

```

1119 : 45 4d 4f 52 59 00 0d 11 0f
1121 : 1d 57 41 52 4e 49 4e 47 7b
1129 : 3a 20 43 48 41 4e 47 45 7b
1131 : 53 20 4e 4f 54 20 53 41 28
1139 : 56 45 44 21 20 51 55 49 dc
1141 : 54 20 28 59 2f 4e 29 00 e5
1149 : 93 11 11 11 11 11 1d 1d 14
1151 : 1d 1d 1d 51 55 49 54 20 a0
1159 : 54 4f 20 12 20 42 20 92 59
1161 : 41 53 49 43 20 4f 52 20 0d
1169 : 54 45 53 54 20 12 20 4d 6d
1171 : 20 92 4f 44 55 4c 20 3f ed
1179 : 00 93 0d 11 11 11 11 11 a8
1181 : 11 11 11 11 1d 1d 45 4e ee
1189 : 54 45 52 20 46 49 4c 45 83
1191 : 4e 41 4d 45 20 3a 0d 1d be
1199 : 1d 1d 1d 1d 1d 1d 1d 1d 99
11a1 : 1d 1d 1d 1d 1d 1d 1d 1d a1
11a9 : 1d 1d 2d 2d 2d 2d 2d 2d 91
11b1 : 2d 2d 2d 2d 2d 2d 2d 2d b1
11b9 : 2d 2d 0d 91 91 1d 1d 1d a3
11c1 : 1d 1d 1d 1d 1d 1d 1d 1d c1
11c9 : 1d 1d 1d 1d 1d 1d 1d 1d c9
11d1 : 00 32 80 5e fe c3 c2 cd 8b
11d9 : 38 30 93 20 2a 2a 20 2a db
11e1 : 2a 2a 2a 2a 2a 2a 2a 2a e1
11e9 : 2a 2a 2a 2a 2a 2a 2a 2a e9
11f1 : 2a 20 4d 4f 44 55 4c 20 c9
11f9 : 56 45 52 53 49 4f 4e 20 79
1201 : 2a 2a 20 78 a2 ff 9a d8 9e
1209 : 8e 16 d0 20 a3 fd 20 50 26
1211 : fd 20 15 fd 20 5b ff 58 b1
1219 : 20 53 e4 20 bf e3 a0 00 be
1221 : a9 80 84 fb 85 fc b1 fb aa
1229 : 91 fb c8 d0 f9 e6 fc a5 1a
1231 : fc c9 c0 d0 f1 ea ea ea 54
1239 : a9 00 a2 80 85 37 86 38 38
1241 : 20 44 a6 a2 fb 9a a9 00 bc
1249 : 85 fb a9 30 85 fc a2 00 07
1251 : bd 09 80 20 d2 ff e8 e0 49
1259 : 28 d0 f5 e6 fc a2 00 bd a4
1261 : 57 81 e0 17 d0 02 a5 fc 41
1269 : c9 00 f0 06 20 d2 ff e8 9a
1271 : d0 ed a5 fb 0a 0a 0a 0a 4e
1279 : aa a0 00 bd 81 81 c9 00 76
1281 : f0 09 20 d2 ff e8 c8 c0 44
1289 : 10 30 f0 a2 00 bd 74 81 04
1291 : c9 00 f0 06 20 d2 ff e8 c2
1299 : d0 f3 e6 fb ad 56 81 c5 bb
12a1 : fb d0 b8 e6 fc 20 e4 ff 74
12a9 : f0 fb c5 fc 10 f7 c9 31 f2
12b1 : 30 f3 e9 31 0a 0a 0a 48 25
12b9 : aa bd d1 81 85 5f e8 bd 59
12c1 : d1 81 85 60 e8 bd d1 81 87
12c9 : 85 5a e8 bd d1 81 85 5b 63
12d1 : e8 bd d1 81 85 58 e8 bd 77
12d9 : d1 81 85 59 e8 bd d1 81 be
12e1 : 85 61 e8 bd d1 81 85 62 0d
12e9 : 20 bf a3 20 44 e5 a5 61 a2
12f1 : c9 00 f0 08 c9 01 f0 16 8c
12f9 : 68 6c 61 00 a5 62 c9 00 84
1301 : d0 f6 4c 75 80 20 8e a6 9f
1309 : 20 60 a6 4c ae a7 a5 62 10
1311 : c9 08 d0 e4 68 aa e8 e8 00
1319 : e8 e8 bd d1 81 85 2d e8 ea
1321 : bd d1 81 85 2e d0 de 00 bd
1329 : 0d 1d 1d 1d 1d 1d 1d 1d 19
1331 : 1d b0 c3 ae 0d 1d 1d 1d d5
1339 : 1d 1d 1d 1d 1d dd 12 1d 13
1341 : 92 dd 20 20 00 0d 1d 1d e5
1349 : 1d 1d 1d 1d 1d 1d ad c3 d9
1351 : bd 00 00 00 00 00 00 00 0f
1359 : 00 00 00 00 00 00 00 00 5a
1361 : 00 00 00 00 00 00 00 00 62
1369 : 00 00 00 00 00 00 00 00 6a
1371 : 00 00 00 00 00 00 00 00 72
1379 : 00 00 00 00 00 00 00 00 7a
1381 : 00 00 00 00 00 00 00 00 82
1389 : 00 00 00 00 00 00 00 00 8a
1391 : 00 00 00 00 00 00 00 00 92
1399 : 00 00 00 00 00 00 00 00 9a
13a1 : 00 00 00 00 00 00 00 00 a2
13a9 : 00 00 00 00 00 00 00 00 aa
13b1 : 00 00 00 00 00 00 00 00 b2
13b9 : 00 00 00 00 00 00 00 00 ba
13c1 : 00 00 00 00 00 00 00 00 c2
13c9 : 00 00 a2 00 bd 0e 82 f0 aa
13d1 : 06 20 d2 ff e8 d0 f5 ad e4
13d9 : ff bf c9 e0 d0 f9 60 4d 3f
13e1 : 4f 44 55 4c 20 41 42 53 ed
13e9 : 43 48 41 4c 54 45 4e 00 d3
13f1 : a9 36 85 01 38 a5 5a e5 1d
13f9 : 5f 85 22 a8 a5 5b e5 60 46
1401 : aa e8 98 f0 23 a5 5a 38 9d
1409 : e5 22 85 5a b0 03 c6 5b a1
1411 : 38 a5 58 e5 22 85 58 b0 00
1419 : 08 c6 59 90 04 b1 5a 91 47
1421 : 58 88 d0 f9 b1 5a 91 58 15
1429 : c6 5b c6 59 ca d0 f2 a9 cc
1431 : 37 85 01 60 00 29 21 ca db

```

Die besten Tips und Tricks

Es ist einfach lästig, immer wieder -zig Hefte nach einem bestimmten Trick durchsuchen zu müssen. Deshalb haben wir die wichtigsten und interessantesten Tips und Tricks aus früheren 64'er-Ausgaben für Sie zusammengestellt.

Kennen Sie folgende Situation? Man sitzt vor einem Programmierproblem und erinnert sich - da stand doch mal was bei den Tips und Tricks. Ein Stapel 64'er wird geholt, und die Sucherei geht los. Wir wollen da Abhilfe schaffen. Auf den folgenden Seiten finden Sie unter anderem kurze Programme zu den Themen: Basic-Programme retten und verbinden, Bildschirm speichern, Cursor steuern, Listschutz, Maschinenprogramme speichern, PET-Simulation, Reset-Hilfen.

Basic-Programme verbinden

So manch einen C 64-Besitzer wird es schon geärgert haben, daß sein Computer keinen MERGE-Befehl besitzt. Mit wenig Aufwand ist es aber dennoch möglich, Basic-Programme aneinanderzuhängen:

1. Im Direktmodus »PRINT PEEK(43); PEEK(44)« eingeben und sich die Ergebnisse merken.

2. Das erste Programm normal laden.

3. Erscheint jetzt nach »PRINT PEEK(45)« eine Null oder eine Eins, dann geben Sie »POKE 43, 256 + PEEK(45) - 2 : POKE 44, PEEK(46) - 1 : NEW« ein. Im anderen Fall wird »POKE 43, PEEK(45) - 2 : POKE 44, PEEK(46) : NEW« eingegeben.

4. Nun wird das anzuhängende Programm geladen (Achtung! Das anzuhängende Programm muß die höheren Zeilennummern haben).

5. Jetzt POKEN Sie in die Speicherstellen 43 und 44 die zu Anfang gemerkten Werte.

Beide Programme sind nun verbunden. Wichtig bei der ganzen Prozedur ist, daß keine Variablen definiert werden, da das MERGEN sonst nicht richtig funktioniert.

(Thomas Lopatic)

Basic-Programme retten

Die Betriebssystemroutine »Angleich von Koppeladressen« ab Adresse 42291 ermöglicht ein schnelles und einfaches »UNNEW« nach einem versehentlichen »NEW« oder Reset: POKE 2049,1 : POKE 2050,1 : SYS 42291

Danach kann zumindest wieder gelISTet werden. Ein vollständiges »UNNEW« verlangt allerdings die Korrektur der Zeiger auf den Beginn der Variablen und Felder. Dazu wäre die Kenntnis der Programmlänge notwendig. Man kann sich aber behelfen, indem man das Programm notfalls in Teilen auf dem Bildschirm auf LISTet und die einzelnen Zeilen mit der RETURN-Taste neu übernimmt.

(Gerhard Wagner)

UNNEW

Haben Sie ein Programm versehentlich mit NEW gelöscht, hilft UNNEW:

```
100 FOR I=525 TO 578
120 READ A : POKE I,A : NEXT I
200 POKE 43,525 AND 255 : POKE 44,2
210 POKE 45,578 AND 255 : POKE 46,2
220 CLR : SAVE "UNNEW",8 : REM bzw. ,1,1
300 DATA 160,003,200,177,043,208,251,200
310 DATA 200,152,160,000,145,043,165,044
320 DATA 200,145,043,133,060,160,000,132
330 DATA 059,162,000,200,208,002,230,060
340 DATA 177,059,208,245,232,224,003,208
350 DATA 242,200,208,002,230,060,132,045
360 DATA 164,060,132,046,096,256
```

Wenn Sie dieses Programm eingeben und starten, wird ein Programm namens »UNNEW« auf Diskette geschrieben. Falls Sie aus Versehen NEW eingetippt haben, dann laden Sie das Programm durch

```
LOAD "UNNEW" ,8,1
und starten es durch
SYS 525
```

Bildschirm auf Kassette/Diskette

Das kurze Maschinenprogramm (Listing 1) dient dazu, den Bildschirminhalt beim Commodore 64 zu speichern und auch wieder einzuladen. Dies kann mit einem Recorder oder Diskettenlaufwerk geschehen. Dabei werden außer dem eigentlichen Bildschirmspeicher (der an eine beliebige Stelle verschoben sein kann und nicht ab Adresse 1024 liegen muß) auch das Color-RAM und die Tabelle der Doppelzeilenkenn-

zeichnungen mitberücksichtigt. Das Speichern und Laden erfolgt über einen Pufferbereich im »versteckten« RAM ab 40960, da die drei verschiedenen Speicherbereiche in einem Stück gespeichert und außerdem die 1000 Farbnibbles zu 500 Bytes zusammengeschoben werden. Das erspart beim Arbeiten mit dem Recorder wertvolle Zeit.

