	42	81	81	FD
BB08	00	FF	00	FF	00	FF	00	FF	FC
BB10	18	18	18	FF	FF	18	18	18	8E
BB18	CC	CC	33	33	CC	CC	33	33	FC
BB20	AA	55	AA	55	AA	55	AA	55	FC
BB28	C3	66	3C	18	3C	66	C3	81	63
BB30	36	7F	7F	7F	3E	3E	1C	08	53
BB38	81	C3	DB	E7	DB	C3	81	00	25
BB40	08	1C	3E	7F	7F	36	08	3E	DC
BB48	08	1C	1C	2A	7F	2A	08	3E	59
BB50	FF	FF	FF	FF	FF	FF	FF	FF	F8
BB58	33	33	33	33	33	33	33	33	98
BB60	49	49	49	49	49	49	49	49	48
BB68	01	03	07	0F	1F	3F	7F	FF	F6
BB70	FF	FF	00	00	FF	FF	00	00	FC
BB78	C0	C0	C0	C0	C0	C0	C0	C0	00
BB80	00	00	FF	FF	00	00	FF	FF	FC
BB88	FF	FF	FF	FF	00	00	00	00	FC
BB90	00	00	00	00	FF	FF	FF	FF	FC
BB98	FF	FF	00	00	00	00	00	00	FE
BBA0	0F	0F	0F	0F	0F	0F	0F	0F	78
BBA8	F0	F0	F0	F0	F0	F0	F0	F0	80
BBB0	F0	F0	F0	F0	0F	0F	0F	0F	FC
BBB8	FF	00	FF	00	FF	00	FF	00	FC
BBC0	92	92	92	92	92	92	92	92	90
BBC8	00	00	00	FF	FF	00	00	00	FE
BBD0	00	00	00	00	00	00	FF	FF	FE
BBD8	03	03	03	03	03	03	03	03	18
BBE0	04	08	11	22	44	88	10	20	3B
BBE8	CC	CC	CC	CC	CC	CC	CC	CC	60
BBF0	20	10	88	44	22	11	08	04	3B
BBF8	01	03	06	0C	18	30	60	C0	7E
BC00	80	C0	E0	F0	F8	FC	FE	FF	01
BC08	18	18	18	18	18	18	18	18	C0
BC10	FF	FE	FC	F8	F0	E0	C0	80	01
BC18	C0	60	30	18	0C	06	03	01	7E

BC20	FF	7F	3F	1F	0F	07	03	01	F6
BC28	FF	FF	7E	3C	00	00	00	00	B8
BC30	3C	7E	FF	FF	FF	FF	7E	3C	70
BC38	99	99	99	99	99	5A	3C	18	AB
BC40	18	3C	5A	99	18	18	18	18	A7
BC48	FF	81	81	99	99	81	81	FF	34
BC50	C0	C0	00	18	18	00	03	03	B6
BC58	3C	7E	C3	C3	C3	C3	7E	3C	80
BC60	00	00	00	00	0F	08	08	08	27
BC68	00	00	00	00	F0	10	10	10	20
BC70	07	0F	1F	1F	1F	1F	0F	07	A8
BC78	08	08	08	0F	00	00	00	00	27
BC80	10	10	10	F0	00	00	00	00	20
BC88	E7	24	24	3C	66	C3	81	81	96
BC90	08	1C	3E	7F	7F	3E	1C	08	C2
BC98	24	42	A5	18	18	A5	42	24	46
BCA0	00	00	00	3C	7E	FF	FF	FF	B7
BCA8	E0	F0	F8	F8	F8	F8	F0	E0	80
BCB0	00	00	00	18	18	00	00	00	30
BCB8	00	66	66	00	00	66	66	00	98
BCC0	FF	C3	81	81	81	81	C3	FF	88
BCC8	C3	C3	00	18	18	00	C3	C3	3C
BCD0	66	FF	FF	66	66	FF	FF	66	94
BCD8	18	18	18	18	99	5A	3C	18	A7
BCE0	00	00	18	18	00	18	18	00	60
BCE8	10	20	40	FF	FF	40	20	10	DE
BCF0	66	66	00	66	66	00	66	66	64
BCF8	08	04	02	FF	FF	02	04	08	1A
BD00	DD	21	00	B5	21	1C	43	FE	31
BD08	80	38	16	FE	C0	38	0A	CB	99
BD10	5E	20	04	DD	21	00	F0	18	88
BD18	08	CB	66	20	04	DD	21	00	5B
BD20	F0	6F	26	00	29	29	29	EB	EB
BD28	DD	19	21	00	42	11	01	42	AD
BD30	01	08	00	36	00	ED	B0	06	E2
BD38	08	16	80	C5	21	07	42	06	D3
BD40	08	1E	01	DD	7E	00	A3	28	4D
BD48	03	7A	B6	77	CB	03	2B	10	B3
BD50	F2	C1	CB	0A	DD	23	10	E3	7B
BD58	C9	00	00	00	00	00	00	00	C9
BD60	21	00	44	3A	F9	42	47	C5	E6
BD68	3E	1B	CD	C0	BD	3E	4B	CD	F9
BD70	C0	BD	3E	50	CD	C0	BD	3E	93
BD78	00	CD	C0	BD	06	50	CD	C0	2D
BD80	BD	10	FB	06	28	7E	C5	D5	0E
BD88	E5	CD	00	BD	3E	1B	CD	C0	55
BD90	BD	3E	4B	CD	C0	BD	3E	08	D6
BD98	CD	C0	BD	3E	00	CD	C0	BD	D2
BDA0	06	08	21	00	42	7E	CD	C0	7C
BDA8	BD	23	10	F9	E1	D1	C1	23	7F
BDB0	10	D3	3E	0D	CD	C0	BD	C1	39
BDB8	05	C2	67	BD	C9	00	00	00	B4
BDC0	F5	3E	07	D3	F8	3E	7F	D3	95
BDC8	F9	3E	0F	D3	F8	DB	F9	E6	CB
BDD0	EF	FE	2F	20	EC	3E	07	D3	40
BDD8	F8	3E	7F	D3	F9	3E	0E	D3	A0
BDE0	F8	F1	D3	F9	F5	3E	07	D3	C2

