```
10 TD=56328:
      REM ECHTZEITUHR SEC/10-REGISTER      <012>
20 POKE TD+6,PEEK(TD+6)AND 127:
      REM 50 HZ EINSTELLEN                 <054>
30 DEF FN U4(X)=(X AND 15):
      REM UNTERE 4 BITS VON X              <186>
40 DEF FN O4(X)=(X AND 240)/16:
      REM OBERE 4 BITS                     <223>
50 DEF FN DC(X)=FN U4(X)+FN O4(X)*10:
      REM WERT BEI BCD-CODIERUNG           <001>
60 DEF FN DI(X)=FN DC(PEEK(X)AND 127):
      REM BCD-INHALT VON X OHNE BIT 7      <074>
70 DEF FN H(X)=FN DI(TD+3):
      REM STUNDEN                          <245>
80 DEF FN M(X)=FN DI(TD+2):
      REM MINUTEN                          <093>
90 DEF FN S(X)=FN DI(TD+1):
      REM SEKUNDEN                         <044>
100 DEF FN DH(X)=X-INT(X/10)*10+INT(X/10)*
    16: REM DEZIMAL NACH BCD               <174>
110 INPUT"UHRZEIT HH,MM,SS";HH,MM,SS       <154>
120 IF HH>12 THEN HH=HH-12                 <053>
130 POKE TD+3,FN DH(HH):REM UHR STELLEN    <177>
140 POKE TD+2,FN DH(MM)                    <029>
150 POKE TD+1,FN DH(SS)                    <009>
160 POKE TD+0,0                            <139>
170 PRINT"{CLR}"                           <158>
180 PRINT"{HOME}"FN H(X)"{LEFT}"FN M(X)"{L
    EFT}"FN S(X)"{LEFT}"PEEK(TD)"{DEL}";   <244>
190 GOTO 180                               <246>
```

**Listing 5. Eine sinnvolle Anwendung der FN-Anweisung**

## Der Super-Autostart

Darauf haben Sie schon lange gewartet: Einen Autostart-Generator, der viele sinnvolle Eigenschaften aufweist. Dazu gehören: Kurzes Listing (sowohl des Generator-Programms als auch des Autostarts selber), einfach in der Anwendung, RUN/STOP-RESTORE- und Reset-Schutz für das fertige Programm und eine eingebaute Codier- und Decodier-Funktion.

Der Autostart-Generator in Listing 6 hat alle genannten Funktionen. Die Anwendung ist äußerst einfach: Abtippen, speichern, absolut laden, »NEW« eintippen. Dann das zu bearbeitende (Basic-)Programm laden und den Autostart mit folgender Zeile aktivieren:

```
SYS 49152,Code,"Haupt-Name","Lader-Name"
```