Nach dem Eintippen des Basic-Loaders kann durch »RUN 16« geprüft werden, ob es richtig eingegeben wurde. Trotzdem sollte es vor dem Start gespeichert werden, da auch durch Prüfsummen nicht alle Fehler erkannt werden können. Ausgegeben werden zwei Adressen für das Laden und Speichern. Das Programm kann an jede Stelle des Arbeitsspeichers geladen werden, in der vorliegenden Version lädt es sich an das Ende des Basic-Arbeitsspeichers. Es ist davon auszugehen, daß das Programm direkt nach dem Einschalten ohne andere Erweiterungen geladen und gestartet wird.

Soll ein Bildschirminhalt auf Diskette gespeichert werden, so erfolgt dies durch

```
SYS 40738 "FILENAME", 8
```

Beim Arbeiten mit Recorder kann (auch beim Laden) die Gerätenummer entfallen; die Angabe einer Sekundäradresse ist nicht erlaubt. Durch das Speichern bedingte Betriebssystemmeldungen wie »PRESS RECORD & PLAY« sowie das Scrolling des Bildschirms sind unbedeutend und werden nicht berücksichtigt. Alle Bildschirmdaten werden zuvor in einen Zwischenspeicher übertragen.

Das Laden solcherart gespeicherter Bildschirminhalte geschieht mit

```
SYS 40833 "FILENAME", 8
```

Zunächst werden die Daten in den Puffer von Adresse 40960 bis 42483 geladen und dann in MSB-Tabelle, Farbspeicher und die momentan gewählte Videomatrix übertragen.

(Ralph Babel)

Cursorsteuerung leichtgemacht

Bei professionellen Programmen der PC-Klasse kann der Cursor meist über Eingabegeräte wie die Maus positioniert werden. Daß es auch recht gut mit dem Joystick und dem C 64 funktioniert, beweist dieses Programm.

Haben Sie den kurzen MSE-Lader (Listing 2) eingetippt und gestartet, können Sie den Cursor mit einem Joystick in Port 2 steuern. Das Steuerprogramm befindet sich von Adresse \$C000 bis \$C066 im Speicher. Die ersten 15 Bytes nimmt eine Initialisierungsroutine in Anspruch, die den Inter-

```
0 REM SCREENSAVER 64 <057>
1 D=55:A=PEEK(56)*256+PEEK(D)-222:GOSUB 4: <216>
  CLR:S=PEEK(56)*256+PEEK(55)
2 FOR I=S TO S+221:READ A:POKE I,A:NEXT A= <101>
  S+173:D=S+1:GOSUB 4:D=S+96:GOSUB 4
3 PRINT{DOWN}SAVE:"S:PRINT"LOAD:"S+95:END <255>
4 POKE D,A-INT(A/256)*256:POKE D+1,A/256:R <071>
  ETURN
5 DATA 32,,162,4,177,7,145,9,200,208,249, <234>
  230,8,230,10,202,208,242,162,24,181
6 DATA 217,157,231,163,202,208,248,162,218 <117>
  ,202,202,134,8,177,7,41,15,133,11,232
7 DATA 232,134,8,177,7,10,10,10,10,5,11,14 <230>
  5,9,200,208,229,230,10,224,219,208,224
8 DATA 169,160,133,8,165,1,72,41,254,133,1 <038>
  ,169,7,162,244,160,165,32,216,255,170
9 DATA 104,133,1,138,176,1,96,76,249,224,3 <130>
  2,,169,,32,213,255,176,243,165,1,72
10 DATA 41,254,133,1,162,4,160,,177,9,145, <154>
  7,200,208,249,230,8,230,10,202,208,242
11 DATA 162,24,189,231,163,149,217,202,208 <100>
  ,248,162,218,202,202,134,8,177,9,145,7
12 DATA 232,232,134,8,74,74,74,74,145,7,20 <131>
  0,208,235,230,10,224,219,208,230,104
13 DATA 133,1,96,32,87,226,162,1,32,121,,2 <036>
  40,3,32,241,183,160,2,32,186,255,173,
14 DATA 221,73,3,133,8,173,24,208,41,240,7 <047>
  0,8,106,70,8,106,133,8,160,,132,7,132
15 DATA 9,169,160,133,10,96 <051>
16 CLR:FOR Q=. TO 221:READ A:X=A+X:Y=A-Y:NE <057>
  XT
17 PRINT{DOWN}CHECKSUM "MID$("ERROR OK",1 <027>
  -5*(X=27349 AND Y=-217),5):END
```

© 64'er

Listing 1. Listing zu »Bildschirm auf Kassette/Diskette«

PROGRAMM : CURSOR

C000 C066

```
C000 : A9 0F 8D 14 03 A9 C0 8D B2
C008 : 15 03 A9 06 85 02 60 C6 41
C010 : 02 F0 03 4C 31 EA A9 06 F2
C018 : 85 02 A6 C6 E0 08 90 03 B7
C020 : 4C 31 EA AD 00 DC 29 01 03
C028 : C9 00 D0 05 A9 91 20 5E 2A
C030 : C0 AD 00 DC 29 02 C9 00 2C
C038 : D0 05 A9 11 20 5E C0 AD 6B
C040 : 00 DC 29 04 C9 00 D0 05 63
C048 : A9 9D 20 5E C0 AD 00 DC C7
C050 : 29 08 C9 00 D0 05 A9 1D 06
C058 : 20 5E C0 4C 31 EA A6 C6 F4
C060 : 9D 77 02 E6 C6 60 00 A0 C7
```

Listing 2. Listing zu »Cursorsteuerung leichtgemacht«. Das Programm muß mit dem MSE eingegeben werden.

```

10 REM*****
20 REM*      CURSOR STEUERUNG      *
30 REM*      *                      *
65 REM*      PETER SIEPEN          *
70 REM*      *                      *
75 REM*      VON-STEPHAN-STR.6     *
80 REM*      *                      *
82 REM*      4200 OBERHAUSEN 1     *
84 REM*      *                      *
85 REM*      TELEFON : (0208) /26555 *
90 REM*****
95 :
100 SYS9*4096
110 .OPT P,00
111 :
112 :
113 :
114 :
120 *= $C000
125 :
130 :
140 :
150 TEST = $02      ;ZAEHLVARIABLE
160 JOY   = 56320    ;PORT #2
170 AZITP = $C6      ;ANZAHL ZEICHEN IM PUFFER
174 :
175 :
180 LDA #<BEGINN; INTERUPTVEKTOR
185 STA $314
190 LDA #>BEGINN; AUF NEUE
195 STA $315 ;ADDRESSE SETZEN
200 LDA #$06 ;ZAEHLVARIABLE
205 STA TEST ;HOCHSETZEN
210 RTS      ;ZURUECK ZU BASIC
215 :
220 :
225 :
300 BEGINN    DEC TEST
320 :         BEQ START ;GENUG LEER IRR
330 :         JMP $EA31  ;NEIN WEITER MIT IRR
340 START     LDA #$06 ;ZAEHLVARIABLE HOCHSETZEN
350 :         STA TEST
360 :         LDX AZITP ;TASTATURPUFFER
370 :         CPX #$08 ;VOLL
380 :         BCC WEITER
390 :         JMP $EA31 ;JA WEITER MIT INTERUPT
400 WEITER    LDA JOY   ;WENN JOY NICHT
410 :         AND #1    ;NACH OBEN
420 :         CMP #0    ;WEITER
430 :         BNE NOBEN
440 :         LDA #"O"  ;WENN JA STEUERZEICHEN
450 :         JSR AUSG  ;AUSGEBEN
460 NOBEN     LDA JOY
470 :         AND #2
480 :         CMP #0
490 :         BNE NUNTEN
500 :         LDA #" "
510 :         JSR AUSG
520 NUNTEN    LDA JOY
530 :         AND #4
540 :         CMP #0
550 :         BNE NLINKS
560 :         LDA #"I"
570 :         JSR AUSG
580 NLINKS    LDA JOY
590 :         AND #8
600 :         CMP #0
700 :         BNE NRECHTS
710 :         LDA #"J"
720 :         JSR AUSG
730 NRECHTS   JMP $EA31
740 :
750 :
760 :
770 :
780 :
790 :
800 AUSG      LDX $C6 ;X-REG FUER
810 :         STA $277,X ;INDIZIERTE
820 :         INC AZITP ;ADDRESSIERUNG
830 :         RTS      ;LADEN
840 :         ;IN TASTPUFFER
850 :         ;SCHREIBEN
860 :         ;ANZAHL ZEICHEN
870 :         ;IM TASTPUFFER
880 :         ;ERHOEHEN
890 :
900 :
READY.

```

Listing 3. Assemblerprogramm der
Cursorsteuerung

ruptvektor auf \$C00F legt und das Steuerprogramm in den Kernelinterrupt einbindet. Die Routine wird mit SYS 49152 aufgerufen. Das Programm benutzt Speicherzelle \$02 als Zählregister, da der Joystick nur bei jedem sechsten Interrupt abgefragt wird. In Zeile 300 des Assemblerlistings (Listing 3) wird die Zählvariable um 1 erniedrigt. Ist die Variable 0, wird sie auf 6 gesetzt und in die Steuerungsroutine verzweigt. Ab Zeile 360 wird der Tastaturpuffer auf freien Platz überprüft. Sollte der Puffer voll sein, wird sofort in die Interruptroutine des Betriebssystems (\$EA31) gesprungen. In Zeile 400 wird der Joystick abgefragt und das entsprechende Cursorsteuerzeichen in den Akku geladen. Ab 800 wird das Zeichen in den Tastaturpuffer geschrieben und der Pufferzeiger erhöht. (P. Siepen)

Cursor steuern

Das Betriebssystem des C 64 enthält eine Routine, die man benutzen kann, um den Cursor an eine beliebige Stelle zu setzen. Geben Sie doch mal folgendes ein:
POKE 214, (Zeile) : POKE 211, (Spalte) : SYS 58640 :
PRINT " TEXT" (Michael Keukert)

Cursor beschleunigt

Für alle diejenigen C64-Besitzer, denen die Bewegung des Cursors bisher zu langsam war, gibt es einen speziellen POKE.