BDE8	F8	3E	FF	D3	F9	3E	0F	D3	21
BDF0	F8	3E	00	D3	F9	3E	0F	D3	22
BDF8	F8	3E	01	D3	F9	F1	C9	00	BD
BE00	3E	1B	CD	C0	BD	3E	41	CD	EF
BE08	C0	BD	3E	08	CD	C0	BD	C3	D0
BE10	60	BD	3A	80	F8	FE	10	C0	9D
BE18	3A	04	F8	FE	01	C0	C5	D5	8F
BE20	E5	DD	E5	21	32	BE	E5	3A	D7
BE28	1C	43	CB	6F	CA	00	BE	C3	E4
BE30	00	BF	DD	E1	E1	D1	C1	C9	B9
BE38	E5	21	E3	03	E3	C3	12	BE	62
BE40	21	38	BE	22	16	40	21	20	D0
BE48	BF	CD	A7	28	21	00	B5	C3	F4
BE50	02	01	00	00	00	00	00	00	03
BE58	00	00	00	00	00	00	00	00	00
BE60	DD	21	00	48	3A	12	43	47	1C
BE68	C5	06	28	21	00	42	DD	7E	B1
BE70	00	E6	C0	0F	0F	0F	0F	0F	F1
BE78	0F	CD	CC	BE	DD	7E	00	E6	A7
BE80	30	0F	0F	0F	0F	CD	CC	BE	C3
BE88	DD	7E	00	E6	0C	0F	0F	CD	38
BE90	CC	BE	DD	7E	00	E6	03	CD	9B
BE98	CC	BE	3E	1B	CD	C0	BD	3E	6B
BEA0	4B	CD	C0	BD	3E	0C	CD	C0	6C
BEA8	BD	3E	00	CD	C0	BD	C5	06	10
BEB0	0C	21	00	42	7E	CD	C0	BD	37
BEB8	23	10	F9	C1	DD	23	10	AB	A8
BEC0	C1	3E	0D	CD	C0	BD	10	A0	06
BEC8	C9	00	00	00	B7	20	0A	36	E0
BED0	00	23	36	00	23	36	00	23	D5
BED8	C9	FE	01	20	0A	36	A0	23	EB
BEE0	36	A0	23	36	A0	23	C9	FE	B9
BEE8	02	20	0A	36	50	23	36	A0	AB
BEF0	23	36	50	23	C9	36	F0	23	DE
BEF8	36	F0	23	36	F0	23	C9	00	5B
BF00	3E	1B	CD	C0	BD	3E	41	CD	EF
BF08	C0	BD	3E	04	CD	C0	BD	C3	CC
BF10	60	BE	00	00	00	00	00	00	1E
BF18	00	00	00	00	00	00	00	00	00
BF20	3C	53	43	52	45	45	4E	20	1C
BF28	50	52	49	4E	54	45	52	3E	62
BF30	0D	28	63	29	20	31	39	38	83
BF38	33	20	76	6F	6E	20	54	43	5D
BF40	53	0D	0D	00	00	00	00	00	6D
BF48	00	00	00	00	00	00	00	00	00
BF50	00	00	00	00	00	00	00	00	00
BF58	00	00	00	00	00	00	00	00	00
BF60	00	00	00	00	00	00	00	00	00
BF68	00	00	00	00	00	00	00	00	00
BF70	00	00	00	00	00	00	00	00	00
BF78	00	00	00	00	00	00	00	00	00
BF80	00	00	00	00	00	00	00	00	00
BF88	00	00	00	00	00	00	00	00	00
BF90	00	00	00	00	00	00	00	00	00
BF98	00	00	00	00	00	00	00	00	00

BFA0	00	00	00	00	00	00	00	00	00
BFA8	00	00	00	00	00	00	00	00	00
BFB0	00	00	00	00	00	00	00	00	00
BFB8	00	00	00	00	00	00	00	00	00
BFC0	00	00	00	00	00	00	00	00	00
BFC8	00	00	00	00	00	00	00	00	00
BFD0	00	00	00	00	00	00	00	00	00
BFD8	00	00	00	00	00	00	00	00	00
BFE0	00	00	00	00	00	00	00	00	00
BFE8	00	00	00	00	00	00	00	00	00
BFF0	00	00	00	00	00	00	00	00	00
BFF8	00	00	00	00	00	00	00	00	00