»Code« ist eine beliebige Zahl zwischen 0 und 255. »Haupt-Name« und »Lader-Name« sind die zukünftigen Namen des

```
programm : autostart        c000 c131
-----------------------------------

c000 : 20 fd ae 20 9e b7 8e ac  0a
c008 : c0 20 fd ae 20 9e ad 20  1b
c010 : a3 b6 8d 6f c0 c9 0d 90  0f
c018 : 03 4c 71 a5 20 bd ff 20  83
c020 : 11 c1 a2 08 86 ba 20 35  e5
c028 : c0 a9 2b a6 2d a4 2e 20  4d
c030 : d8 ff 4c bf c0 a5 2b 85  04
c038 : fb a5 2c 85 fc a0 00 b1  fa
c040 : fb 4d ac c0 91 fb c8 d0  e3
c048 : f6 a5 fc e6 fc c5 2e d0  85
c050 : ee 60 a2 ea 8e 28 03 bd  26
c058 : 77 02 4d 00 03 9d 80 7f  42
c060 : ca 30 f4 a2 04 bd 10 fd  3e
c068 : 9d 04 80 ca 10 f7 a9 0c  00
c070 : a2 0d a0 80 20 bd ff a9  14
c078 : 00 85 9d 20 d5 ff 86 2d  78
c080 : 98 a6 2b 86 fb a4 2c 84  a6
c088 : fc 20 57 a6 a8 b1 fb 4d  e2
c090 : 00 03 91 fb c8 d0 f6 a5  30
c098 : 2e e7 fc 10 f0 20 53 e4  22
c0a0 : 4c ae a7 e2 fc 5e fe 43  cf
c0a8 : 48 44 38 36 00 45 a6 02  b0
c0b0 : 28 43 29 38 36 20 42 59  eb
c0b8 : 20 43 48 44 4c 79 00 20  e5
c0c0 : 35 c0 20 79 00 c9 2c d0  2d
c0c8 : f3 20 73 00 20 9e ad 20  96
c0d0 : a3 b6 c9 00 d0 05 a2 08  11
c0d8 : 4c 37 a4 20 bd ff a2 52  f8
c0e0 : a0 c0 86 ac 84 ad a2 bc  d1
c0e8 : a0 c0 86 ae 84 af a9 61  8f
c0f0 : 85 b9 20 d5 f3 20 8f f6  81
c0f8 : a9 08 20 b1 ff a9 61 20  f7
c100 : 93 ff a9 a6 20 dd ed a9  ce
c108 : 02 20 dd ed a0 00 4c 24  d3
c110 : f6 86 fb 84 fc a8 88 b1  73
c118 : fb 4d ac c0 99 b0 c0 88  30
c120 : 10 f5 a0 03 b9 a3 c0 4d  0a
c128 : ac c0 99 a3 c0 88 10 f4  8a
c130 : 60 ff ff ff ff ff ff ff  90
```

**Listing 6. Der Super-Autostart, den Sie schon immer suchten**

codierten Hauptteils beziehungsweise des Lade-Programms auf der Diskette. Der Lader ist später mit »,8,1« in den C 64 zu lesen.

Läßt man »,"Lader-Name"« weg, so wird nur der codierte Hauptteil neu gespeichert (wenn man Änderungen am Hauptteil vorgenommen hat). Wichtig ist dann nur, daß die Code-Zahl des Hauptprogramms mit der des Laders übereinstimmt. Im Zweifelsfall sollte man lieber den alten Lader löschen und beide Teile neu generieren.

(Christoph Dautzenberg/tr)

# Tips & Tricks zum C 128

**Weiter geht's mit den Tips und Tricks. Diesmal dabei: Variablendump, ein FIND-Befehl, C 64-Modus im 2-MHz-Takt, ein Trick zur 1571 und vieles mehr.**

Tips und Tricks nutzen jedem Anwender. Wie könnte man seine Maschine sonst ausnutzen, wenn man nicht die kleinen Kniffe wüßte, die dem Computer die Feinheiten entlocken. Lesen Sie hier, wie Sie Ihren C 128 bändigen.

## Cursor auch bei GET

Mit dieser kleinen Zeile, die als Erste in einem Programm stehen muß, können Sie beim C 128 auf dem 80-Zeichen-Bildschirm den Cursor bei GET blinken lassen:

```
0 PRINT CHR$(27)+"U"
```

(Thomas Tschink/dm)

## Komfortable Joystick-Abfrage

Dieses Programm ermöglicht eine komfortable Joystickabfrage. Statt umständlichen IF..THEN-Anweisungen werden einfach indizierte Variablen benutzt.

**Zum Programm**

In den Zeilen 40 und 50 werden die Variablen A(X) und B(X) definiert. Der Index X entspricht den Richtungen 1 bis 8 des Joysticks.

In Zeile 60 wird die Variable J mit dem Zustand des Joysticks in Port 1 belegt. Ist der Feuerknopf gedrückt (Richtung + 128), springt das Programm wieder zu Zeile 60 (es würde sonst zu einer Fehlermeldung kommen).

In Zeile 70 werden jetzt die Werte der Variablen A(J) und B(J) zu der X- und Y-Position dazuaddiert. Dabei kommt uns ein Gesetz der Mathematik zugute: Addiert man zu einer positiven eine negative Zahl, kommt dies einer Subtraktion gleich (zum Beispiel: 100 + (-20) = 80).

Mit X und Y können jetzt Sprites, Shapes oder ähnliches bewegt werden (hier ist es ein Sprite).