Mit POKE 56325,5 wird der Cursor rasend schnell und flitzt bei Betätigung der Cursortasten nur noch so über den Bildschirm. Wer's lieber gemütlicher mag, der sollte es statt dessen einmal mit POKE 56325,255 probieren.

(Oliver Bausch)

Directory ohne Programmverlust

Häufig möchte man sich das Directory einer Diskette ansehen, ohne das gerade im Speicher befindliche Programm zu zerstören. Wenn man das DOS 5.1 nicht geladen hat, behilft man sich meist mit der zeitaufwendigen Zwischenspeicherung des Programms auf der Diskette. Es geht jedoch auch einfacher und schneller. Geben Sie einfach den folgenden Befehl ein:

POKE 44, PEEK(46) + 1

Damit wird der Basic-Anfang auf einen freien Speicherbereich gestellt. Sie können jetzt wie gewohnt mit »LOAD "\$",8« das Directory laden und anschließend auflisten.

Mit POKE 44,8 sind Sie dann wieder im eigentlichen Programm.

(Heinzpeter Oelkers)

Floppy-Lister

Mit einem einfachen SYS-Aufruf können Sie Programme und sequentielle Dateien direkt von Diskette listen. Programme im Speicher bleiben dabei erhalten.

»Floppy-Lister« bietet zwei Möglichkeiten, ein Programm von Diskette zu listen. Basic-Programme und sequentielle Dateien können entweder als Klartext oder als Speicherauszug (Dumps) gelistet werden. Im Dumpmodus werden sämtliche Daten als Hexcodes ausgegeben und, soweit möglich, in ASCII-Zeichen übersetzt. Deshalb eignet er sich besonders

zum Analysieren von unbekannten Programmdateien. Maschinenprogramme dürfen grundsätzlich nur auf diese Art gelistet werden, will man einen Absturz vermeiden.

Ist das Ladeprogramm (Listing 4) mit dem MSE eingetippt, kann das Maschinenprogramm auf Diskette oder Kassette gespeichert werden. Der »Floppy-Lister« kann dann direkt geladen (LOAD "name", 8, 1, oder LOAD "name", 1, 1) und mit SYS 49152, "XY:filename" aufgerufen werden. Vergessen Sie nach dem absoluten Laden nicht, den erforderlichen NEU-Befehl im Direktmodus einzugeben, um die Basic-Zeiger in einen vernünftigen Zustand zu bringen. »X« steht für die Fileart: entweder »P« für Programm oder »S« für sequentielle Datei. »Y« steht für den Modus, in dem gelistet werden soll: »L« für einfaches Listen und »D« für Ausgabe in Form eines Speicherauszugs. Zum Beispiel listet der Befehl SYS 49152, "SL:TESTSEQUENZ" eine sequentielle Datei mit dem Namen »TESTSEQUENZ«. Abkürzungen des Filenamens mit »*« sind erlaubt. Falsche Eingaben werden mit einer Fehlermeldung quittiert.

Mit der CTRL-Taste kann die Ausgabe verlangsamt und mit der RUN/STOP-Taste unterbrochen werden. Eine Fortsetzung erfolgt mit der A-, und vorzeitiger Abbruch des Listens mit der DEL-Taste. Am Ende eines Listings muß immer die SPACE-Taste gedrückt werden.

Das Maschinenprogramm für »Floppy-Lister« liegt im Bereich von \$C000-\$C2B2 (49152-49842). An das Programm schließt sich noch ein Pufferbereich an, wo Daten zwischengespeichert werden. »Floppy-Lister« läuft mit den gängigen Erweiterungen, wie DOS 5.1, Simons Basic und Exbasic.

(B. Schulzki)

Joystickabfrage:

Auch die Joystickabfrage ist beim C64 nicht ganz einfach. Dabei hilft folgendes Hilfsprogramm:

```
10 POKE 56322,224
20 JO=PEEK(56320)
30 IF (JO AND 1)=0 THENPRINT"OBEN"
40 IF (JO AND 2)=0 THENPRINT"UNTEN"
50 IF (JO AND 4)=0 THENPRINT"LINKS"
60 IF (JO AND 8)=0 THENPRINT"RECHTS"
70 IF (JO AND 16)=0 THENPRINT"FEUER"
```

Dieses Programm gilt für eine Joystickabfrage am Controlport 2.

Commodore-Joystick verbessert

Der Commodore-Joystick VIC-1311 benötigt eine relativ große Hebelbewegung, um die Kontakte zu schließen. Bei Spielen, die eine hohe Reaktionsgeschwindigkeit erfordern, ist diese Eigenschaft sehr ungünstig. Man kann jedoch recht einfach Abhilfe schaffen:

Man entfernt die vier Schrauben an der Unterseite des Gehäuses und hebt den oberen Teil des Joysticks mit der Platine vorsichtig ab. Nun wird die Platine an den Durchtrittsöffnungen der Schrauben mit je einer etwa 1 Millimeter dicken Unterlegscheibe verstärkt. Die vier Unterlegscheiben können mit einem Tropfen Alleskleber (Vorsicht, nicht die Kontakte verkleben!) gegen Verrutschen gesichert werden. Danach wird der Joystick wieder zusammengeschraubt. Wenn Sie alles richtig gemacht haben und insbesondere keine Teile übriggeblieben sind, dann werden die Kontakte des Joysticks nun bei erheblich kleineren Hebelbewegungen geschlossen.

(M. Kunze)

programm : floppylistер c000 c2b5

```
c000 : 20 fd ae 20 9e ad 20 a3 ee
c008 : b6 a5 64 85 a7 a5 65 85 a3
c010 : a8 a0 00 b1 a7 99 1f c3 8a
c018 : c8 c0 03 90 f6 ad 1f c3 f4
c020 : c9 14 b0 1e c9 04 90 24 2a
c028 : a0 00 ad 21 c3 85 a8 ad be
c030 : 20 c3 85 a7 b1 a7 99 22 8b
c038 : c3 c8 cc 1f c3 d0 f5 4c a9
c040 : 53 c0 a9 6f a0 c2 20 1e 28
c048 : ab 4c 74 a4 a9 88 a0 c2 b2
c050 : 4c 46 c0 a2 00 bd 22 c3 42
c058 : 8d 1e c3 c9 50 f0 08 c9 5f
c060 : 53 d0 e9 a0 04 d0 02 a0 ba
c068 : 00 e8 bd 22 c3 c9 4c f0 2e
c070 : 06 c9 44 d0 d7 c8 c8 e8 3f
c078 : bd 22 c3 c9 3a d0 cd 8c eb
c080 : 0e c3 ae 1f c3 a9 2c 48 ca
c088 : 9d 22 c3 e8 ad 1e c3 9d 5a
c090 : 22 c3 68 e8 9d 22 c3 a9 18
c098 : 52 e8 9d 22 c3 a9 00 e8 65
c0a0 : 9d 22 c3 ee 1f c3 ad 28 34
c0a8 : 03 8d 0f c3 ad 29 03 8d f9
c0b0 : 10 c3 a9 22 8d 28 03 a9 ca
c0b8 : c2 8d 29 03 ad 1f c3 a2 14
c0c0 : 25 a0 c3 20 bd ff a9 02 b1
c0c8 : a2 08 a0 02 20 ba ff 20 ef
c0d0 : c0 ff a2 02 20 c6 ff ac 0a
c0d8 : 0e c3 b9 ab c2 8d 11 c3 10
c0e0 : b9 ac c2 8d 12 c3 6c 11 65
c0e8 : c3 a9 00 8d 13 c3 8d 14 df
c0f0 : c3 8d 15 c3 a2 00 20 0f 00
c0f8 : c2 b0 03 4c e9 c1 20 cf 29
c100 : ff 9d 16 c3 a5 90 d0 05 f8
c108 : e8 e0 08 90 e9 8d 15 c3 5b
c110 : a9 00 e8 e0 08 b0 07 9d 6d
c118 : 16 c3 e8 4c 13 c1 20 4c 2c
c120 : c2 ad 14 c3 20 2a c2 ad f0
c128 : 13 c3 20 2a c2 a9 3a 20 0d
c130 : d2 ff a2 00 bd 16 c3 20 86
c138 : 23 c2 e8 e0 08 90 f5 20 30
c140 : 4c c2 a9 12 20 d2 ff a2 78
c148 : 00 bd 16 c3 29 7f c9 20 1b
c150 : b0 04 a9 2e d0 03 bd 16 7b
c158 : c3 20 d2 ff e8 e0 08 90 b7
c160 : e8 a9 0d 20 d2 ff ad 15 72
c168 : c3 d0 11 18 ad 13 c3 69 30
c170 : 08 8d 13 c3 90 03 ee 14 81
c178 : c3 4c f4 c0 4c e9 c1 20 12
c180 : 51 c2 20 51 c2 a2 ff 20 e6
c188 : 51 c2 a5 a7 e8 9d b3 c2 68
c190 : e0 03 90 f3 20 51 c2 e8 fe
c198 : a5 a7 9d b3 c2 d0 f5 e8 4b
c1a0 : 9d b3 c2 e8 9d b3 c2 a9 ba
c1a8 : b3 85 5f a0 c2 84 60 20 1c
c1b0 : 37 a5 ad 00 03 48 ad 01 50
c1b8 : 03 48 a9 22 8d 00 03 a9 c6
c1c0 : c2 8d 01 03 20 c3 a6 68 75
c1c8 : 8d 01 03 68 8d 00 03 20 c9
c1d0 : 0f c2 b0 b1 4c e9 c1 20 fe
c1d8 : cf ff 20 d2 ff a5 90 d0 1a
c1e0 : 08 20 0f c2 b0 f1 4c e9 b4
c1e8 : c1 20 cc ff a9 02 20 c3 9f
c1f0 : ff ad 0f c3 8d 28 03 ad 84
c1f8 : 10 c3 8d 29 03 a9 60 a0 b3
c200 : c2 20 1e ab a5 cb c9 3c 28
c208 : d0 fa a9 00 85 c6 60 a5 1b
c210 : cb f0 0e c9 3f d0 08 a5 f6
c218 : cb f0 06 c9 0a d0 f8 38 92
c220 : 60 18 60 48 a9 20 20 d2 6f
c228 : ff 68 48 4a 4a 4a 20 17
c230 : 42 c2 a8 68 29 0f 20 42 1b
c238 : c2 48 98 20 d2 ff 68 4c b0
c240 : d2 ff 18 69 f6 90 02 69 14
c248 : 06 69 3a 60 a9 20 4c d2 10
c250 : ff 20 cf ff 85 a7 a5 90 a0
c258 : f0 05 68 68 4c e9 c1 60 ce
c260 : 0d 20 20 20 20 12 20 53 43
c268 : 50 41 43 45 20 0d 00 0d 57
c270 : 46 45 48 4c 45 52 3a 20 05
c278 : 4e 41 4d 45 20 5a 55 20 cd
c280 : 4c 41 4e 47 20 21 0d 00 29
c288 : 0d 46 45 48 4c 45 52 3a bf
c290 : 20 46 41 4c 53 43 48 45 a8
c298 : 53 20 45 49 4e 47 41 42 1f
c2a0 : 45 46 4f 52 4d 41 54 20 97
c2a8 : 21 0d 00 7f c1 e9 c0 d7 5e
c2b0 : c1 e9 c0 32 a9 00 ff 00 77
```

Listing 4. »Floppy-Lister«

List-Schutz für Basic-Programme

Dieser List-Schutz (Listing 5) ist für Nichteingeweihte sehr verblüffend. Die Grundidee dazu stammt aus dem Bericht »Disketten-Manipulationen« aus der 64'er, Ausgabe 6/85. Er wurde jedoch etwas ausgebaut, so daß hier beim Listen alle SteuerCodes aktiv werden. Dies wird dadurch erreicht, daß man in eine Speicherstelle vor den Codes die Zahl 141 schreibt. Um nun ein Programm zu schützen, lädt man es und gibt zwei Zeilen ein (Listing 5).