# Anhang A: Basic Einfachtokens

128	END	129	FOR
130	RESET	131	SET
132	CLS	133	CMD
134	RANDOM	135	NEXT
136	DATA	137	INPUT
138	DIM	139	READ
140	LET	141	GOTO
142	RUN	143	IF
144	RESTORE	145	GOSUB
146	RETURN	147	REM
148	STOP	149	ELSE
150	TRON	151	TROFF
152	DEFSTR	153	DEFINT
154	DEFSNG	155	DEFDBL
156	LINE	157	EDIT
158	ERROR	159	RESUME
160	OUT	161	ON
162	OPEN	163	FIELD
164	GET	165	PUT
166	CLOSE	167	LOAD
168	MERGE	169	NAME
170	KILL	171	LSET
172	RSET	173	SAVE
174	SYSTEM	175	LPRINT
176	DEF	177	POKE
178	PRINT	179	CONT
180	LIST	181	LLIST
182	DELETE	183	AUTO
184	CLEAR	185	CLOAD
186	CSAVE	187	NEW
188	TAB(	189	TO
190	FN	191	USING
192	VARPTR	193	USR
194	ERL	195	ERR
196	STRING\$	197	INSTR
198	CHECK	199	TIME\$
200	MEM	201	INKEY\$
202	THEN	203	NOT
204	STEP	205	+
206	-	207	*
208	/	209	[
210	AND	211	OR
212	>	213	=
214	<	215	SGN
216	INT	217	ABS
218	FRE	219	INP
220	POS	221	SQR
222	RND	223	LOG
224	EXP	225	COS
226	SIN	227	TAN
228	ATN	229	PEEK
230	CVI	231	CVS
232	CVD	233	EOF
234	LOC	235	LOF
236	MKI\$	237	MKS\$
238	MKD\$	239	CINT
240	CSNG	241	CDBL
242	FIX	243	LEN
244	STR\$	245	VAL
246	ASC	247	CHR\$
248	LEFT\$	249	RIGHT\$
250	MID\$	251	'

# Colour-Basic Doppeltokens

255 128	↵	COLOUR	255 129	⌈	FCOLOUR
255 130	⌋	KEYPAD	255 131	⌈	JOY
255 132	⌈	PLOT	255 133	⌋	FGR
255 134	⌋	LGR	255 135	⌋	FCLS
255 136	—	PLAY	255 137	⌈	CIRCLE
255 138		SCALE	255 139	—	SHAPE
255 140	—	NSHAPE	255 141	□	XSHAPE
255 142	⌈	PAINT	255 143	·	CPOINT
255 144	⌈	NPLOT	255 145	—	SOUND
255 146	⌈	CHAR	255 147	⌈	RENUM
255 148	⌋	SWAP	255 149	⌋	FKEY
255 150	⌈	CALL	255 151	⌈	VERIFY
255 152	·	BGRD	255 153	⌈	NBGRD

(Die Tokens sind die der neuen Basic-ROMs,  
die seit April '83 eingebaut sind.)

Folgendes Programm erzeugt eine äquivalente Liste auf dem Bildschirm:

```

10 CLS:C=-1:A=127:B=3:CHAR4
20 PRINT"Basic Einfachtokens":X=&H1650
30 IFPEEK(X)>127THEN50
40 PRINTCHR$(PEEK(X)AND127);:X=X+1:GOTO30
50 B=B+1:C=C+1:IFC=124THEN90
60 IFB<48THEN80
70 GOSUB160:CLS:B=0
80 A=A+1:PRINT:COLOUR1:PRINT$(INT(B/2)*40)+(A-INT(A/2)*2)*20,A;" "":CO
LOUR5:PRINTCHR$(A);" "":COLOUR2:GOTO40
90 GOSUB160:CLS:B=3:C=-1:A=127:X=&H3930
100 PRINT"Colour-Basic Doppeltokens"
110 IFPEEK(X)>127THEN130
120 PRINTCHR$(PEEK(X)AND127);:X=X+1:GOTO110
130 IFA=129THENPRINT"R";
140 B=B+1:C=C+1:IFC=26THENEND
150 A=A+1:PRINT:COLOUR1:PRINT$(INT(B/2)*40)+(A-INT(A/2)*2)*20,"255";A;
" "":COLOUR5:PRINTCHR$(A);" "":COLOUR2:GOTO120
160 PRINT$960,"Bitte druecken Sie <RETURN>":INPUTR$:RETURN

```

Beachten Sie, daß ein Teil der Befehle erst im Disk-Basic benutzt werden (z.B. TIME \$).  
Im normalen Basic haben diese Befehle keine Funktion und führen zu "?SN ERROR".

# Anhang B:

## Dezimal/Hexadezimal/ASCII-Tabelle

Dez	Hex	Z.	Dez	Hex	Z.	Dez	Hex	Z.
0	= 00H		1	= 01H		2	= 02H	
3	= 03H		4	= 04H		5	= 05H	
6	= 06H		7	= 07H		8	= 08H	
9	= 09H		10	= 0AH		11	= 0BH	
12	= 0CH		13	= 0DH		14	= 0EH	
15	= 0FH		16	= 10H		17	= 11H	
18	= 12H		19	= 13H		20	= 14H	
21	= 15H		22	= 16H		23	= 17H	
24	= 18H		25	= 19H		26	= 1AH	
27	= 1BH		28	= 1CH		29	= 1DH	
30	= 1EH		31	= 1FH		32	= 20H	
33	= 21H	!	34	= 22H	"	35	= 23H	#
36	= 24H	\$	37	= 25H	%	38	= 26H	&
39	= 27H	'	40	= 28H	(	41	= 29H	)
42	= 2AH	*	43	= 2BH	+	44	= 2CH	,
45	= 2DH	-	46	= 2EH	.	47	= 2FH	/
48	= 30H	0	49	= 31H	1	50	= 32H	2
51	= 33H	3	52	= 34H	4	53	= 35H	5
54	= 36H	6	55	= 37H	7	56	= 38H	8
57	= 39H	9	58	= 3AH	:	59	= 3BH	;
60	= 3CH	<	61	= 3DH	=	62	= 3EH	>
63	= 3FH	?	64	= 40H	@	65	= 41H	A
66	= 42H	B	67	= 43H	C	68	= 44H	D
69	= 45H	E	70	= 46H	F	71	= 47H	G
72	= 48H	H	73	= 49H	I	74	= 4AH	J
75	= 4BH	K	76	= 4CH	L	77	= 4DH	M
78	= 4EH	N	79	= 4FH	O	80	= 50H	P
81	= 51H	Q	82	= 52H	R	83	= 53H	S
84	= 54H	T	85	= 55H	U	86	= 56H	V
87	= 57H	W	88	= 58H	X	89	= 59H	Y
90	= 5AH	Z	91	= 5BH	[	92	= 5CH	\
93	= 5DH	]	94	= 5EH	^	95	= 5FH	_
96	= 60H	`	97	= 61H	a	98	= 62H	b
99	= 63H	c	100	= 64H	d	101	= 65H	e
102	= 66H	f	103	= 67H	g	104	= 68H	h
105	= 69H	i	106	= 6AH	j	107	= 6BH	k
108	= 6CH	l	109	= 6DH	m	110	= 6EH	n
111	= 6FH	o	112	= 70H	p	113	= 71H	q
114	= 72H	r	115	= 73H	s	116	= 74H	t
117	= 75H	u	118	= 76H	v	119	= 77H	w
120	= 78H	x	121	= 79H	y	122	= 7AH	z
123	= 7BH	{	124	= 7CH		125	= 7DH	}
126	= 7EH	~	127	= 7FH	■	128	= 80H	␣
129	= 81H	␣	130	= 82H	␣	131	= 83H	␣
132	= 84H	␣	133	= 85H	␣	134	= 86H	␣