**Ein Beispiel für die Arbeitsweise**

Wird der Joystick nach Rechts-Oben bewegt, ist J = 2; also ist A(J) = 1 und B(J) = -1. Angenommen, X = 100 und Y = 90, so ist X + A(J) = 101 ((100 + 1 = 101)) und Y + B(J) = 89 ((90 + (-1) = 89)).

Es ist natürlich möglich, auch andere Schrittweiten als 1 zu nehmen, vorausgesetzt, X und Y sind nicht größer oder kleiner als die zugelassenen Werte.

```
10 X=160:Y=100
20 SPRDEF:SPRITE 1,1,2
30 COLOR 0,1:COLOR 4,1
40 A(1)=0:B(1)=-1:A(2)=1:B(2)=-1:A(3)=1
50 B(3)=0:A(4)=1:B(4)=1:A(5)=0:B(5)=1
60 A(6)=-1:B(6)=1:A(7)=-1:B(7)=0
70 A(8)=-1:B(8)=-1
80 J=JOY(2):IF J>127 THEN J=J-128
90 X=X+A(J):Y=Y+B(J)
100 MOVSPR 1,X,Y:GOTO 80
```

(Thorsten Wanschura/dm)

## Eine Grafik-Routine

Mit dem kleinen Programm

```
10 GRAPHIC 1,1:COLOR 4,15
20 COLOR 1,2:Color 0,8
30 DO UNTIL E=180
40 E=E+4
50 CIRCLE 1,159,99,E,,,,E,72
60 LOOP
```

läßt sich mit wenig Aufwand eine schöne Grafik berechnen, die auf dem HiRes-Schirm dargestellt wird.

(Jürgen Strehle/dm)

```
63000 FAST:BANK 1:FORI=PEEK(47)+PEEK(48)
*256TOPEEK(49)+PEEK(50)*256-1STEP7
63010 FORII=0TO6:S(II)=PEEK(I+II):NEXT
63020 IF S(0)>128 AND S(1)<128 THEN BEGI
N:REM DEF FN-VARIABLE
63030 PRINT"DEF FN";CHR$(S(0)AND127);CHR
$(S(1)AND127)
63040 BEND
63050 IF S(0)>128 AND S(1)>=128 THEN BEG
IN:REM INTEGER-VAR
63060 PRINTCHR$(S(0)AND127);CHR$(S(1)AND
127);"%";TAB(4);"=";
63070 S(4)=S(3)+S(2)*256:IFS(4)>32767 TH
EN S(4)=S(4)-65536
63080 PRINTS(4):BEND
63090 IF S(0)<128 AND S(1)<128 THEN BEGI
N:REM REAL
63100 PRINTCHR$(S(0)AND127);CHR$(S(1)AND
127);TAB(4);"=";
63110 S(2)=S(2)-129
63120 IFS(3)>127THEN S(3)=S(3)-128:MA=-1
:ELSE MA=1
63130 S(3)=S(3)/128
63140 S(4)=S(4)/128/256
63150 S(5)=S(5)/128/2562
63160 S(6)=S(6)/128/2563
63170 S(0)=MA*2S(2)*(1+S(3)+S(4)+S(5)+S(
6))
63180 PRINTS(0)
63190 BEND
63200 IF S(0)<128 AND S(1)>=128 THEN BEG
IN:REM STRING-VAR
63210 PRINTCHR$(S(0)AND127);CHR$(S(1)AND
127)"$";TAB(4);"= ";
63220 PRINTCHR$(34);:S(0)=S(3)+S(4)*256
63230 IFS(2)<>0THEN BEGIN
63240 FOR O=0TOS(2)-1:PRINTCHR$(PEEK(S(0
)+O));:NEXT:PRINTCHR$(34)
63250 BEND:ELSE PRINTCHR$(34)
63260 BEND
63270 NEXT
63280 END
```

**Listing 1. »UNDIM.VAR.DUMP«**

## Zwei nützliche Hilfsroutinen

Das Programm UNDIM.VAR.DUMP (Listing 1) gibt auf dem aktuellen Ausgabegerät die nicht DIMensionierten Variablen aus. Der Aufruf erfolgt mit GOTO 63000.