Danach gibt man im Direktmodus ein:

```
POKE 2067,71 : POKE 2073,141 : POKE 2118,0 :
POKE 2119,0
```

Listet man nun das Programm, so wird der Bildschirm gelöscht und der Text in der REM-Zeile ausgegeben. Durch das künstlich erzeugte Basic-Programm-Endezeichen (drei Nullen) wird das Listen abgebrochen. Wird im Programm dann auch noch durch POKE 788,52 die RUN-STOP-, und durch POKE 792,193 die RESTORE-Taste ausgeschaltet, kann keiner mehr an das Programm. Aufheben läßt sich dieser List-Schutz nur mit einem Monitor und mit der Kenntnis der Funktionsweise des Schutzes.

(Thomas Uttendorfer)

```
1 POKE 2067,73:GOTO 10 <134>
2 REM "A{CLR,3DOWN,YELLOW}IT IS NOT ALLOWED <123>
   TO LIST THIS PROGRAM{BLUE}AA
```

64'er

Listing 5. Listing zu »List-Schutz für Basic-Programme«

List- und Löschschutz leichtgemacht

Es wurden schon viele Methoden veröffentlicht, um ein Basic-Programm gegen Listen zu schützen. Aber alle mir bekannten Möglichkeiten weisen unterschiedene Nachteile auf. Entweder der Schutz ist nicht sicher genug und leicht zu entfernen, oder er ist viel zu aufwendig.

Ich habe mich daher entschlossen, ein Programm zu schreiben, das diese Mängel umgeht und sogar noch andere positive Merkmale aufweist.

Zunächst eine Zusammenfassung von drei mir bekannten Listschutzmöglichkeiten mit ihren Vor- und Nachteilen:

1. Möglichkeit

In die erste Zeile des Basic-Programms (zum Beispiel Zeilennummer 1) wird REM, gefolgt von zwei Anführungszeichen und SHIFT L, geschrieben.

```
1 REM " " L (RETURN)
```

Der Cursor wird nun auf das zweite Anführungszeichen gesetzt und sechsmal SHIFT INST gedrückt (das Anführungszeichen wird um sechs Positionen nach rechts geschoben). Dann wird sechsmal DEL eingegeben (es erscheinen als Steuerzeichen sechs reverse T) und die Zeile mit (RETURN) abgespeichert. Wird nun der LIST-Befehl aufgerufen, meldet sich der Rechner mit:

```
?SYNTAX ERROR
```

```
READY.
```

Auf den ersten Blick sehr beeindruckend, aber durch Entfernen dieser Zeile ist der Listschutz wieder aufgehoben. Außerdem ist ein »LIST 2« noch möglich.

2. Möglichkeit

In jede Basic-Zeile werden synthetische Steuerzeichen eingefügt (genaue Beschreibung im 64'er-Magazin, Ausgabe

```
0 REM ***** <131>
1 REM * (C) U. V. GAISBERG * <195>
2 REM * AM ZUCKERBERG 70 * <080>
3 REM * 7140 LUDWIGSBURG * <233>
4 REM * TEL. 07141/55910 * <056>
5 REM ***** <136>
6 FOR I=0 TO 340:READ A:B=B+A:POKE 50000+I
  ,A <099>
7 NEXT I <091>
8 IF B <> 33527 THEN PRINT"FEHLER IN DATAS
  !":END <042>
9 PRINT"OK !":END <113>
10 REM DATAS FUER MASCHINENPROGRAMM <157>
11 DATA 169,0,141,32,208,141,33,208,169,1,
  141,134,2,32,68,229,174,3,8,172 <018>
12 DATA 4,8,192,0,208,7,224,2,176,3,76,206
  ,195,162,0,142,134,2,169,32,32 <221>
13 DATA 210,255,232,224,50,208,246,162,0,1
  89,21,196,157,0,4,232,224,29,208 <001>
14 DATA 245,169,24,157,0,4,232,224,69,208,
  246,162,0,189,50,196,157,80,4,232 <190>
15 DATA 224,8,208,245,162,0,189,58,196,157
  ,120,4,232,224,8,208,245,162,10 <073>
16 DATA 160,0,24,32,240,255,169,19,162,13,
  160,4,141,119,2,142,120,2,142,121 <061>
17 DATA 2,142,122,2,132,198,96,162,0,189,9
  9,196,32,210,255,232,224,31,208 <065>
18 DATA 245,96,32,68,229,162,10,160,0,24,3
  2,240,255,162,1,142,134,2,202,189 <210>
19 DATA 66,196,32,210,255,232,224,33,208,2
  45,169,20,162,17,160,255,141,18 <250>
20 DATA 8,142,29,8,140,4,8,162,0,189,130,1
  96,157,32,8,232,224,34,208,245 <171>
21 DATA 96,48,18,5,13,34,148,148,148,148,1
  48,148,148,148,148,34,12,12 <122>
22 DATA 9,19,20,19,3,8,21,20,26,26,76,49,1
  9,25,19,50,48,57,56,19,25,19,53 <045>
23 DATA 48,49,52,48,80,82,79,71,82,65,77,7
  7,32,45,32,40,67,41,32,85,46,86 <224>
24 DATA 46,71,65,73,83,66,69,82,71,32,32,4
  9,57,56,52,66,73,84,84,69,32,90 <032>
25 DATA 69,73,76,69,32,48,32,85,78,68,32,4
  9,32,69,78,84,70,69,82,78,69,78 <005>
26 DATA 32,33,169,255,141,4,8,169,131,162,
  164,141,2,3,142,3,3,76,131,164 <172>
27 DATA 165,2,141,4,8,169,32,162,8,141,2,3
  ,142,3,3,96,0 <150>
```

64'er

Listing 6. Listing zu »List- und Löschschutz leichtgemacht«

6/84). Diese Methode ist zwar recht sicher, will man aber alle Zeilen eines längeren Basic-Programms schützen, ist der Aufwand viel zu groß, vom Speicherplatzbedarf der Steuerzeichen ganz abgesehen.

3. Möglichkeit

Durch POKE 775,200 wird der Listbefehl außer Kraft gesetzt, durch POKE 775,167 wird diese Wirkung wieder aufgehoben. Dieser Listschutz ist zwar wirkungsvoll, aber er muß erst durch diesen POKE-Befehl aktiviert werden. Nach dem Laden eines Programms ist er daher noch nicht aktiv.

Das hier vorgestellte Programm erzeugt nicht nur einen sicheren Listschutz, sondern schützt auch vor dem Löschen einzelner Basic-Zeilen. So können zum Beispiel Hinweise auf ein Kopierrecht und auf den Autor eines Programms nicht geändert oder entfernt werden. Auch kann ein so gesichertes Programm nur mit RUN gestartet werden, ein RUN, gefolgt von einer Zeilennummer, führt zu einer Fehlermeldung. Jede Zeile des Programms ist geschützt, es können also auch einzelne Zeilen nicht gelistet werden. Einzige Bedingung für die Verwendung dieses Schutzes: Das zu schützende Programm darf keine Zeilennummern 0 und 1 enthalten. Ansonsten wird eine Fehlermeldung ausgegeben und das Programm bleibt unverändert.

Das Listschutzprogramm (Listing 6) liegt als Basic-Lader vor. Nachdem es richtig abgetippt wurde, kann es durch RUN gestartet werden. Das Maschinenprogramm steht dann im Speicher ab der Adresse 50000 zur Verfügung. Das zu schützende Basic-Programm kann nun geladen werden, durch SYS 50000 wird das Schutzprogramm aktiviert und das Basic-Programm geschützt. Es kann nun wieder auf Kassette/Diskette gespeichert werden. Das mit dem Listschutz versehene Programm ist nur um wenige Bytes größer als vorher.

Funktionsweise

Das Maschinenprogramm (Listing 6) generiert zwei Basic-Zeilen mit den Zeilennummern 0 und 1. Die Zeile 0 ist eine REM-Zeile, in der ein unlistbares Zeichen (SHIFT L) steht. Hinter diesem Zeichen stehen dann noch zwei kurze Maschinenprogramme, deren Funktionen im folgenden noch erklärt werden. In der zweiten Zeile steht ein SYS-Befehl, der eine der beiden Maschinenroutinen in Zeile 0 startet. Sind diese beiden Zeilen nun erzeugt, wird die Zeilennummer 0 durch eine höhere, eigentlich unerlaubte Zeilennummer (größer 64000) ersetzt. Diese Zeile kann daher auch nicht gelöscht werden.

Da alle nun folgenden Zeilen des Programms kleiner sind als die erste, können diese vom Computer nicht mehr erkannt werden. Ein Sprung in eine solche Zeile führt zu der Fehlermeldung: ?UNDEF'D STATEMENT ERROR. Es kann daher auch keine Zeile gelöscht werden, da diese für den Computer ja nicht mehr vorhanden sind.

Der einzige Nachteil ist, daß es nicht nur ein perfekter List- und Löschschutz, sondern auch ein RUN-Schutz ist (auch Sprungziele innerhalb des Programms können nicht gefunden werden).