135 = 87H	┐	136 = 88H	┐	137 = 89H	
138 = 8AH	┐	139 = 8BH	┐	140 = 8CH	┐
141 = 8DH	□	142 = 8EH	┐	143 = 8FH	┐
144 = 90H	┐	145 = 91H	┐	146 = 92H	┐
147 = 93H	┐	148 = 94H	┐	149 = 95H	┐
150 = 96H	┐	151 = 97H	┐	152 = 98H	┐
153 = 99H	┐	154 = 9AH	┐	155 = 9BH	┐
156 = 9CH	┐	157 = 9DH	┐	158 = 9EH	┐
159 = 9FH	┐	160 = A0H	┐	161 = A1H	┐
162 = A2H	┐	163 = A3H	┐	164 = A4H	┐
165 = A5H	┐	166 = A6H	┐	167 = A7H	┐
168 = A8H	┐	169 = A9H	┐	170 = AAH	┐
171 = ABH	┐	172 = ACH	┐	173 = ADH	┐
174 = AEH	┐	175 = AFH	┐	176 = B0H	┐
177 = B1H	┐	178 = B2H	┐	179 = B3H	┐
180 = B4H	┐	181 = B5H	┐	182 = B6H	┐
183 = B7H	┐	184 = B8H	┐	185 = B9H	┐
186 = BAH	┐	187 = BBH	┐	188 = BCH	┐
189 = BDH	┐	190 = BEH	┐	191 = BFH	┐
192 = C0H	┐	193 = C1H	┐	194 = C2H	┐
195 = C3H	┐	196 = C4H	┐	197 = C5H	┐
198 = C6H	┐	199 = C7H	┐	200 = C8H	┐
201 = C9H	┐	202 = CAH	┐	203 = CBH	┐
204 = CCH	┐	205 = CDH	┐	206 = CEH	┐
207 = CFH	┐	208 = D0H	┐	209 = D1H	┐
210 = D2H	┐	211 = D3H	┐	212 = D4H	┐
213 = D5H	┐	214 = D6H	┐	215 = D7H	┐
216 = D8H	┐	217 = D9H	┐	218 = DAH	┐
219 = DBH	┐	220 = DCH	┐	221 = DDH	┐
222 = DEH	┐	223 = DFH	┐	224 = E0H	┐
225 = E1H	┐	226 = E2H	┐	227 = E3H	┐
228 = E4H	┐	229 = E5H	┐	230 = E6H	┐
231 = E7H	┐	232 = E8H	┐	233 = E9H	┐
234 = EAH	┐	235 = EBH	┐	236 = ECH	┐
237 = EDH	┐	238 = EEH	┐	239 = EFH	┐
240 = F0H	┐	241 = F1H	┐	242 = F2H	┐
243 = F3H	┐	244 = F4H	┐	245 = F5H	┐
246 = F6H	┐	247 = F7H	┐	248 = F8H	┐
249 = F9H	┐	250 = FAH	┐	251 = FBH	┐
252 = FCH	┐	253 = FDH	┐	254 = FEH	┐
255 = FFH	┐				

Folgendes Programm erzeugt ein aquivalentes Listing auf dem Bildschirm:

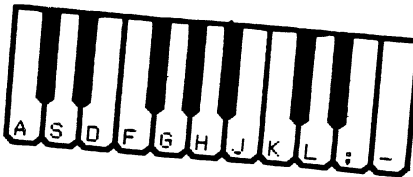
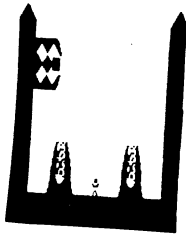
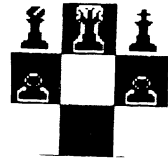
```

10 CLS:B=3
15 PRINT"Dez      Hex Z.  Dez      Hex Z.  Dez      Hex Z."
20 FORA=0TO255
30 IFA<32THENAS$="" "ELSEAS$=CHR$(A)
40 A1$=STR$(A):A1$=RIGHT$(A1$,LEN(A1$)-1)
50 IFLEN(A1$)<3THENA1$=STRING$(3-LEN(A1$),32)+A1$
60 A1$=A1$+" " = "
70 A1=INT(A/16):A2=AAND15
80 IFA1<10THENA1$=A1$+CHR$(48+A1)ELSEA1$=A1$+CHR$(55+A1)
90 IFA2<10THENA1$=A1$+CHR$(48+A2)ELSEA1$=A1$+CHR$(55+A2)
100 A1$=A1$+"H "+A$
110 PRINT$INT(B/3)*40+(B-INT(B/3)*3)*13,;
120 B=B+1
130 COLOUR3:PRINTMID$(A1$,1,3)::COLOUR1:PRINTMID$(A1$,4,3)::COLOUR6:PR
INTMID$(A1$,7,3)::COLOUR2:PRINTMID$(A1$,10,2);
140 IFB<69THENNEXTA
150 IFINKEY$=""THEN150ELSEB=0:CLS:GOTO140