Das Programm F.KEY-DISPLAY (Listing 2) stellt vier zusätzliche Bildschirmzeilen zur Verfügung, in denen die beim Aufruf gültige Belegung der Funktionstasten angezeigt werden. Es ist nötig, vor dem Start den 80-Zeichen-Modus eingeschaltet zu haben.

(Christian Klein/dm)

```
100 FAST:COLOR6,16:COLOR5,1:PRINT"";:BAN
K15:V1=54784:V2=V1+1
110 POKEV1,20:POKEV2,16
120 POKE 2607,16
130 POKEV1,06:POKEV2,29
140 POKEV1,07:POKEV2,34
150 GOSUB420
170 O=49:F=4096:T=4106:FORI=2000TO 2319
180 A=I  /  256
190 X=I AND 255
200 Y=6
210 :SYS 3340,A,X,Y
220 I=I+1
230 A=I  /  256
240 X=I AND 255
250 Y=0
260 :SYS 3340,A,X,Y
270 I=I+1
280 A=I  /  256
290 X=I AND 255
300 Y=61
310 :SYS 3340,A,X,Y
320 I=I+1
330 FT=PEEK(F+O-49):REM LAENGE DES STRIN
GS DER FUNKTIONSTASTE
340 FORII=ITOI+38
350 A=II /  256
360 X=IIAND 255
370 : IF II>=FT+I THEN Y=32:ELSE IFPEEK(
T+II-I)<>13THEN Y=PEEK(T+II-I)AND63:ELSE
Y=31
380 :SYS 3340,A,X,Y
390 NEXT:O=O+1:I=2000+(O-49)*40:T=T+FT
410 I=I-1:NEXT:GOTO600
420 REM
430 FOR I=0 TO 45
440 : READ X
450 : POKE 3328+I,X
470 NEXT
500 DATA 142,0,214,44,0,214,16,251,141,1
,214,96,141,24,13,142,30,13,140,37,13
510 DATA 162,18,169,0,32,0,13,232,169,0,
32,0,13,162,31,169,32,32,0,13
520 DATA 162,18,76,0,13
530 :
540 RETURN
600 FORI=IITO2319
610 A=I  /  256
620 X=I AND 255
640 Y=32
650 : SYS 3340,A,X,Y
660 NEXT
700 REM ATTRIBUTRAM VERAENDERN
710 FORI=6096TO6096+319
720 A=I  /  256
730 X=I AND 255
735 Y=66 +  128
740 : SYS 3340,A,X,Y
745 NEXT:END
800 FORI=2000 TO2319:SYS 3340,IAND255,I/
256,32:NEXT
```

**Listing 2. »F.KEY-DISPLAY«**

## Auch im C 64-Modus mit 2 MHz

Laut Auskunft bei Commodore ist es möglich, den C 128 im C 64-Modus mit 2 MHz arbeiten zu lassen. Diese wenigen Programmzeilen wirken wie FAST beim C 128 im 40-Zeichen-Modus.

Das bedeutet, der Computer arbeitet mit 2 MHz. Jedoch ist es nicht möglich, im C 64-Modus über die RGBI-Buchse zu gehen und somit die Arbeit am Bildschirm zu verfolgen.

Diese POKEs eignen sich also besonders für die Leute, die ihre eigenen Basic V2-Programme zwar nicht umschreiben, aber doch die volle Geschwindigkeit des C 128 nutzen wollen. Sie können sie als kleine Routine vor den Rechenteil Ihres Programmes setzen.

```
10 POKE 53265,PEEK (53265) AND 239: REM Video aus
20 POKE 53296,PEEK (53296) OR 1: REM 2 MHz an
30 FOR A = 1 TO 1000:NEXT A: REM Programmteil
40 POKE 53296,PEEK (53296) AND 254: REM 1 MHz an
50 POKE 53265,PEEK (53265) OR 16: REM Video an
```

(Sanjiv Singh/dm)