Wird das geschützte Programm gestartet, trifft der Interpreter als erstes auf den SYS-Befehl in Zeile 1. Es folgt ein Sprung in das Maschinenprogramm in der REM-Zeile. Dort wird die Zeilennummer wieder auf 0 gesetzt, und der Vektor auf den Basic-Warmstart wird auf die zweite Maschinenroutine gesetzt.

Nun kann das Basic-Programm ohne Fehler ausgeführt werden. Wird der Programmlauf unterbrochen (durch STOP-Taste, Fehlermeldungen, Programmende und so weiter), wird das zweite Maschinenprogramm über den Basic-Warmstartvektor angesprungen. Dort wird die Zeilennummer wieder hochgesetzt, der Warmstartvektor wieder auf den normalen Wert gebracht und die Warmstartroutine angesprungen. Das Programm liegt nun wieder in der geschützten Form vor.

(Ulrich von Gaisberg)

Maschinenprogramme auf Tastendruck

Mit einer kleinen Routine kann man ein Maschinenprogramm mit einem Tastendruck aufrufen. Dafür benutzt man ein Zeichen, das normalerweise nicht oder nur selten verwendet wird. Ich habe mich für das @-Zeichen entschieden.

Im Interpreter existiert eine Schleife, die einen Basic-Befehl holt und ausführt.

```
A7E1 JMP (0308) ; zeigt normalerweise auf A7E4
A7E4 JSR 0073 ; nächstes Zeichen aus Basic-Text holen
A7E7 JSR A7ED ; Statement ausführen
A7EA JMP A7AE ; zurück zur Interpreterschleife
```

In den Speicherzellen \$0308 und \$0309 (776 und 777 dez) liegt ein Zeiger, der normalerweise auf den Beginn dieser Schleife zeigt. Verbiegt man nun den Zeiger auf eine eigene Routine, kann man den Basic-Befehl auf das eigene Zeichen überprüfen.

Wird es erkannt, springt man auf den Anfang des gewünschten Unterprogramms. Wurde das Zeichen nicht vorgefunden, macht man in der Schleife normal weiter.

Dieses Verfahren verwende ich bei der Programmierhilfe »Merge« aus Ausgabe 4/84, die normalerweise mit SYS 50000 gestartet werden muß. Es kann aber auch für andere Maschinenprogramme umgeschrieben werden.

»Merge« belegt den Speicherbereich 50000 bis 50264. Die eigene Routine beginnt auf Adresse 49152 (C000 hex). Der Wert in den Adressen \$0308 und \$0309 muß deshalb auf C000 abgeändert werden. Der Computer durchläuft dann vor jedem Befehl, den er ausführen soll, folgende Schleife:

```
C000 JSR 0073 ; nächstes Zeichen holen
C003 CMP 40 ; Vergleich mit @-Zeichen
C005 BEQ ; verzweigen wenn erkannt
C007 JSR 0079 ; Flags setzen
C00A JMP A7E7 ; Rücksprung
C00D JSR 0073 ; nächstes Zeichen holen
C010 JSR C350 ; zur eigenen Routine
C013 JMP A7AE ; Rücksprung
```

Nach dem Drücken von @ und RETURN wird nun das Programm ab Adresse 50000 (C350 Hex) ausgeführt. Auf die anderen Befehle hat diese Routine keinen Einfluß. Eine Hürde gibt es noch zu meistern. Die Änderung der Adressen 0308 und 0309 ist auf der Basic-Ebene mit POKE nicht möglich. Dies ist auch verständlich, da POKE auch ein Basic-Befehl ist und durch die Änderung der ersten Adresse der Einsprung verändert wird.

Deshalb muß diese Adreßänderung in Maschinensprache durchgeführt werden.

```
C100 LDA 00 ; Lade Akku mit 00
C102 STA 0308 ; Speichere Akku nach 0308
C105 LDA C0 ; Lade Akku mit C0
C107 STA 0309 ; Speichere Akku nach 0309
C10A RTS ; Rückkehr nach Basic
```

Basic-Lader für Befehlserweiterung

```
240 FOR I= 49152 TO 49152 + 21
250 READ Q : POKE I, Q
260 NEXT
300 FOR I = 49408 TO 49408 + 10
310 READ Q : POKE I, Q
320 NEXT : SYS 49408
11000 DATA 32,115,0,201,64,240,6,32,121
12000 DATA 0,76,231,167,32,115,0,32,80
13000 DATA 195,76,174,167
14000 DATA 169,0,141,8,3,169,192,141,9,3,96
```

Diesen Basic-Lader tippt man hinter das Programm »Merge« und speichert es gemeinsam ab.

Die Zeilennummern sind so gewählt, daß man sie direkt zum Basic-Lader von »Merge« dazutippen kann. In Zeile 10260 müssen aber dann die letzten fünf Nullen gelöscht werden.

Mit SYS 49408 wird die Befehlserweiterung aktiviert und steht dann zur Benutzung bereit.

(Patrik Fleig)

Maschinenprogramme auf Diskette speichern

Reine Maschinenprogramme haben gegenüber einem Basic-Lader zwei entscheidende Vorteile. Sie sind wesentlich kürzer und können direkt, ohne zeitraubendes POKEn, geladen werden. Dieses Programm soll Ihnen helfen, Maschinenprogramme auf Diskette zu speichern.

Ein einfacher Basic-Lader besteht aus einer READ/POKE-Schleife, in der DATAs in den Speicher geschrieben werden.

```

50000 REM      BASIC-LADER                <034>
50010 DATA    .....                    <058>
50020 DATA    .....                    <068>
50030 DATA    .....                    <078>
50040 :        .....                    <231>
50050 REM      MASCHINENPROGRAMME AUS    <065>
50060 REM      DATALADERN                <219>
50070 :        .....                    <005>
50080 REM S KOENNEN WEGGELASSEN WERDEN  <056>
50090 :        .....                    <025>
50100 REM      DA = STARTADRESSE          <249>
50110 REM      EA = ENDADRESSE           <125>
50120 :        .....                    <055>
50130 OPEN 1,8,1, "NAME"                <131>
50140 :        .....                    <075>
50150 REM SA U.EA ZUORDNEN              <036>
50160 :        .....                    <095>
50170 SH = INT(SA/256) :REM HIGH-BYTE    <166>
50180 SL = SA-SH*256 :REM LOW-BYTE       <064>
50190 PRINT#1,CHR$(SL);CHR$(SH);        <065>
50200 :        .....                    <137>
50210 FOR I=SA TO EA                    <249>
50220 READ WERT                         <064>
50230 PRINT#1,CHR$(WERT);              <079>
50240 NEXT I                            <029>
50250 :        .....                    <187>
50260 CLOSE 1                          <232>

```

© 64'er

Listing 7. Listing zu »Maschinenprogramme auf Diskette speichern«

Jeder Wert, der in dieser Schleife gePOKEt wird, ist ein Byte eines Maschinenprogramms, ein Maschinenbefehl oder ein Teil davon. Anstatt die DATAs in den Speicher zu POKEn, kann ein Basic-Lader dazu benutzt werden, das Maschinenprogramm auf Diskette abzuspeichern. Der Lader wird dazu nur etwas geändert.

Ein Maschinenprogrammfile auf Diskette enthält, vor den Maschinenbefehlen, in den ersten beiden Bytes, die Startadresse des Programms. Der Computer erkennt daran, ab welcher Speicherstelle er das Programm laden soll.

Der Basic-Lader (Listing 7) muß also ein Programmfile eröffnen, die Startadresse des Maschinenprogramms ins File schreiben und anschließend alle Werte aus den DATA-Zeilen. Wie das im einzelnen programmiert wird, können Sie am Listing des Beispielprogramms sehen, das Sie ohne weiteres an jedes Ladeprogramm anpassen können.

Der wichtigste Befehl in diesem Beispiel steht in Zeile 50130:

```
OPEN 1,8,1, "Programmname"
```

Er öffnet eine Datei. Allerdings keine sequentielle, sondern eine Programmdatei. Die Sekundäradresse 1 sagt der Floppy 1541, daß nun ein Programm (NAME.PRG) geschrieben wird. Häufig findet man für diesen Befehl die Syntax

```
OPEN 2,8,2, "Programmname,P,W"
```

bei der man die Sekundäradresse frei wählen darf. Hier muß jedoch nach dem Namen angegeben werden, daß eine Programmdatei (P) geöffnet werden soll, in die geschrieben wird (Write). Entsprechend kann bei sequentiellen Dateien optional »S,W« bei Schreib- oder »S,R« bei Lesezugriffen angegeben werden.

Nachdem in Zeile 50130 die Programmdatei angelegt wurde, muß die Startadresse des Maschinenprogrammes gespeichert werden. Die Startadresse wird dazu in den Zeilen 50170 und 50180 in Low- und High-Byte zerlegt und die zwei Byte in Zeile 50190 in das Programmfile geschrieben. Jetzt wird das eigentliche Programm gespeichert. Die numerischen Werte müssen dazu mit der CHR\$()-Funktion in die entsprechenden Strings übersetzt werden. Um die erforderliche Datendichte beim Schreiben des Files zu erreichen, muß

jedem String ein Semicolon »;« folgen. Ohne Semicolon ist das Programm später nicht lauffähig. Sind alle Werte, die der Basic-Lader in den Speicher gePOKEt hätte, gespeichert, wird die Datei mit CLOSE geschlossen. Damit ist das Maschinenprogramm auf Diskette gespeichert und kann mit

```
LOAD "NAME",8,1
```

geladen und, wenn dies notwendig ist, mit dem entsprechenden SYS-Befehl gestartet werden.

Laden von Maschinenprogrammen in Basic

Sie können Maschinenroutinen mit Hilfe von Basic-Programmen laden, ohne sie zu zerstören. Dabei ist zu beachten, daß nach dem Laden das Basic-Programm von neuem startet. Die Basic-Zeile

```
10 LOAD "KEIN ENDE",8,1
```

bewirkt deshalb eine Endlosschleife, die immer wieder das Maschinenprogramm »KEIN ENDE« lädt. Da angelegte Variablen dabei erhalten bleiben, kann mit

```
10 IF A=0 THEN A=1:LOAD "KEIN ENDE",8,1
```

ein mehrmaliges Laden verhindert werden, da beim Neustart A=1 ist.

(S. Wengler)

Der C 64 als PET

Wenn Sie CBM 2000, 3000 oder 4000 geschriebene Programme auf Ihrem C 64 laufen lassen wollen, müssen Sie umständlich PEEKs und POKEs ändern. Der »Pet-Simulator« nimmt Ihnen diese Arbeit ab.

Ist das Programm (Listing 8) eingegeben und gestartet, werden als erstes die DATAs für das Maschinenprogramm in den Bereich ab Adresse 49152 gePOKEt (SU = Prüfsumme für die Daten). Danach fragt das Programm nach der Zeichenfarbe. Sie werden aufgefordert, eine Zahl zwischen 0 und 15 einzugeben.