```

Anhang C:

# Colour-Genie Software



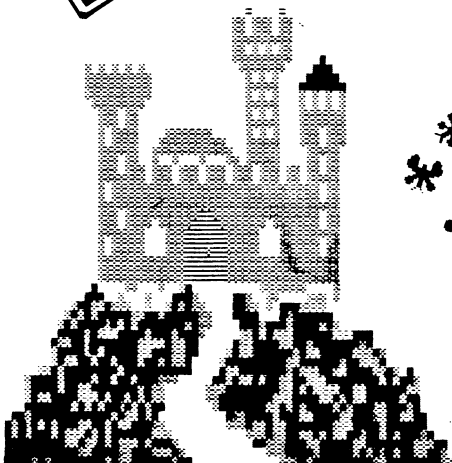
COLOUR-ASSEMBLER  
(C) 1983 BY TCS

DURCHGANG NR. 1 (SYMBOLTABELLE)  
DURCHGANG NR. 2 (ASSEMBLIERUNG)

LAENGE DES QUELLTEXTES : 1232  
LAENGE DER SYMBOLTABELLE : 84  
LAENGE DES OBJECTCODES : 115  
(RETURN) FUER PROGRAMMAUSFUEHRUNG

COLMON 1..2  
D0000 F3  
0001 AF  
0002 C37406  
0005 C30040  
0008 C30040  
000C E1  
000D C30000  
0010 C30340  
0013 C5  
0014 0601

DI A 067  
XDR 4000  
JP 4000H  
POP HL  
JP (HL) 0000H  
JP 4003H  
PUSH BC  
LD B, 01H



# TCSA

\*\*\*\*\*

## Verzeichnis der verfügbaren Colour-Genie-Software

\*\*\*\*\*

### 1.) HILFSPROGRAMME

COLOUR-COMPILER: Dieses Programm ist extrem wertvoll. Es bietet die Möglichkeit, Basicprogramme in Maschinenspracheprogramme zu übersetzen.

Der wesentliche Vorteil: Ihr Programm läuft nach Compilierung c.a. 40 mal (!) schneller.

Dies geschieht interaktiv, d.h. Basicprogramm, Maschinenprogramm und Compiler stehen gleichzeitig im Speicher, was ein sehr schnelles Arbeiten ermöglicht. Dabei können c.a. 9.5 kByte Basicprogramm verarbeitet werden. Verzichtet man auf die hochauflösende Grafik, steht noch mehr Speicher zur Verfügung. Voraussetzung: Ihr COLOUR-GENIE MUSS AUF 32K RAM ERWEITERT SEIN. Die meisten Colour-Basic-Befehle werden vom Colour-Compiler unterstützt. Die wesentlichste Einschränkung ist, daß der Colour-Compiler nur Ganzzahlen (Integers) verarbeitet.

Preis: 69.- DM

COLOUR-MONITOR I: Ein Maschinensprachemonitor mit starkem Befehlssatz, z.B. Disassemblieren, Ascii/Hex-Dump, Bänder laden/schreiben, Bytes suchen, Speicher editieren/verschieben/relozieren, Hex/Dezimal-Umwandlung u.a.m.

Preis: 39.- DM

COLOUR-ASSEMBLER: Mit diesem Programm können Maschinenspracheprogramme auf komfortable Weise entwickelt werden. Das Programmieren geschieht mit Z80-Mnemonics, Labels etc. Der Assembler erzeugt dann das Maschinenspracheprogramm, das zur Probe auch direkt ausgeführt werden kann.

Eine weitere Besonderheit des Assemblers ist, daß der Assembler-Text im ganz normalen Basic-Modus erzeugt wird, so daß alle Basic-Befehle wie EDIT, LLIST, CSAVE vom Assembler ausgenutzt werden.

Preis: 69.- DM

SOUND-EDITOR (auch im Handbuch "Colour Basic leicht gelernt" aufgelistet) : Ein nützliches Hilfsprogramm zur Programmierung des PSG-ICs (Sound-Chip). Sie editieren die PSG-Register auf dem Bildschirm und der entsprechende Ton wird gleichzeitig ausgegeben. Abschließend gibt das Programm alle PSG-Registerinhalte in dezimaler Schreibweise aus.

Preis: 25.- DM

ZEICHEN-EDITOR (auch im Handbuch "Colour Basic leicht gelernt" aufgelistet) : Dieses Programm ermöglicht es, 64 der 128 frei definierbaren Zeichen auf dem Bildschirm übersichtlich zu editieren. Es können auch Grafiken, wie z.B. eine Schrift, erstellt werden. Abschließend werden die definierten Zeichen in ein Basicprogramm geschrieben, was diese für den späteren Gebrauch oder zur Entwicklung eigener Programme wieder definiert. So wird die umständliche Handhabung der Programmierung der definierbaren Zeichen umgangen.

Preis: 25.- DM

GRAFIK-EDITOR: Wenn Sie Ihr Colour-Genie auf 32K RAM ausgerüstet haben, können Sie dieses Programm einsetzen, denn der "Grafik-Editor" hat eine Länge von 26000 Bytes! Er bietet die Möglichkeit, Grafiken im FGR-Modus mit einer Vielzahl von leistungsstarken Kommandos zu kreieren. Abschließend wird ein Basic-Programm erzeugt werden, das, eingebunden in ein eigenes Programm, das Bild blitzschnell wieder auf den Bildschirm bringt. Sie können also auch komplexe Grafiken sehr schnell fertigstellen und sie hinterher beliebig verwenden.