## Ein FIND-Befehl

Obwohl das Basic 7.0 des Commodore 128 viele neue Befehle beinhaltet, gibt es doch einige, die man vermißt. Ein solcher ist der von vielen Erweiterungen für den C 64 her bekannte Befehl FIND. Der hier abgedruckte Basic-Lader (Listing 3) bringt dem C 128 eine solche Routine bei.

```
10 REM     FIND  FUER PC 128
20 REM     ****
30 REM
40 REM           HERBERT KUNZ
50 REM
60 REM           12/1985
70 REM
100 DATA A9,0B,8D,08,03,A9,13,8D
101 DATA 09,03,60,20,80,03,C9,40
102 DATA F0,06,20,86,03,4C,F3,4A
103 DATA A9,00,85,FA,A2,00,20,80
104 DATA 03,F0,EF,20,29,13,4C,F6
105 DATA 4A,C9,22,D0,07,A9,01,85
106 DATA FA,4C,43,13,9D,00,0C,E8
107 DATA 20,80,03,D0,F7,9D,00,0C
108 DATA 4C,5C,13,20,80,03,C9,22
109 DATA F0,06,9D,00,0C,E8,D0,F3
110 DATA A9,00,9D,00,0C,20,80,03
111 DATA A9,01,85,FA,A5,2D,85,FC
112 DATA A5,2E,85,FD,A9,00,85,FB
113 DATA A0,01,20,DF,13,F0,9B,A0
114 DATA 04,A2,00,20,DF,13,48,C9
115 DATA 22,D0,06,A9,01,45,FB,85
116 DATA FB,68,F0,46,48,A5,FA,C5
117 DATA FB,F0,05,68,C8,4C,71,13
118 DATA 68,DD,00,0C,F0,03,C8,D0
119 DATA DA,C8,E8,BD,00,0C,F0,0E
120 DATA 48,20,DF,13,85,63,68,C5
121 DATA 63,F0,EE,4C,71,13,A9,0D
122 DATA 20,0C,56,A5,FC,85,61,A5
123 DATA FD,85,62,A0,03,20,DF,13
124 DATA 48,88,20,DF,13,AA,68,20
125 DATA 23,51,A0,00,20,DF,13,AA
126 DATA C8,20,DF,13,85,FD,86,FC
127 DATA A5,91,C9,7F,D0,86,60,A9
128 DATA FC,4C,9F,03
129 S=0:FORI=4864TO5091:READA$:A=DEC(A$)
130 POKEI,A:S=S+A:NEXT
131 PRINT"DIE DATAZEILEN SIND ";
132 IFS<>25906THENPRINT"FEHLERHAFT":STOP
133 PRINT"IN ORDNUNG":SYSDEC("1300")
```

**Listing 3. »FIND«**

Aktiviert wird die Routine mit SYS DEC("1300"). Nun können Sie Ihr Basic-Programm nach einem Begriff durchsuchen. Als Befehlserkennung dient der Klammeraffe @. Danach schreiben Sie das Suchwort (zum Beispiel einen Befehl, einen String oder einen Variablennamen). Wenn eine Zeile gefunden wird, in der der Begriff auftaucht, so wird die komplette Zeile auf dem Bildschirm gelistet (mit OPEN 1,4:CMD 1 auch auf dem Drucker).

Das Listen kann mit der Commodore-Taste verlangsamt und mit der NO SCROLL-Taste aufgehalten werden. Wenn der Klammeraffe in einer Zeile (Bildschirmzeile, nicht Programmzeile) steht, so ist davor ein Doppelpunkt einzugeben.

Hier noch einige Beispiele:

: @DATA — listet alle DATA-Zeilen

: @GOTO 10 — listet alle Zeilen, in denen GOTO 10 vorkommt, aber auch GOTO 100, GOTO 1000 etc.

: @"MESSUNG" — alle Zeilen, in denen der String gefunden wird, werden gelistet. (Herbert Kunz/dm)

## Programme transferieren mit FLASHMOVE

Spielen Sie folgenden Gedankengang einmal durch: Sie besitzen einen C 128 und eine Floppy 1570/1571. Diese Gerätekombination besitzt die gute Eigenschaft, daß im C 128-Modus lange Programme in wenigen Sekunden geladen werden können. Sie möchten aber Ihre C 64-Software aktiv weiterbenutzen. Also nichts wie GO 64 eingeben und das entsprechende Programm laden...