(0 = schwarz 1 = weiß,..., 15 = grau 3).

```

1 REM ***** <132>
2 REM *      PET - SIMULATOR      * <251>
3 REM *      *                      * <052>
4 REM *      (C) BY W. HOPF 1984   * <218>
5 REM ***** <136>
6 : <238>
7 REM PROGRAMM VOR DEM START ABSPEICHERN <071>
8 : <240>
10 FOR I=49152 TO 49152+91:READ A:SU=SU+A <231>
20 POKE I,A:NEXT <117>
30 IF SU<>12552 THEN END <064>
40 PRINT "{CLR,2DOWN}BITTE WAEHLEN SIE DIE <063>
   ZEICHENFARBE" <124>
50 PRINT "{DOWN}{0-15 EINGEBEN}!{DOWN}" <124>
60 INPUT ZF:IF ZF>=0 AND ZF<=15 THEN POKE <089>
   49239,ZF:POKE 646,ZF:GOTO 80
70 PRINT "{DOWN}NICHT ERLAUBT":FOR I=1 TO 1 <066>
   000:NEXT:GOTO 40
80 SYS 49152:POKE 1,54:PRINT "{CLR}PET-SIMU <208>
   LATOR AKTIV":NEW <121>
10000 DATA 160,0,132,254,169,160,133,255 <103>
10005 DATA 177,254,145,254,200,208,249,230 <189>
10010 DATA 255,166,255,224,192,208,241 <221>
10015 DATA 169,5,141,0,221,141,24,208,169 <229>
10020 DATA 128,141,136,2,133,56,169,4 <241>
10025 DATA 133,44,169,0,141,0,4,169 <018>
10030 DATA 63,141,37,184,169,192,141,38 <211>
10035 DATA 184,169,193,141,24,3,96,32 <095>
10040 DATA 235,183,24,165,21,201,128,144 <177>
10045 DATA 18,201,132,176,14,24,105,88 <209>
10050 DATA 133,255,165,20,133,254,169,0 <037>
10055 DATA 234,145,254,96 <029>
60000 : <140>
60010 REM LISTE DER VERWENDETEN COMMODORE- <245>
   STEUERZEICHEN <186>
60020 REM "{CLR}" = CLR
60030 REM "{DOWN}" = CRSR-DOWN

```

Listing 8. Listing zu »Der C 64 als PET«

Bei anschließendem Starten des Maschinenprogrammes wird das Basic-ROM in das darunterliegende RAM gePOKEt (Basic-Interpreter kopieren). Anschließend wird das Bildschirm-RAM von Adresse 1024 nach Adresse 32768 verlegt. Basic-Speicheranfang und -ende werden dem des PET angepaßt. Weiterhin wird in der POKE-Routine des Basic-Interpreters ein Eingriff vorgenommen, nach der der Computer aus dem Interpreter in eine Routine des Maschinenprogramms springt. Hier wird überprüft, ob das Bildschirm-RAM angesprochen wurde. Trifft dies zu, wird die dazugehörige Farb-RAM-Adresse berechnet und der vorher festgelegte Farbwert (Zeichenfarbe) hineingePOKEt. Um das Zurücksetzen des Bildschirms auf das C 64-Format zu vermeiden (durch Drücken der RUN/STOP- und RESTORE-Tasten), wird die RESTORE-Taste durch Verändern des NMI-Vektors ausgeschaltet. Programme können aber noch mit der RUN/STOP-Taste unterbrochen werden.

Nach Ablauf des Maschinenprogrammes meldet sich der Computer mit »PET-SIMULATOR AKTIV«. Sie können jetzt immer noch die Zeichenfarbe mit POKE 49239, ZF (ZF = Zeichenfarbe – siehe oben) ändern. Wenn Sie jetzt zum Beispiel POKE 32768,1 eingeben, erscheint ein »A« am linken oberen Bildschirmrand in der gewählten Zeichenfarbe. Schlußbemerkung: Bevor Sie das Programm starten, empfiehlt es sich, es vorher zu speichern, da sich das Programm selbständig löscht.

(Wolfgang Hopf)

Pseudo-Interrupt

Diese Befehlserweiterung erlaubt es, ein Basic-Programm zu jedem beliebigen Zeitpunkt per Tastendruck durch die F1-Taste unterbrechen zu lassen. Es kann dann in eine vorher definierte Basic-Routine gesprungen werden. Diese könnte zum Beispiel den noch freien Speicherplatz oder die Uhrzeit anzeigen.

Das Programm (Listing 9) bitte mit dem MSE eingeben. Es liegt dann im Speicherbereich von 40499 bis 40768. Geladen wird es absolut mit LOAD "PSEUDO-IRQ",8,1. Da das Programm im Bereich für die Basic-Variablen steht, muß es durch POKE 56,158:CLR vor Überschreiben geschützt werden. Nach dem Start durch »SYS 40541« stehen die neuen Befehle zur Verfügung:

!F1JUMP <Zeilennummer> legt fest, in welche Zeile im Falle einer Unterbrechung durch die F1-Taste gesprungen werden soll. Tritt der Befehl mehrmals auf, so gilt die zuletzt angegebene Zeilennummer.

!JBACK bewirkt die Fortsetzung des Basic-Programms ab der Stelle, an der unterbrochen wurde.

!SF1 verhindert Unterbrechungen. Dies kann zum Beispiel beim Aufbau einer Grafik oder bei Arbeiten mit der Diskettenstation wichtig sein.

!CF1 läßt gesperrte Unterbrechungen wieder zu.

(Guido Schuhmacher)

Es muß nicht immer »READY.« sein...

Wenn Sie sich darüber ärgern, daß der C 64 nach jedem ausgeführten Befehl sein stupides »READY.« auf den Bildschirm schreibt, dann geben Sie doch die beiden folgenden Zeilen im Direktmodus ein:

```
FOR I=40960 TO 49151 : POKE I,PEEK(I) : NEXT :
POKE 1,54 : REM Basic-Interpreter ins RAM laden
FOR I= 41848 TO 41853 : POKE I,32 : NEXT :
REM READY-Meldung überschreiben
```

programm : pseudo-irq 9e01 9f41

```
9e01 : ff 89 00 20 73 00 c9 21 6a
9e09 : d0 03 4c 6d 9e a5 cb c9 f5
9e11 : 40 d0 06 20 79 00 4c e7 d8
9e19 : a7 c9 04 f0 06 20 79 00 0b
9e21 : 4c e7 a7 a5 cb c9 40 d0 ad
9e29 : fa ad 01 9e f0 06 20 79 c1
9e31 : 00 4c e7 a7 18 a9 03 20 61
9e39 : fb a3 a5 7a 48 a5 7b 48 ef
9e41 : a5 39 48 a5 3a 48 a9 75 c1
9e49 : 48 8d 01 9e ad 02 9e 85 dc
9e51 : 14 ad 03 9e 85 15 20 a3 99
9e59 : a8 4c ae a7 a9 04 8d 08 c9
9e61 : 03 a9 9e 8d 09 03 a9 75 cc
9e69 : 8d 01 9e 60 20 73 00 c9 5c
9e71 : 46 f0 07 c9 4a f0 41 4c f4
9e79 : f1 9e 20 73 00 c9 31 d0 e5
9e81 : 34 20 73 00 c9 4a d0 2d 2f
9e89 : 20 73 00 c9 55 d0 26 20 51
9e91 : 73 00 c9 4d d0 1f 20 73 8e
9e99 : 00 c9 50 d0 18 20 73 00 fc
9ea1 : 20 6b a9 a5 14 8d 02 9e 89
9ea9 : a5 15 8d 03 9e a9 00 8d ef
9eb1 : 01 9e 4c ae a7 4c 08 af 47
9eb9 : 20 73 00 c9 42 d0 f6 20 93
9ec1 : 73 00 c9 41 d0 ef 20 73 c3
9ec9 : 00 c9 43 d0 e8 20 73 00 f6
9ed1 : c9 4b d0 e1 68 c9 75 d0 fc
9ed9 : 59 68 85 3a 68 85 39 68 77
9ee1 : 85 7b 68 85 7a a9 00 8d ff
9ee9 : 01 9e 20 79 00 4c e7 a7 c2
9ef1 : c9 53 f0 07 c9 43 f0 1c 34
9ef9 : 4c 08 af 20 73 00 c9 46 24
9f01 : d0 b3 20 73 00 c9 31 d0 d6
9f09 : ac a9 75 8d 01 9e 20 73 05
9f11 : 00 4c ae a7 20 73 00 c9 09
9f19 : 46 d0 9a 20 73 00 c9 31 33
9f21 : d0 93 a9 00 8d 01 9e 20 c1
9f29 : 73 00 4c ae a7 4a 42 41 de
9f31 : 43 cb a9 2e 85 22 a9 9f d9
9f39 : 85 23 4c 47 a4 01 00 a5 e9
```

Listing 9. Listing zu »Pseudo-Interrupt«

Wenn Sie jetzt irgendeinen Befehl eingeben, erscheint kein »READY.« mehr. Natürlich können Sie aber auch einfach den Text ändern:

```
A$="HALLO." : FOR I=1 TO 6 : POKE 41847+I,ASC
(MID$(A$,I,1)) : NEXT
```

Nach Eingabe dieser Zeile meldet sich der Interpreter nach jeder Eingabe mit »Hallo.«. Sie können jeden beliebigen Text wählen, vorausgesetzt, er ist maximal sechs Zeichen lang.

(Andreas Scharrer)

Reset

Falls Sie in einem Programm einen Reset wünschen, dann benutzen Sie folgenden Befehl:

```
SYS 64738
```

Mit folgendem kleinen Programm sorgt man von Basic aus dafür, daß ein Reset keinen Effekt hat:

```
100 FOR I=32770 TO 32778
110 READ A : POKE I,A
120 NEXT I
130 DATA 10, 128, 195, 194, 205
140 DATA 56, 48, 88, 0
```

(Daniel Kossmann)

Einfacher Reset-Schalter selbstgebaut

Sehr preiswert und einfach kommt man im Selbstbau zu einem Reset-Schalter, wenn man sich im Elektronikfachgeschäft einen sechspoligen Diodenstecker und einen Miniatur-Tastschalter besorgt. Der Tastschalter wird zwischen die Pins 2 und 6 des Diodensteckers gelötet. Zur Pinbelegung vergleiche auch Seite 142 im C 64-Handbuch. Der

so präparierte Diodenstecker wird nun in den seriellen Port des C 64 beziehungsweise VC 20 gesteckt. Falls der Port bereits von Floppy oder Drucker belegt ist, kann der Stecker natürlich auch in die freie Buchse des entsprechenden Zusatzgerätes eingesteckt werden. Auf Knopfdruck wird nun in jedem Fall wieder der Einschaltzustand des Computers hergestellt.