Preis: 69.- DM

SHAPER: Die Handhabung des Basicbefehls "SHAPE" ist recht umständlich. Hier bringt der "Shaper" Abhilfe: Mit wenigen Tastendrücken erzeugen Sie eine Figur, die abschließend im Foramt der "Shape-Table" abgespeichert wird.

Preis: 25.- DM

COLROT: Ein sehr nützliches Grafikprogramm, für alle die mit großen Texten Aufmerksamkeit erregen wollen (z.B. im Schaufenster). Colrot erzeugt Laufschriften mit bis zu 512 Zeichen Länge, Titel und Fußschrift, Inversdarstellung, Intermezzo u.v.a.m.

Ein komfortabler Editor ermöglicht einfaches Arbeiten.

Preis: 69.- DM

BASICODE: Dieses Programm macht ihr Colour-Genie Basicode-kompatibel ! Basicode ist das Standard-Basic, in dem z.B. das WDR-Fernsehen Programme austrahlt. Eine Hardwareänderung ist nicht nötig.

Preis: 25.- DM

PLOTTER: Plotter ist ein anspruchsvolles Mathematikprogramm, das beliebige Funktionen zeichnet, Wertetabellen erstellt u.s.w.

Preis: 39.- DM

ZEICHENEDITOR+: Dieser Editor, in Maschinensprache geschrieben, ermöglicht es alle 128 Zeichen sehr schnell und komfortabel zu editieren. Dabei gibt es sogar Befehle um Kreise zu zeichnen, Zeichen zu kopieren/duplizieren/rotieren/invertieren u.v.a.m.

Preis: 39.- DM

BASIC+5: Dieses Programm erweitert das normale Basic um 5 Befehle: Im Grafikmodus können beliebige Texte dargestellt werden, Rechteckflächen können gemalt werden, mit "SAVE" können Maschinenspracheprogramme gesichert und mit "LOAD" komfortabel geladen werden. Ferner steht ein Eingabebefehl zur Verfügung, der an beliebige Speicherzellen schreibt.

Preis: 39.- DM

ZEICHENSÄTZE: Dieses Programm läßt Sie den vorhandenen Zeichensatz für alle ASCII-Zeichen wahlweise gegen einen von acht neuen, interessanten Zeichensätzen ersetzen.

Für dieses Programm muß Ihr Colour-Genie mit 32K RAM ausgerüstet sein.

Preis: 25.- DM

SCREEN-PRINTER: (Auch in dem Buch "Das Colour-Genie-Buch 1" aufgelistet.) Auf dieses Programm haben die Besitzer des STAR-Druckers DP 510 bzw. DP 515 sicher gewartet !

Es ermöglicht es Ihnen, den Bildschirm jederzeit komplett auf Ihren Drucker auszugeben, egal ob Sie im FGR- oder im LGR-Modus sind. Definierte und feste Grafikzeichen werden ebenfalls mit ausgedruckt. Auch für dieses Programm brauchen Sie 32K RAM und natürlich einen STAR DP 510 / DP 515 Drucker.

Preis: 39.- DM

## 2.) SPIEL- u. GRAFIKPROGRAMME

INVASION AUS DEM WELTRAUM: Ein Maschinenspracheprogramm, das die Sound- u. Grafikmöglichkeiten des Colour-Genies voll ausnutzt. Ihre Aufgabe ist es, einen Pulk von Invasoren abzuwehren, der sich der Erde nähert.

Die Invasoren werden dabei immer schneller und gefährlicher.

Preis: 39.- DM

PUNKTEJAGD: Bei diesem Spiel geht es darum, alle Punkte in einem Labyrinth aufzusammeln, bevor Sie von einem Wächter eingeholt werden. Schnelle Grafik und Ton durch Maschinensprache.

Preis: 25.- DM

WURM: Unser derzeit schnellstes Action-Spiel. Ein Wurm kommt von oben den Bildschirm herab und versucht Sie zu vernichten. Dabei hat er die Spinne, die Fliege und die Ente als Helfer. Kein Spiel für ruhige Stunden!

Preis: 39.- DM

BREAK OUT: Bei diesem Video-Spiel muß eine Mauer mit Ihrem Ball zerstört werden. Dabei können verschiedenen Schwierigkeitsgrade vorgewählt werden. Hohe Geschwindigkeit durch Maschinenspracheprogrammierung.

Preis: 39.- DM

DEMOPROGRAMM: Hiermit können Sie besser als mit der mit Ihrem Colour-Genie z.Zt. mitgelieferten englischen Demokassette die Möglichkeiten Ihres Colour-Genies demonstrieren.

Preis: 25.- DM

ANDROMEDA: Ein erstklassiges dreidimensionales Weltraumspiel mit ansprechender Grafik. Feindliche Raumschiffe kommen auf Sie zu; vernichten Sie diese, bevor sie Ihnen wertvolle Energie abgesaugt haben. Ist die Zeit abgelaufen, so eilt Ihnen Ihre Mutterbasis zu Hilfe; aber das Auftanken will auch gelernt sein.

Preis: 39.- DM

MAU-MAU: Endlich haben Sie einen Spielpartner, der nicht wütend die Karten wegwirft, wenn er am verlieren ist. Dies dürfte wohl aber auch selten der Fall sein, vielmehr wahrscheinlich ist es, daß Sie den Computer vor Wut ausschalten, da ihr Computer Dank einer hervorragenden Taktik auf Sieg programmiert ist.

(Er schummelt nicht !!) Gespielt wird nach den üblichen Regeln.