Man freut sich immer wieder, daß ein Programm im C 64-Modus nur so »reinfetzt«, was gewöhnlich bei längeren Programmen zu erfreulich »langen« Wartezeiten führt. Es muß also eine Programmroutine her, die es ermöglicht, ein im C 128-Modus geladenes Programm in den C 64-Modus umzuwandeln, so daß man es dort wie gewohnt weiterverwenden kann. Die Lösung: »Flashmove«.

Da Bank 0 sowohl im C 128- als auch im C 64-Modus für die Speicherung von Basic-Programmen dient, der Basic-Anfang aber verschoben ist (C 128: $1C00, C 64: $0800), liegt es nahe, daß ein im C 128-Modus geladenes Programm nur noch verschoben werden muß, um es im C 64-Modus zu gebrauchen. Und eben das erledigt das Programm »Flashmove« (Listing 4).

Der einzige Nachteil der Sache ist, daß das Programm nicht länger als 45 KByte sein darf. Übrigens, wenn die Startadresse eines Programmes über $1C00 liegt, benötigt man Flashmove nicht.

Bank 0 wird durch GO 64 nur in die C 64-Speicherverwaltung umgeschaltet, aber nicht total gelöscht.

Und so benutzt man Flashmove im C 128-Modus:

1) »Flashmove« (wird als % gespeichert) mit ,8,1 laden
2) LOAD "beliebiges Programm" ,8
3) GO 64
4) »Flashmove« mit SYS 2060 starten, fertig

**Eingabehinweise**

Um den Maschinencode von Flashmove zu erhalten, geht man folgendermaßen vor:

— Zuerst geben Sie bitte das Listing »Flashmove« (Listing 4) ein und speichern es

— Nach dem Start durch RUN wird auf zwei Bildschirmseiten die Funktion und Handhabung nochmals beschrieben

— Schließlich wird man nach dem Namen gefragt, unter welchem Flashmove gespeichert werden soll (ACHTUNG: ist schon ein File mit gleichen Namen auf Diskette vorhanden, so wird dieses gelöscht)

Mit dem »Autoboot Maker« (zu finden auf der 1570/1571 Test/Demo-Disk) kann man nun den abgespeicherten Maschinencode bootfähig machen:

— »Autoboot Maker« laden und starten

— Den Namen eingeben, unter dem das Maschinenprogramm von Flashmove gespeichert wurde

— Die Frage nach dem Datentyp beantwortet man mit »binary« (Stefan Seidenberg/dm)

```
10 printchr$(14):scnclr:color4,1:color0,1
23 print:printtab(7)"Fuer PC-128 und 1570/1571":po
ke162,0:wait162,128
24 fast:scnclr
25 print"PC-128"tab(10)chr$(18)" Flashmove V2 "chr
$(146)"    1570/1571
26 print:print"Flashmove verschiebt Programme vom
27 print"PC-128er Basicanfang an den C-64er
28 print"Basicanfang. So kann man im 128er Modus
29 print"Programme mit der 1570/1571 in wenigen
30 print"Sekunden laden und nach Aufruf dieses
31 print"Programmes im 64er Modus ohne Probleme
32 print"weiterverwenden.
33 print"Die Programme duerfen bis 45k lang sein!
34 print:print:print"Die Ihnen nun vorliegende 2.
Version
35 print "von 'Flashmove' ist erweitert worden:
36 print:print"1. Nach dem Aufruf von 'Flashmove V
2'
37 print"   erscheint 'Run' auf dem Bildschirm.
38 print"   Will man das verschobene Programm
39 print"   jetzt starten, so drueckt man einfach
40 print"   'RETURN'.
41 print: print"2. 'Flashmove V2' ist bootfaehig!!
42 gosub73
43 fast:scnclr:print"Dieses Programm speichert 'Fl
ashmove V2'
44 print"als Maschinencode ab. Dieser kann sowohl
45 print"manuell geladen, als auch als Autoboot-
46 print"Programm generiert werden. Dazu benutzt
47 print"man den auf der PC-128 Test/Demo
48 print"Diskette befindlichen 'Autoboot Maker'.
49 print:print"'Flashmove V2' benutzt man wie folg
t:
50 print"Der PC-128 befindet sich im 128er-Modus.
51 print"Man gibt eine der folgenden Befehles-
52 print"sequenzen ein:
53 print:print"1. Load'Flash*',8,1  1. Load'Name',
8
54 print"2. Load'Name' ,8     2. go 64
55 print"3. go64                3. Load'Flash*',8,1
56 print"4. sys 2060             4. sys 2060
57 print:print"1. Boot                  1. load'Name',
8
58 print"2. Load'Name',8       2. Boot
59 print"3. go 64               3. go 64
60 print"4. sys 2060            4. sys 2060
61 print"-> Nun kann das Programm benutzt werden.
62 gosub73
63 scnclr:print"Bitte legen Sie nun eine formatier
te
64 print"Diskette in das Laufwerk!
65 print"Der Maschinencode von 'Flashmove V2'
66 print"wird abgespeichert.
67 print:poke208,0:input"Filename:";f$
68 scratch (f$)
69 open1,8,1,f$:print#1,chr$(12)chr$(8);
70 reada:ifa<>-1 then print#1,chr$(a);:x=x+a:goto7
0
71 close1:ifx<>13269 then print:print"Pruefsummenf
ehler!!
72 print:print"Diskstatus: "ds$:end
73 slow:char1,2,24,"(c) 1986  by Stefan Seidenberg
":poke208,0
74 if peek(208)<>0 then return
75 char 1,0,23,"      ":poke162,0:wait162,32:char 1
,0,23,"Taste"+chr$(7)
76 poke162,0:wait162,32:goto74
77 data165,44,201,8,208,101,162,0,189,32,8,157,0,4
,232,16,247,76,0,4,169,1
78 data141,32,208,173,17,18,133,2,56,233,20,133,46
,173,16,18,133,45,120,169
79 data18,133,1,169,0,133,251,133,253,169,28,133,
252,169,8,133,254,160,0
80 data177,251,145,253,200,208,249,165,252,197,2,2
40,6,230,252,230,254,208
81 data235,169,119,133,1,88,169,0,141,32,208,169,8
2,141,119,2,169,213,141
82 data120,2,169,2,133,198,76,51,165,96,-1
```

**Listing 4. »FLASHMOVE«**

## Sprites invertieren

Dieses kleine Programm (Listing 5) (re)invertiert Sprites, die im Speicher abgelegt sind. Der Aufruf der Routine ist sehr einfach. Die Syntax lautet:
SYS anfadr, sprnr
— anfadr steht für die Startadresse des Maschinenprogrammes im Speicher (zum Beispiel $0b00 bzw. dez. 2816)
— sprnr steht für die Nummer des zu (re)invertierenden Sprites. Ist die Spritenummer größer 8 oder kleiner gleich 0, erfolgt ein ILLEGAL QUANTITY ERROR, auch im Programmmodus. Die fehlerhafte Zeile läßt sich dann mit HELP listen.

Das Programm ist an keine bestimmte Adresse gebunden und kann durch Ändern der Startadresse oder dem T(ranslate)-Befehl des Monitors an eine andere Adresse verschoben werden.

Die Vorzüge der Routine liegen vor allem in der Geschwindigkeit, in der diese ausgeführt wird. Außerdem geht kein Basic-Speicher verloren. Für Assembler-Fans ist in Listing 6 der kommentierte Quellcode abgedruckt. (Udo Miller/dm)

```
100 rem ******************************
110 rem *         sprinv 1.0           *
120 rem *                              *
130 rem *       1986 by udo miller     *
140 rem ******************************
150 :
160 color0,1:color4,1:scnclr
170 print"startadresse (z.b. dez. 2816)";
180 inputad
190 :
200 do
210 read a$:ifa$="ende"thenexit
220 pokead+i,dec(a$)
230 i=i+1
240 loop
250 :
260 data c9,00,f0,06,c9,08,90,07,f0,05,a2
270 data 0e,6c,00,03,aa,ca,a9,00,86,fc,a0
280 data 00,84,fa,a0,0e,84,fb,e0,03,f0,0b
290 data 90,09,e6,fb,a9,fb,65,fc,aa,86,fc
300 data a0,00,a9,00,c4,fc,f0,07,c8,69,40
310 data c4,fc,d0,f9,85,fa,a0,00,b1,fa,49
320 data ff,91,fa,c8,c0,40,d0,f5,60
330 data ende
```