Die Kosten für Stecker und Taste betragen etwa 4 Mark.
(Henning Zipf)

Reset-Helfer für C 64

Das Betriebssystem des C 64 enthält ab der Adresse \$FD02 ein Unterprogramm, das im Steckmodulbereich ab \$8000 nach der Zeichenfolge »CBM80« sucht. Nach dem Einschalten des Computers oder nach einem Reset wird dieses Unterprogramm jedesmal aufgerufen. Werden ab der Adresse \$8003 die Zeichen »CBM80« gefunden, dann wird nicht zum Basic-Start gesprungen, sondern das Betriebssystem nimmt an, daß ein Modul eingesteckt ist, holt sich aus der Speicherzelle \$8000/\$8001 die Startadresse des Modulprogramms und verzweigt dorthin.

Das kleine »Reset-Helfer«-Programm (Listing 10) nutzt dies aus, indem es die genannten Speicherstellen in geeigneter Weise abändert. Es schreibt die »CBM80«-Zeichenfolge ab \$8003 in RAM und läßt die Speicherstellen \$8000/\$8001 auf den Basic-Warmstart zeigen.

Wenn man jetzt einen Reset auslöst, kommt man wie gewohnt aus allen »abgestürzten« Programmen heraus, ein vorhandenes Basic-Programm bleibt aber erhalten.

(Henning Zipf)

RESTORE für Unterprogramme

Will man in ein Programm ein schon vorhandenes Unterprogramm einfügen, kann es beim Lesen von DATAs Schwierigkeiten geben. Oft genügt die RESTORE-Anweisung nicht.

Wenn beide Programmteile, Hauptprogramm und Unterprogramme, DATAs enthalten, muß sichergestellt werden, daß auch wirklich die richtigen Werte gelesen werden. Wenn man nicht aufpaßt, kann es passieren, daß das Unterprogramm DATAs aus dem Hauptprogramm liest. Wie kann man das verhindern? Es gibt eine umständliche Methode: Man kann eine kleine Basic-Erweiterung einbauen, den RESTORE X-Befehl. Es geht aber auch einfacher. Die Zeropage, das

```

100 REM *** RESET-HELPER ***           <141>
110 REM                               <172>
120 REM HENNING ZIPF                   <168>
130 REM KIRCHSTR. 8                    <036>
140 REM 6086 RIEDSTADT 5               <150>
150 REM TEL. (06158) 72453             <243>
160 REM                               <222>
170 FOR I=1 TO 9                       <016>
180 READ D                             <244>
190 POKE 32767+1,D                     <196>
200 NEXT I                             <028>
210 POKE 53280,14:POKE 53281,6         <166>
220 PRINT CHR$(147);CHR$(5)           <071>
230 PRINT " C 64 CHANGED RESET VEKTOR" <067>
240 PRINT                              <086>
250 PRINT " 64 K RAM SYSTEM 38911 BASIC BYT
    ES FREE"                           <233>
260 PRINT                             <108>
270 NEW                               <154>
280 DATA 0,0,255,0,195,194,205,56,48 <015>
290 REM ERST SAVE, DANN RUN !         <160>

```

Listing 10. Listing zu »Reset-Helfer für C 64«

sind die ersten 256 Bytes des Speichers, hilft uns bei der Lösung des Problems. Genauer gesagt, die Adressen 65/66 und 122/123. Schlagen wir im C 64-Handbuch auf Seite 162 nach, dann steht dort:

65– 66 Adresse des aktuellen DATA-Elements
122–123 Basic-Zeiger innerhalb der Subroutine

Mit diesen Informationen läßt sich schon etwas anfangen. Wenn das Unterprogramm angesprungen wird, dann sollte der Zeiger in Speicherstelle 122/123 auf die Adresse des Unterprogramms im Speicher stehen. POKET man diese Werte in die Zeilen 65/55 mit

```
POKE 65,PEEK(122)
```

```
POKE 66,PEEK(123),
```

so wird beim nächsten READ der Wert gelesen, der hinter dieser Basic-Zeile mit den POKES steht, also das erste DATA-Element innerhalb des Unterprogramms. Nach dem Rücksprung aus dem Unterprogramm muß der Zeiger eventuell auch im Hauptprogramm wieder gestellt werden.

In dem kurzen Demo-Listing (Listing 11) werden drei Unterprogramme in zufälliger Reihenfolge aufgerufen.

(Stephan Pätzold)

RAM-Floppy

Wer kennt das nicht: ein paar Veränderungen an einem Programm – eine Zeile rein, eine andere raus – und nichts geht mehr. Das lästige Neuladen des Originalprogramms von Diskette können Sie ab jetzt vergessen.

Ist ein Programm mal wieder zu Tode editiert, werden Sie nun nicht mehr von den langen Ladezeiten der 1541 in Ihrem

```

1 REM ***** <128>
2 REM * DEMO * <010>
3 REM * SUBROUTINE-RESTORE * <071>
4 REM ***** <131>
5 PRINT "{CLR,6SPACE}TASTE DRUECKEN !" <104>
6 PRINT:PRINT <114>
10 X=INT(RND(TI)*3)+1 <125>
20 ON X GOSUB 1000,2000,3000 <040>
25 POKE 65,PEEK(122):POKE 66,PEEK(123) <001>
30 READ A$:PRINT A$ <066>
50 DATA " HAUPTPRG." <146>
100 GOTO 10 <078>
1000 REM *** SUBROUTINE 1 *** <183>
1005 : <042>
1010 POKE 65,PEEK(122):POKE 66,PEEK(123) <221>
1020 FOR I=1 TO 4:READ A:PRINT A$:NEXT <145>
1030 READ A$:PRINT A$; <105>
1040 POKE 198,0:WAIT 198,1 <116>
1050 DATA 1,11,111,1111,"UP1 {2SPACE}" <085>
1060 RETURN <182>
1070 : <108>
2000 REM *** SUBROUTINE 2 *** <164>
2005 : <022>
2010 POKE 65,PEEK(122):POKE 66,PEEK(123) <201>
2020 FOR I=1 TO 4:READ A:PRINT A$:NEXT <125>
2030 READ A$:PRINT A$; <084>
2040 POKE 198,0:WAIT 198,1 <095>
2050 DATA 2,22,222,2222,"UP 2 " <076>
2060 RETURN <162>
2070 : <088>
3000 REM *** SUBROUTINE 3 *** <145>
3005 : <002>
3010 POKE 65,PEEK(122):POKE 66,PEEK(123) <181>
3020 FOR I=1 TO 4:READ A:PRINT A$:NEXT <105>
3030 READ A$:PRINT A$; <064>
3040 POKE 198,0:WAIT 198,1 <075>
3050 DATA 3,33,333,3333,"UP {2SPACE}3" <066>
3060 RETURN <141>
3070 : <067>

```

64

Listing 11. Demoprogramm zu »RESTORE für Unterprogramme« mit zwei POKES

Programmierdrang gebremst. Mit »RAM-Floppy« (Listing 12) kann ein Programm bearbeitet werden, während man eine Kopie davon im RAM hat. In Sekundenschnelle kann die Kopie in den Basic-Speicher gebracht oder mit der Originalversion vertauscht werden. Ganz einfach durch Eingabe von »@V« oder »@T«. Numerische Variablen bleiben dabei erhalten. Die »RAM-Floppy« besitzt eine Speicherkapazität von maximal 25 KByte. Der Speicher beginnt ab Adresse 40960.

Ein Problem ergibt sich im Speicherbereich des Kern- und Basic-ROMs. Ein POKE-Befehl schreibt ins RAM, während die PEEK-Funktion auf das ROM zugreift. Noch komplizierter sieht es beim Zeichen-ROM und den I/O-Bausteinen aus. Wie Sie vielleicht aus unserem Grafikkurs wissen, gibt es in diesen Bereichen drei Speicheretagen. Der Inhalt der Zelle 1 regelt den Zugriff des Computers auf die verschiedenen Speicherebenen. Werden die Bits 0 und 1 in Adresse 1 gelöscht, sieht der Computer nur noch das RAM. Basic- und Kernel-ROM sind verschwunden. Löschen Sie diese Bits deshalb nur durch ein Maschinenspracheprogramm, wenn vorher Ein- und Ausgaben gesperrt wurden. Dies wird durch Setzen des Interruptregisters erreicht.

Das Maschinenprogramm besteht aus drei Teilen. Im Bereich von 40704 bis 40768 erfolgt die Auswertung der Befehle von »RAM-Floppy« und der Aufruf der beiden Unterprogramme, die das Tauschen oder Verschieben der Basic-Programme erledigen.

Das abgedruckte Basic-Programm (Listing 12) POKEt das Maschinenprogramm ab Adresse 40704. Mit SYS 40704 wird es initialisiert.

Die »RAM-Floppy« hat eine Kapazität von 25 KByte. Das Programm im Basic-Speicher kann zwar 38 KByte lang sein, läßt sich dann allerdings nicht mehr vollständig verschieben oder vertauschen. Der Speicherbedarf sollte auch bei Programmen mit vielen Variablen nicht außer acht gelassen werden. Bei langen Programmen mit vielen Variablen kann es durchaus vorkommen, daß die Programme zwar getauscht, die Variablen allerdings nicht mehr übernommen werden können.

(Uwe Klatt)

Sequentielle Datei als Basic-Programm laden

Es sind eine Reihe von Anwendungen denkbar, bei denen aus einer sequentiellen Datei auf Diskette oder Kassette ein lauffähiges Basic-Programm erstellt werden soll (Datenfernübertragung, Umwandlung von Textfiles in Basic-Programme). Der folgende Einzeiler macht's für den VC 20 möglich:
OPEN 1, (Gerät), (Sekundäradresse), "(Name)" :
POKE 812, 238 : POKE 781, 1 : SYS 65478

Dieses Miniprogramm öffnet das File Nummer 1 als Eingabefile (anstelle der Tastatur). Außerdem wird der CLALL-Vektor des Betriebssystems auf ein »RTS« gesetzt, so daß beim Einlesen von Programmzeilen keine Files geschlossen werden.

Deshalb werden nach Eingabe der obigen Befehlszeile von der ausgewählten Datei so lange Zeilen eingelesen und anschließend sofort im Direktmodus ausgeführt, bis die Betriebssystem-Routine CLRCHN aufgerufen wird (zum Beispiel durch einen Syntaxfehler in den gelesenen Zeilen oder durch GET #1, A\$).