Preis: 25.- DM

HEKTIK: Stellen Sie sich vor, Sie wären in einem Neubau mit 6 Geschossen, bei dem die Ebenen durch Leitern verbunden sind. Jetzt kommen Ihnen von oben Verfolger entgegen. Sie müssen sich ihnen stellen, denn es gibt keinen Fluchtweg. Graben Sie an strategisch wichtigen Stellen Löcher in den Boden, um so Ihre Verfolger auszuschalten.

Preis: 39.- DM

METEOR: Ein Super-Action-Spiel im Grafik-Modus Ihres Colour-Genies. Sie befinden sich in einem Meteoritenfeld und werden von Meteoriten, Sternen und schießenden Ufos bedrängt. Das Programm zeichnet sich durch sehr schnelle Grafik und gute Toneffekte aus.

Preis: 69.- DM

MOTTEN: Bei diesem Videospiel müssen Sie Kolonnen von Motten bekämpfen, die in gefährlichen Sturzflügen anfliegen und dabei auch noch schießen.

Preis: 39.- DM

PANIK: Ein Programm für 2 Spieler. Man muß versuchen, den Gegner einzumauern. Wählbare Geschwindigkeit macht "Mauer" entweder zu einem Reaktions- oder zu einem Strategiespiel. Das Programm kann wahlweise mit Joysticks oder mit der Tastatur bedient werden.

Preis: 25.- DM

TAUSENDFUß: Hier sollen Sie einen Wurm so steuern, daß er nur Futter aber kein Gift frißt. Dazu kommen noch Kraftfutter und Gegengift. Ein sehr unterhaltsames Geschicklichkeitsspiel für die ganze Familie. Bei Spielbeginn können viele verschiedene Geschwindigkeiten und Schwierigkeitsstufen vorgewählt werden.

Preis: 39.- DM

COLOUR-SCHACH: Jetzt können Sie gegen Ihr Colour-Genie auch Schach spielen. Colour-Schach bietet 7 verschiedene Spielstärken, Aufzeichnung eines laufenden Spieles auf Kassette, Ändern von Stellungen, einen Demonstrationsmodus und anderes mehr. Die Figuren werden auf ansprechende Weise grafisch dargestellt.

Preis: 69.- DM

EXNIMROID: Ein Denkspiel, abgeleitet vom bekannten Nim-Spiel. Es geht darum, aus mehreren Häufchen den letzten Spielstein zu nehmen. Sie spielen gegen den Computer.

Preis: 25.- DM

KINGS: Eine Regierungssimulation, bei der Sie für eine Dauer von 8 Jahren eine Insel regieren sollen. Dabei kann nur der kluge Einsatz aller Faktoren zu einem Erfolg führen. Haben Sie Ihre Amtszeit überlebt, werden Ihre Taten ausgewertet, und Sie können dann auch weiter regieren.

Preis: 25.- DM

MAUSI: Ein lustiges Videospiel, bei dem Sie als Maus im strömendem Regen Käse holen sollen, ohne naß zu werden.

Preis: 25.- DM

MAMPFMAN: Ähnlich wie unser Spiel "Punktejagd", nur daß Sie hier von mehreren Gespenstern verfolgt werden, die Sie unter bestimmten Bedingungen jedoch auch fressen können.

Preis: 39.- DM

COLOUR-KONG: Das absolute Videospiel ! Retten Sie Ihre Freundin vor dem wilden Affen. Mehrere Ebenen, super Grafik und Musik. Auf dem Band finden Sie Versionen für 32K und 16K RAM

Preis: 69.- DM

PANZERKAMPF: Ein Spiel für zwei Personen (JOYSTICKS und 32K RAM erforderlich). Liefern Sie sich mit Ihrem Gegner spannende Panzerschlachten ! Das Programm verfügt über 3 verschiedene Gelände mit Minen etc.

Preis: 69.- DM

EIS: Ein spannendes, zugleich jedoch auch recht schwieriges Actionspiel mit strategischen Elementen. Sie befinden sich in einem Kühlraum und müssen Eisblöcke so ans Rutschen bringen, daß die gefährlichen Schneemonster vernichtet werden.

Preis: 39.- DM

EAGLE: Vernichten Sie verschiedene Schwadronen von Vögeln, bevor Sie versuchen das große Mutterschiff zu zerstören.

Preis: 39.- DM

GAME of LIFE: Dieses bekannte Programm simuliert das Wachstum von Bakterienkulturen. Die Regeln werden im Programm erklärt und einige interessante Figuren sind fest abgespeichert.

Preis: 25.- DM

SAUG: Ein lustiges Videospiel, bei dem Sie aus einem unterirdischen Labyrinth Termiten ansaugen müssen, wobei Sie sich vor giftigen, roten Termiten hüten müssen.

Das Spiel kann auch mit Joysticks gesteuert werden.

Preis: 39.- DM

ExREVERSI: Spielen Sie Reversi gegen Ihr Colour-Genie ! Dabei kann zwischen mehreren Spielstufen gewählt werden. Das Spielbrett wird auf ansprechende Weise graphisch dargestellt.

Preis: 39.- DM

BÄNG-BÄNG: Zwei Cowboys begegnen sich in der Prärie und liefern sich ein erbittertes Duell. Für dieses Spiel sind Joysticks erforderlich.

Preis: 39.- DM

NETZO: Versuchen Sie mit Ihrem Pinsel alle Flächen auszumalen, ohne von Ihren Gegnern erwischt zu werden.