**Listing 5. »Sprinv 1.0«**

## Noch ein kleiner Trick

Der C 128 bietet ja eine variable Funktionstastenbelegung mit dem Befehl KEY. Will man eine selbsterstellte Belegung auf Diskette speichern, so kann man folgende Befehlsfolge benutzen: `BSAVE"FUNKTIONS-T.",D0,U8,ON B0,P4096 TO P4352`

Damit wird der Bereich der Funktionstasten (4096 bis 4352) gespeichert. Geladen wird er mit: `BLOAD"FUNKTIONS-T."`

(Holger Brömmelsiek/dm)

```
00b00  c9 00     cmp #$00      ;Akku(Spritenummer)= 0, wenn ja
00b02  f0 06     beq $0b0a     ;dann Fehlermeldung
00b04  c9 08     cmp #$08      ;Akku(Spritenummer)<=8, dann
00b06  90 07     bcc $0b0f     ;weiter ab $0b0f
00b08  f0 05     beq $0b0f     ;sonst
00b0a  a2 0e     ldx #$0e      ;Fehlermeldung ($0e)
00b0c  6c 00 03  jmp ($0300)   ;illegal quantity und STOP!!
                               ;Beginn Hauptprogramm
00b0f  aa        tax           ;Zuweisung des Akku ins X-Register
00b10  ca        dex           ;und um 1 verringern
00b11  a9 00     lda #$00      ;Hilfszeiger mit
00b13  86 fc     stx $fc       ;der Basisadresse des
00b15  a0 00     ldy #$00      ;Spritedatenspeichers laden
00b17  84 fa     sty $fa       ;und sichern
00b19  a0 0e     ldy #$0e      ;
00b1b  84 fb     sty $fb       ;
00b1d  e0 03     cpx #$03      ;wenn X-Register <=3, dann
00b1f  f0 0b     beq $0b2c     ;Sprung nach $0b02,
00b21  90 09     bcc $0b2c     ;andernfalls
00b23  e6 fb     inc $fb       ;HI-Byte(Zeiger) um 1 erhöhen,
00b25  a9 fb     lda #$fb      ;Akku mit #$fb laden  und
00b27  65 fc     adc $fc       ;in Zelle $fc addieren
00b29  aa        tax           ;Zuweisung des Akku ins X-Register
00b2a  86 fc     stx $fc       ;und sichern
00b2c  a0 00     ldy #$00      ;Y-Register und
00b2e  a9 00     lda #$00      ;Akku löschen
00b30  c4 fc     cpy $fc       ;wenn Y-Register = Zelle $fc
00b32  f0 07     beq $0b3b     ;weiter ab $0b3b, sonst
00b34  c8        iny           ;Y-Register erhöhen und
00b35  69 40     adc #$40      ;#$40 zum Akku addieren
00b37  c4 fc     cpy $fc       ;wenn Y-Register <> Zelle $fc,
00b39  d0 f9     bne $0b34     ;dann wieder zu $0b34
00b3b  85 fa     sta $fa       ;LOW-Byte(Zeiger) sichern
00b3d  a0 00     ldy #$00      ;Y-Register löschen
00b3f  b1 fa     lda ($fa),y   ;Akku durch Hilgszeiger laden,
00b41  49 ff     eor #$ff      ;invertieren und
00b43  91 fa     sta ($fa),y   ;zurückschreiben
00b45  c8        iny           ;Y-Register erhöhen und
00b46  c0 40     cpy #$40      ;mit #$40 vergleichen,
00b48  d0 f5     bne $0b3f     ;wenn <> dann zu $0b3f
00b4a  60        rts           ;ENDE
```

**Listing 6. Der Quellcode zu Listing 5**