Um in den normalen Eingabemodus zurückzukehren, muß nur »POKE 812, 239 : CLR« eingegeben werden.

Zum Ausprobieren: Laden Sie ein beliebiges Basic-Programm und geben Sie danach im Direktmodus ein:

OPEN 1,8,3,"LISTING,S,W": CMD 1 : LIST : PRINT #1 :
CLOSE 1

```

0 REM***** <037>
1 REM* RAM-FLOPPY * <029>
2 REM***** <039>
3 REM* UWE KLATT * <042>
4 REM* BILLERBECKER STR. 27 * <148>
5 REM* 4939 STEINHEIM * <067>
6 REM* TEL. 05233/5672 * <106>
7 REM***** <044>
8 POKE 53280,0:POKE 53281,11 <021>
9 POKE 646,8 <072>
10 PRINT"BITTE WARTEN" <040>
11 REM ***** <023>
12 REM *** DATAS LESEN *** <020>
13 REM ***** <025>
14 FOR I=40704 TO 40768:READ A:POKE I,A:S= <100>
S+A:NEXT <181>
15 IF S<>6567 THEN END <181>
16 FOR I=40784 TO 40849:READ A:POKE I,A:S= <143>
S+A:NEXT <098>
17 IF S<>14392 THEN END <147>
18 FOR I=40853 TO 40902:READ A:POKE I,A:S= <169>
S+A:NEXT <032>
19 IF S<>20412 THEN END <197>
20 REM ***** <034>
21 REM *** MENUE *** <052>
22 REM ***** <194>
23 PRINT CHR$(147)
24 PRINT" (RVSON,40SPACE)";
25 PRINT" (RVSON,SPACE)RAM-FLOPPY 25.5 KBYT <048>
E (18SPACE)"; <246>
26 PRINT" (RVSON,40SPACE)"
27 PRINT"@V (2SPACE)VERSCHIEBT PROGRAMM I <254>
N(2SPACE)RAM-FLOPPY"
28 PRINT"@T (2SPACE)VERTAUSCHT PROGRAMM M <075>
IT RAM-FLOPPY"
29 REM ***** <041>
30 REM *** MC PROGRAMM STARTEN *** <222>
31 REM ***** <043>
32 SYS 40704:NEW <211>
33 REM ***** <045>
34 REM *** DATAS FUER 1. MC TEIL *** <097>
35 REM ***** <047>
36 DATA 169,159,133,56,133,52,169,0,133,55 <063>
,169,21,141,8,3,169
37 DATA 159,141,9,3,96,32,115,0,240,4,201, <111>
64,240,3,76,231
38 DATA 167,32,115,0,201,84,240,7,201,86,2 <255>
40,12,76,8,175,32
39 DATA 115,0,32,80,159,76,174,167,32,115, <127>
0,32,149,159,76,174
40 DATA 167 <008>
41 REM ***** <053>
42 REM *** DATAS FUER 2. MC TEIL *** <234>
43 REM ***** <055>
44 DATA 169,0,133,45,169,104,133,46,120,16 <051>
5,1,41,252,133,1,169
45 DATA 0,133,98,133,100,141,0,160,169,160 <246>
,133,101,169,8,133,99
46 DATA 162,96,160,0,177,98,133,102,177,10 <094>
0,145,98,165,102,145,100
47 DATA 200,208,241,230,99,230,101,202,208 <062>
,232,165,1,9,3,133,1
48 DATA 88,96 <112>
49 REM ***** <061>
50 REM *** DATAS FUER 3. MC TEIL *** <114>
51 REM ***** <063>
52 DATA 120,165,1,41,252,133,1,169,0,133,9 <069>
8,133,100,141,0,160
53 DATA 169,160,133,101,169,8,133,99,162,9 <197>
6,160,0,177,98,145,100
54 DATA 200,208,249,230,99,230,101,202,208 <197>
,240,165,1,9,3,133,1
55 DATA 88,96 <119>

```

Listing 12. Listing zu »RAM-Floppy«

Spezialeffekt

Wenn man beim C 64 in die Speicherstelle 53270 Werte zwischen 0 und 15 schreibt (POKE 53270,x), kann man den Bildschirm um bis zu sieben Bildpunkte nach links oder rechts scrollen lassen. Ist x kleiner als 8, dann scrollt der Bildschirmausschnitt um x Bildpunkte nach links, sonst um x-8 Bildpunkte nach rechts.

POKE 53270,8 stellt den Normalzustand wieder her.
Dieser Trick läßt sich gut bei Action-Spielen als optische Untermauerung beispielsweise einer Explosion einsetzen.
(Michael Keukert)

Text und Grafik mischen

Im Leserforum des 64'er-Magazins, Ausgabe 8/84, fragte Frank Schager nach einer Möglichkeit, mit Simons Basic ein Textfenster in der hochauflösenden Grafik zu erzeugen.

Mit dem folgenden kleinen Programm wird die normale Tastaturabfrage mit dem Simons Basic-Befehl »TEXT« verbunden:

```
10 X = 2 : Y = 2 : REM Text-Anfang
20 HIRES 15, 11 : REM Grafik ein
30 GET A$
40 IF A$ < > " " THEN GOSUB 100
50 GOTO 30
60 REM
100 X = X + 8 : REM X-Koordinate erhöhen
110 IF X > 38 * 8 THEN X = 2 : Y = Y + 8 :
REM Zeilenende? Dann neue Zeile
120 TEXT X,Y,A$,1,1,8 : REM Zeichen drucken
130 AA$ = AA$ + A$ : REM Wort erzeugen
140 IF AA$ = "GEHE" THEN 1000 : REM Zum Beispiel
150 RETURN
```

Veränderbar ist auch der Faktor 38 in Zeile 110, je nachdem, welche Zeilenlänge gewünscht wird. Ebenso besteht die Möglichkeit, zwischen den Zeilen 130 und 150 weitere IF-Abfragen einzubauen. Empfehlenswert ist die besondere Abfrage der Tasten SPACE und RETURN.

(Jörg Prante)

VC 20-Grundversion simuliert

Mit der folgenden kurzen Routine lassen sich die meisten Grundversions- oder +3 KByte-Programme auch mit einer 8 KByte-Erweiterung laden und ausführen:

```
POKE 648, 30 : SYS 64821
POKE 4096, 0 : POKE 44, 16 : NEW
```

Da der Bildschirmspeicher durch die kleine Routine an der gleichen Stelle wie in der Grundversion liegt, sind die meisten Grundversions-Programme direkt lauffähig.

Automatische Zeilennumerierung

Das lästige Durchnummerieren der Zeilen bei der Programmierung kann Ihnen dieses kleine Programm abnehmen.

Die Syntax des AUTO-Befehls ist:

—A anfangszeilennummer, schrittweite

Nach Eingabe dieses Befehls wird die Zeilennummer vorgegeben und nach RETURN um »schrittweite« erhöht.

Um aus dem AUTO-Modus wieder herauszukommen, muß man nach Vorgabe einer Zeilennummer

»—« RETURN eingeben.

Falls man nach Vorgabe einer Zeilennummer die RETURN-Taste betätigt, wird die entsprechende Zeile, falls sie vorhanden ist, gelöscht. Hiermit lassen sich auch sehr schnell Programmblöcke löschen, falls man die RETURN-Taste gedrückt hält, die Zeilenvorgabe weiterläuft und die entsprechenden Zeilennummern gelöscht werden.

»—«=CHR\$(95)

»A«=CHR\$(65)

Das Programm (Listing 13) als Basic-Lader eintippen, anschließend mit RUN starten. Falls »FEHLER IN DEN DATA-

ZEILEN« erscheint, DATAs auf Tippfehler überprüfen. Falls »OK«, kann die Basic-Erweiterung mit SYS 49152 initialisiert werden. Nun hat man das Basic um den Befehl »A« erweitert.

(Frank Siedel)

Zwei Tips für den C64

Die Speicherstellen 57 und 58 enthalten die Zeilennummer der aktuellen Basic-Zeile. Die Abfrage geschieht mit »PRINT PEEK(57) + 256 * PEEK(58)«.

Mit »PRINT PEEK(1)« kann abgefragt werden, ob eine Taste an der Datasette gedrückt ist. Es gibt drei mögliche Werte:

7: Taste gedrückt,

55: keine Taste gedrückt,

39: Taste gedrückt, aber Programmlauf unterbrochen.

Diese Abfragen sind für die benutzerfreundliche Programmierung von Kassettenoperationen recht nützlich.

(Wolfgang Meyer/kn)

Und noch ein Tip

Der FORMULAR TOO COMPLEX-Error ist sehr unangenehm, da sich das Programm danach oft nicht mehr listen läßt. Nach Eingabe von POKE 24,0 verhält sich der Computer aber wieder normal.

(Roger Limberg)

Tips und Tricks, die nur ein PEEK oder POKE beinhalten, finden Sie in der POKE-Liste. (kn)

```
1010 REM*****
1020 REM**      AUTO FUER C 64      **
1030 REM**      VON                  **
1040 REM**      FRANK SIEDEL         **
1050 REM**      POSENER STR. 18      **
1060 REM**      2945 SANDE           **
1070 REM*****
1080 :
1090 :
1100 :
1110 :DATA 169,11,141,8,3,169,192,141,9,3,
96,32,115,0,8,201,95,240,4,40,76,231
1120 :DATA 167,32,115,0,201,65,208,245,32,
115,0,24,32,107,169,165,20,133,38
1130 :DATA 165,21,133,39,32,253,174,24,32,
107,169,165,20,133,40,165,21,133,41
1140 :DATA 169,129,141,2,3,169,192,141,3,3,
169,128,141,138,2,165,39,133,98,165
1150 :DATA 38,133,99,162,144,56,32,73,188,
32,221,189,162,0,189,1,1,240,9,157
1160 :DATA 0,2,32,210,255,232,208,242,32,1
8,225,201,95,240,30,201,13,240,45
1170 :DATA 157,0,2,232,32,98,165,76,134,16
4,24,165,38,101,40,133,38,165,39,101
1180 :DATA 41,133,39,76,75,192,169,131,141
,2,3,169,164,141,3,3,169,0,141,138
1190 :DATA 2,40,76,116,164,32,118,165,76,1
34,164
1200 :
1210 :
1220 PRINT CHR$(147)
1230 SU=0
1240 FOR I=1 TO 170
1250 READ A
1260 SU=SU+A
1270 POKE 49151+I,A
1280 NEXT
1290 IF SU<>17417 THEN PRINT "FEHLER IN DE
N DATAZEILEN":END
1300 PRINT"OK":END
```

© 64'er

Listing 13.

Listing zu »Automatische Zeilennumerierung«