Preis: 39.- DM



DEATH-TRAP, ein dreidimensionales Grafik-Abenteuerspiel (Adventure):

Ein Programm zum Wahnsinnig werden! Sie befinden sich in einem Labyrinth mit über 1100 Räumen, in dem es von gefährlichen Einwohnern und Gegenständen wimmelt. Finden Sie die wichtigen Gegenstände und wenden Sie sie richtig an, um aus dem Labyrinth zu entkommen. Zwei Voraussetzungen müssen allerdings erfüllt sein: Ihr Colour-Genie muß 32K RAM haben, und Sie müssen elementare Englischkenntnisse besitzen, da die Sie Kommandos in Form von englischen Sätzen eingeben.

Preis: 69.- DM

TCS-CHOPPER: Die Geschmäcker sind verschieden - aber dies ist wohl das beste Videospiel, das es z.Zt. für das Colour-Genie gibt! Dreizehn Ihrer Kameraden sind in einem von vier riesigen Labyrinth ausgesetzt worden. Versuchen Sie nun, diese mit Ihrem Hubschrauber zu retten! Aber Vorsicht, Ihre Freunde werden scharf bewacht!

32K RAM sind auch hier notwendig.

Preis: 69.- DM

DOPPEL-WURM: Vielleicht kennen Sie unser beliebtes Spiel "Tausendfuß". "Doppel-Wurm" ist sehr ähnlich, allerdings spielen hier zwei Spieler gegeneinander, was die Spannung natürlich erhöht. Achtung: Nur für Joystick-Besitzer!

Preis: 39.- DM

MADTREE: Ein schwieriges Videospiel: Sie sind eine fleißige Biene und müssen eine Reihe von Blumen ständig bestäuben, damit diese nicht eingehen. Doch diese Blumen sind undankbar und gefährden Sie.

Preis: 39.- DM

DIG-BOY: In Ihrem unterirdischen Reich kämpfen Sie gegen Drachen und andere Untiere. Locken Sie diese unter einen der wackligen Felsen, um sie zu zerschmettern.

Dieses Spiel ist sehr empfehlenswert, da es eine Menge Abwechslung bietet.

Preis: 39.- DM

EMPIRE: Für alle, die nicht nur reine Aktion wollen, ist "Empire" das ideale Spiel. Dieses Spiel kann mit bis zu sechs Spielern gleichzeitig gespielt werden, der Computer spielt ebenfalls mit. Jeder Spieler hat ein kleines Reich, das nun durch Handel, Politik, Krieg u.s.w. zu Wohlstand kommen soll. Dabei stehen Sie mit den anderen Nationen ständig in Beziehung, sei es durch Handel oder durch Krieg.

Ansprechende Grafik macht das Programm noch interessanter.

Ihr Colour-Genie muß für "Empire" auf 32K RAM aufgerüstet sein.

Preis: 69.- DM

### 3.) MUSIKPROGRAMME

MUSIK: Dieses Programm spielt festprogrammierte Musikstücke mit verblüffender Qualität.

Preis: 25.- DM

ORGEL: Spielen Sie Orgel auf Ihrem Colour-Genie !

Ihnen stehen zwei Manuale, einstellbares Delay und sogar eine Schlagzeugbegleitung zur Seite.

Preis: 25.- DM

COLOUR-SYNTHESIZER: Ein Programm, das man gesehen haben muß, um es überhaupt für möglich zu halten. Der Colour-Synthesizer macht aus Ihrem Colour-Genie einen vollwertigen 3-Kanal Synthesizer mit VCO, VCA, Hüllkurve, Schlagzeug...

Acht Einstellungen aller Regler können vorprogrammiert, auf Tastendruck abgerufen und auf Band gesichert werden.

Verblüffend ist auch die graphische Darstellung.

Preis: 69.- DM

COMPOSER: Mit diesem Programm können Sie sehr einfach eigene Musikstücke komponieren. Noten, Pausenzeichen u.s.w. werden grafisch dargestellt, können editiert und auf Band gespeichert werden.

Preis: 39.- DM

### 4.) LERNPROGRAMME (z.Zt. nur eines, demnächst mehr):

MATHEMATIK-LERNPROGRAMM: Interessant für Kinder bis in's Grundschulalter. Wahlweise können die vier Grundrechenarten geübt werden, wobei man zwischen verschiedenen Schwierigkeitsgraden wählen kann. Lustige grafische Gestaltung sorgt dafür, daß das Üben Spass macht.

Preis: 25.- DM

\*\*\* STAND DER TCS-SOFTWARELISTE, COLOUR-GENIE, VOM 12.8.1983 \*\*\*

In Vorbereitung: Colour-Text, Lernprogramme für Kinder, verbesserter Maschinensprache-Monitor und viele interessante Spiele !

# BESTELLFORMULAR

Zur Bestellung von Colour-Genie-Programmen können Sie dieses  
Formular benutzen. Schicken Sie dieses an Ihren GENIE-Händler  
oder direkt an TCS Computer GmbH  
Kölustraße 4  
5205 St. Augustin 2  
Tel.: 02241 / 28071

Bankverbindung:  
Raiffeisenbank Menden BLZ 37069507 Konto-Nr. 1525

Der Versand erfolgt per Nachnahme oder Vorkasse auf obiges  
Konto (dies natürlich nicht, wenn Sie bei Ihrem Händler  
bestellen).

-----  
Ich bitte um Zusendung folgender TCS Colour-Genie-Programme:

Anzahl	:	Name des Programms	:	Preis/DM	:	Gesamt
=====	:		:		:	
-----	:		:		:	
-----	:		:		:	
-----	:		:		:	
-----	:		:		:	
-----	:		:		:	
-----	:		:		:	
-----	:		:		:	
-----	:		:		:	
-----	:		:		:	
=====	:		:		:	
					Gesamt:	DM

Meine Adresse ist:

Name: .....

Straße: .....

Ort: .....

Datum: ..... Unterschrift: .....

Raum für NOTIZEN:

Raum für NOTIZEN:

