

Tips & Tricks für Profis

Auch diesmal haben wir für die Profis unter unseren Lesern einige interessante Hinweise. Beachtenswert ist unter anderem »das geheimnisvolle, reverse SHIFT-M«.

Schalten Sie den Reverse-Modus ein und drücken Sie dann die Tastenkombination SHIFT-M. Was dieser entstandene reverse Schrägbalken alles bewirken kann, möchte ich kurz schildern:

1. Innerhalb von REM-Zeilen

Geben Sie eine Zeilennummer und REM "" ein. Nun löschen Sie das zweite Hochkomma mit der DEL-Taste, schalten den Reverse-Modus ein und drücken die Tastenkombination SHIFT-M. Schalten Sie den Reverse-Modus wieder aus und setzen abermals ein Hochkomma, das Sie anschließend mit der DEL-Taste wieder löschen. Nun können Sie wie innerhalb eines PRINT-Befehls einen Text mit Farbcodes etc. eingeben. Nach der Eingabe drücken Sie die RETURN-Taste und listen das Programm. Sie werden erstaunt sein. Zur Veranschaulichung soll folgende Zeile gelten (die Kommata nicht mit eintippen):

```
10 REM " " {DEL, RVSON, SHIFT M, RVSOFF} " {DEL, SHIFT CLR/HOME, WHITE, RVSON} This Program is protected {RVSOFF, DARK BLUE} "
```

2. Im Directory

Geben Sie »SAVE« ein und erzeugen Sie den reversen Schrägbalken wie oben beschrieben. Dann tippen Sie den Programmnamen mit beliebig vielen Steuerzeichen (insgesamt 16 Zeichen) ein und schließen die Eingabe mit ",8 und RETURN ab. Auch hier zur Veranschaulichung wieder ein Beispiel:

```
SAVE " " {DEL, RVSON, SHIFT M, RVSOFF} " {DEL, RVSON} TEST {RVSOFF} ",8
```

Alle Codes werden ebenfalls aktiviert, wenn Sie eingeben: SAVE CHR\$(34) + {RVSON} Test {RVSOFF} ",8

3. Als Programmschutz

Wenn Sie am Anfang einer Programmzeile PRINT" (reverser Schrägbalken) und danach weitere Basic-Befehle eingeben, können diese nach Drücken der RETURN-Taste nicht mehr geändert werden (nur durch Zeile löschen oder mit einem Monitor). Beim Listen des Programms werden die nachfolgenden Befehle um eine oder mehrere Zeilen nach unten gerückt (je nach dem, wieviele reverse Schrägbalken Sie in dem PRINT-Befehl verwendet haben), 80 Zeichen pro Zeile werden meistens überschritten. Will man diese Zeile durch Return abermals speichern, so wird nur der PRINT-Befehl gespeichert, alle anderen nachfolgenden Befehle gehen verloren. Ein Beispiel:

```
10 PRINT " " {DEL, RVSON, SHIFT M, RVSOFF} " :PRINT "COPYRIGHT BY 64'ER-REDAKTION"
```

Nachdem Sie diese Zeile eingetippt haben und RETURN gedrückt haben, versuchen Sie einmal, den in den Hochkomma des zweiten PRINT-Befehls stehenden Text auszubessern. Nach dem Ausbessern und RETURN-Drücken tritt entweder ein Syntax Error auf und die Zeile bleibt unverändert, oder in der Zeile 10 steht nur mehr PRINT".

4. Als Listschutz

Geben Sie das erste Beispiel ein, fügen Sie nach dem letzten Hochkomma noch das Zeichen SHIFT L hinzu, drücken Sie RETURN. Nun tippen Sie im Direktmodus POKE 2052,255 ein und speichern das Programm. Nach dem Laden ist der Listschutz noch immer vorhanden.

Durch diesen POKE bekommt die erste Zeile in einem Programm eine unzulässige Zeilennummer (größer als 63999). Sie kann daher nicht mehr gelöscht werden. Das Listen des Programms wird bei der ersten Zeile abgebrochen, Befehle wie LIST 20- (oder ähnliches) werden auch nicht ausgeführt, da der Computer glaubt, die erste Zeile sei wegen der hohen Zeilennummer die letzte. Der Listschutz kann jedoch mit POKE 2052,0 wieder aufgehoben werden.

(Gert Harlacher/tr)

Versteckspiel mit dem Directory

Wenn eine Diskette mit "N:NAME" + CHR\$(0) + CHR\$(0) + CHR\$(0) + ", ID" formatiert wird, so wird beim Laden und Auflisten des Directories nur noch der Name der Diskette angezeigt. Der Interpreter »denkt« nämlich, daß das Directory nach den drei Nullen zu Ende ist. Die Programme auf dieser Diskette lassen sich aber trotzdem noch laden.

Ein weiterer Trick besteht darin, beim Speichern eines Programms an den Namen ein »,S,W« anzuhängen (also zum Beispiel SAVE "TEST, S, W", 8).

Im Directory steht dieses File nun unter »SEQ« vermerkt. Wer versucht, dieses Programm »normal« mit LOAD zu laden, der wird von der Floppy die Fehlermeldung »64, FILE TYPE MISMATCH« erhalten.

Der Kundige jedoch wird das Programm mit »LOAD "NAME, S,R",8« laden, und sich des gelungenen Tricks erfreuen. Beide Methoden kombiniert ergeben einen recht wirkungsvollen Programmschutz.

(Oliver Rahm/tr)

Englisch für Fortgeschrittene

Geben sie einmal folgende kleine Basic-Zeile sorgfältig ein.

```
REM">> 7 95 70 T 0ITI- 7 H 2 0 N 5 << <247>
```

Wenn Sie die Zeile listen, sehen Sie nichts Ungewöhnliches. Löschen Sie nun das Anführungszeichen hinter der REM-Anweisung und listen Sie die Zeile. Mehr wird hier nicht verraten. Lassen Sie sich überraschen!

(Yves Forkl/tr)

Die mysteriöse Zahl 35072120

Zur Erinnerung: In der Ausgabe 1/86 schrieben wir, daß der List-Schutz-»POKE 808,225« durch die Eingabe der Zahl 35072120 wieder zurückgesetzt werden kann. Wir fragten weiterhin, worauf dieses Phänomen zurückzuführen ist.

Wir danken allen Lesern, die sich bei der Lösung des Problems beteiligt haben. Hier stellvertretend für viele Einsendungen die Erklärung von Frank Sander:

Nun, was geschieht da, besonders wo der C 64 derartig hohen Zeilennummern sonst nur mit einem kalten Lächeln (auch Syntax-Error genannt) begegnet?

Es geschieht folgendes:

Die Eingabewarteschleife (\$A480) stellt fest, daß die soeben eingegebene Zeile (»35072120«) mit einer Ziffer beginnt und entscheidet daher, daß es sich um eine in das Programm einzubauende Programmzeile handelt.

Also übergibt sie die Zeile an die Routine zum Einfügen von Programmzeilen (\$A49C).

Die Einfügeroutine versucht nun als erstes (und in diesem Falle auch als letztes), die Zeilennummer ins Low-/High-Byte-Format zu verwandeln. Dazu verzweigt sie nach \$A96B, wo die Routine liegt, die für das ganze »Wunder« verantwortlich ist. Denn: Ist die Zeilennummer (wie unsere) zu groß, so versucht die Routine verzweifelt, einen Syntax-Error von sich zu geben (aber die entsprechende Ausgabeschleife ist für einen BRANCH-Befehl zu weit entfernt). Also (da man ja unmöglich drei Byte für einen »JMP \$AF08« opfern konnte) wird nach \$A953 verzweigt, in der Hoffnung, daß der Akku etwas anderes als \$89 enthält. Denn \$A953 liegt mitten in der ON-GOTO-Routine und dort wird, falls der Akku etwas anderes als den GOTO-Code (eben \$89) enthält, zu einem Syntax-Error-Ansprung verzweigt.

Nur lautet die Zeilennummer 35072120, so ist der im Akku enthaltene »Rest« dummerweise gerade \$89, also wird nicht zum Syntax-Error gesprungen, sondern der Prozessor beginnt eine Fahrt ins Ungewisse — vorbei an total unpassenden Befehlen:

— vorbei an einem JSR CHRGET

— weiter bei einem erneuten Versuch, eine Zeilennummer zu holen (leider ist aber die Zeile zu Ende, also \$00)

— dieses \$00 ist auch nicht gleich einem Komma... also muß der auf dem Stack liegende Code für ON-GOTO geholt werden (PLA) und das Ganze endet mit einem RTS.

Nur... es liegt ja gar kein ON-GOTO-Code auf dem Stack, sondern die Rücksprungadresse aus der Eingabewarteschleife!

Also wirft der ahnungslose Prozessor das Rücksprung-Low-Byte in die Ecke und hält nun das Rücksprung-High-Byte für ein Low-Byte und das nachfolgende Byte für ein High-Byte. Daraus folgt: der Prozessor springt ins Ungewisse: zu einer Adresse mit Low-Byte \$A5. Als High-Byte liegt bei ordnungsgemäß gereinigtem Stack \$79 obenauf (ehemaliger Rücksprung nach \$A67A während der Basic-Initialisierung). Dies führt zu einem Sprung nach \$79A5, wo der Professor hoffentlich auf ein BRK trifft, den sonst... Einen schweigenden, cursorlosen C 64 hat wohl jeder schon einmal gesehen.

Aber meist liegt dort ein BRK (oder ein mit TAX gepflasterter Weg dorthin), was den wild gewordenen Computer wieder auf den rechten Weg bringt: Nämlich die NMI-Routine.

Diese NMI-Routine, die bei unblockierter RUN/STOP-Taste auch über RUN/STOP-RESTORE erreichbar ist, setzt unter anderem den Stop-Vektor (Adresse 808/809!) neu und schon ist das Listing wieder auf den Beinen.

Hier noch einmal zusammengefaßt die Irrfahrt des C 64:

A494	BCC A49C	falls Zeile mit Ziffer am Anfang
A49C	JSR A96B	Zeilennummer in Low/High-Format wandeln
A96B		
A97B	CMP #19	
A97D	BCS A953	falls die Zeilennummer zu groß wird
A953	CMP #89	was ja zutrifft
A955	BNE A8E8	Syntax-Error-Ansprung
A957	DEC \$65	
A959	BNE A95F	\$65 war alles andere nur nicht 1, also verzweigen
A95F	JSR 0073	CHRGET
A962	JSR A96B	Zeilennummer holen, bei Zeilenende \$00
A965	CMP #", "	\$00 ist aber nicht gleich ", "
A967	BEQ A957	also nicht verzweigen
A969	PLA	läßt das Rücksprung-Low-Byte verschwinden

A96A RTS

Oben auf dem Stack hätte \$9E, \$A4, \$79, \$A6 gelegen, das \$9E wurde jedoch durch das PLA vom Stack geholt. Also erfolgt ein »Rücksprung« nach \$79A5.

Man kann dies ganz einfach ausprobieren, indem man in den (frisch eingeschalteten) Computer, ein POKE 31141,2 (= \$79A5) eingibt. Dies führt dann bei Eingabe von »35072120« zum Absturz des Prozessors. (Befehl \$02 = Prozessorabsturz).

So weit so gut. Nur — hätte es ein POKE 2,0:SYS2 nicht auch getan?

(Frank Sander/tr)

HiRes-Scrolling per Interrupt

Wer hat nicht schon solche Titelbilder wie bei dem Spiel »Boulderdash« gesehen? Man hat den Eindruck, als ob der Hintergrund stufenlos scrollen würde, die Schrift aber stehenbliebe. Solch ein Effekt wäre natürlich auch hervorragend für eigene Spielszenen geeignet. Das folgende Maschinenprogramm läuft völlig interruptgesteuert und simuliert diese Funktion in vielfältiger Weise. Das Maschinenprogramm (Listing 1) wird zuerst mit »SYS 49152« gestartet.

Der erste Befehl schaltet das Pseudo-Scrolling ein. Die Syntax lautet: »SYS 49232,a,b,c«. Nun zu den Parametern: Der Parameter a ist für die Geschwindigkeit zuständig. Sinnvoll

programm : pseudo-scroll c000 c15f

```
c000 : 78 a5 01 48 29 fb 85 01 1f
c008 : a9 d0 85 03 a9 38 85 05 58
c010 : a0 00 84 02 84 04 a2 10 25
c018 : b1 02 91 04 c8 d0 f9 e6 78
c020 : 03 e6 05 ca d0 f2 68 85 82
c028 : 01 ad 18 d0 09 0e 8d 18 87
c030 : d0 58 a9 08 20 d2 ff a2 76
c038 : 28 a0 00 bd 07 c1 99 c0 ce
c040 : 3a e8 c8 c0 08 d0 f4 a9 67
c048 : 05 85 fa a9 00 85 fe 60 ed
c050 : 20 37 c0 20 fd ae 20 9e 53
c058 : b7 8e 05 c1 20 fd ae 20 bd
c060 : 9e b7 e0 00 d0 03 20 8d d3
```

```
c068 : c0 20 fd ae 20 9e b7 e0 25
c070 : 00 d0 0d 78 a9 aa a0 c0 1f
c078 : 8d 14 03 8c 15 03 58 60 ed
c080 : 78 a9 bf a0 c0 8d 14 03 9f
c088 : 8c 15 03 58 60 a2 00 a9 d9
c090 : 58 9d 00 04 e8 d0 fa 9d 74
c098 : 00 05 e8 d0 fa 9d 00 06 18
c0a0 : e8 d0 fa 9d e8 06 e8 d0 67
c0a8 : fa 60 20 e7 c0 d0 35 e6 0d
c0b0 : fa a6 fa e0 0b d0 05 a9 77
c0b8 : 03 aa 85 fa 4c d1 c0 20 68
c0c0 : e7 c0 d0 20 c6 fa a6 fa 14
c0c8 : e0 ff d0 05 a9 07 aa 85 05
c0d0 : fa a0 00 8a 0a 0a 0a aa da
c0d8 : bd 07 c1 99 c0 3a e8 c8 cf
c0e0 : c0 08 d0 f4 4c 31 ea e6 3f
```

```
c0e8 : fe a5 fe cd 05 c1 f0 03 5a
c0f0 : a9 0a 60 a9 00 85 fe 60 d5
c0f8 : 78 a9 31 a0 ea 8d 14 03 17
c100 : 8c 15 03 58 60 00 00 3c 61
c108 : 3c 18 00 00 00 00 18 3c 29
c110 : 18 00 00 00 00 18 3c 18 0a
c118 : 00 00 00 00 18 3c 00 6d
c120 : 00 00 00 18 3c 3c 18 00 2a
c128 : 00 00 18 3c 3c 18 00 00 3b
c130 : 00 18 3c 3c 18 00 00 00 55
c138 : 18 3c 3c 18 00 00 00 18 b1
c140 : 3c 3c 18 00 00 00 00 3c 19
c148 : 3c 18 00 00 00 00 18 3c 69
c150 : 18 00 00 00 00 18 3c 18 4a
c158 : 00 00 00 00 18 3c 3c 00 ad
```

Listing 1. Das Stenoprogramm zum HiRes-Scrolling. Bitte mit dem MSE eingeben.

sind nur die Werte 1 bis 255, wobei 1 die höchste Geschwindigkeit bewirkt. Der Parameter b kann die Werte 0 oder 1 annehmen. Ist b=0, so wird nach dem Aufruf der gesamte Bildschirm mit dem zu scrollenden Zeichen aufgefüllt. Das ist zu Anfang ein kleiner Kreis, kann aber von Ihnen im Quelltext geändert werden. Ist b=1, so wird der Bildschirm nicht aufgefüllt. Dies wird zum Beispiel in dem Demoprogramm ausgenutzt.

Der Parameter c kann ebenfalls die Werte 0 oder 1 annehmen. Ist c=0, so wird nach oben gescrollt, ist c=1, so wird nach unten gescrollt. Abgeschaltet wird das Ganze mit SYS 49400. Am besten starten Sie zu Anfang das Demoprogramm (Listing 2), um sich mit den Befehlen vertraut zu machen. Zuvor müssen Sie aber das Maschinenprogramm absolut (mit »8,1«) geladen haben.

Zur Funktionsweise:

Zuerst wird der Zeichensatz umkopiert und einige Zeiger initialisiert. Der Trick beim Scrolling besteht darin, daß im Interrupt kontinuierlich das Zeichen »SHIFT-X« (Bildschirmcode=88) umdefiniert wird. Dadurch scheint der Hintergrund gescrollt zu werden. Unterbrechen Sie doch einmal das Demoprogramm mit RUN/STOP, löschen Sie den Bildschirm und listen das Programm. Sie werden sehen, daß sogar »im Listing« die Zeichen gescrollt werden.

Hinweis: Wenn Sie die neue Version vom »SMON« besitzen (64'er, Ausgabe 10/85), dann können Sie, nachdem Sie den SMON mit »Y8000« verschoben haben, das zu scrollende Zei-

chen beliebig mit dem »Z«-Befehl umdefinieren. Die Zeichendefinitionen liegen ab \$C107.

(Bergerhoff/Nikolas/tr)

1520-Plotter als vollwertiger Drucker

Dieses kleine Programm (Listing 3) schickt alle Druckausgaben, die zur Geräteadresse 4 gesendet werden, automatisch zum 1520-Plotter mit der Geräteadresse 6. Das heißt, daß fertige Programme nicht mehr mühselig nach OPEN-Befehlen durchkämmt werden müssen, um die Druckerausgaben entsprechend umzulenken. Weiterhin wird der Plotter automatisch auf Kleinschrift gestellt, wenn diese am C 64 eingeschaltet ist.

Auch läßt sich die Zeilenlänge am Plotter einstellen: »POKE 713,Z:POKE 754,Z«, wobei folgende Werte für Z zulässig sind:

- 0: 80 Zeichen/Zeile (eingestellter Wert)
- 1: 40 Zeichen/Zeile
- 2: 20 Zeichen/Zeile

(N.J.W. Cuppen/tr)

Kalauer des Monats:

»God Save the Queen Komma acht Komma eins!«

```

0 REM ***** <053>
1 REM ** PSEUDO-SCROLL-DEMO ** <214>
2 REM ***** <218>
3 REM ** (C) 1985 BY ** <183>
4 REM ** N.BERGERHOFF ** <064>
5 REM ** MEISENWEIG 22 ** <003>
6 REM ** 5020 FRECHEN 4 ** <022>
7 REM ** TEL. 02234/64610 ** <139>
8 REM ***** <061>
9 : <241>
10 INIT=49152:EIN=49232:AUS=49400 <034>
11 SYS INIT <241>
12 : <244>
100 PRINT " {CLR,WHITE,5SPACE}XXX{3SPACE}XXX <038>
    XXXX X{2SPACE}X XXX{2SPACE}XXXX
110 PRINT " {5SPACE}X{2SPACE}X X{4SPACE}X{4S <137>
    PACE}X{2SPACE}X X{2SPACE}X X{2SPACE}X
120 PRINT " {5SPACE}X{2SPACE}X X{4SPACE}X{4S <147>
    PACE}X{2SPACE}X X{2SPACE}X X{2SPACE}X
130 PRINT " {5SPACE}XXX{3SPACE}XX{2SPACE}XXX <200>
    {2SPACE}X{2SPACE}X X{2SPACE}X X{2SPACE
    }X
140 PRINT " {5SPACE}X{7SPACE}X X{4SPACE}X{2S <140>
    PACE}X X{2SPACE}X X{2SPACE}X
150 PRINT " {5SPACE}X{7SPACE}X X{4SPACE}X{2S <150>
    PACE}X X{2SPACE}X X{2SPACE}X
160 PRINT " {5SPACE}X{4SPACE}XXX{2SPACE}XXXX <156>
    {2SPACE}XX{2SPACE}XXX{2SPACE}XXXX
170 PRINT <016>
180 PRINT " {6SPACE}XXX{2SPACE}XXX XXX{2SPAC <093>
    E}XXXX X{4SPACE}X
190 PRINT " {5SPACE}X{4SPACE}X{4SPACE}X{2SPA <049>
    CE}X X{2SPACE}X X{4SPACE}X
200 PRINT " {5SPACE}X{4SPACE}X{4SPACE}X{2SPA <059>
    CE}X X{2SPACE}X X{4SPACE}X
210 PRINT " {6SPACE}XX{2SPACE}X{4SPACE}XXX{2 <237>
    SPACE}X{2SPACE}X X{4SPACE}X
220 PRINT " {8SPACE}X X{4SPACE}X{2SPACE}X X{ <079>
    2SPACE}X X{4SPACE}X
230 PRINT " {8SPACE}X X{4SPACE}X{2SPACE}X X{ <089>
    2SPACE}X X{4SPACE}X
240 PRINT " {5SPACE}XXX{3SPACE}XXX X{2SPACE} <169>
    X XXXX XXXX XXXX

```

Listing 2. Ein Demo-Programm zum HiRes-Scrolling.

```

250 SYS EIN,3,1,0 <094>
260 GOSUB 1000:PRINT " {LIG.BLUE,CLR}" <252>
270 SYS EIN,3,0,0 <114>
280 PRINT TAB(11) " {3DOWN,WHITE}PSEUDO-SCRO <007>
    LL-DEMO"
290 PRINT TAB(17) " {4DOWN}HOCH" <002>
300 GOSUB 1000
310 SYS EIN,3,1,1 <164>
320 PRINT TAB(16) " {UP}RUNTER" <135>
330 GOSUB 1000 <032>
340 SYS EIN,1,1,1 <193>
350 PRINT TAB(16) " {UP}SCHNELL" <132>
360 GOSUB 1000 <062>
370 SYS EIN,10,1,1 <216>
380 PRINT TAB(16) " {UP}LANGSAM" <203>
390 GOSUB 1000 <092>
400 PRINT " {LIG.BLUE,CLR}":SYS EIN,3,0,0 <095>
410 PRINT TAB(12) " {WHITE}TASTE {LIG.BLUE}X{ <096>
    WHITE}DRUECKEN"
420 WAIT 198,255:POKE 198,0:END <126>
430 : <152>
1000 FOR I=1 TO 3000:NEXT:RETURN <132>

```

© 64'er

Listing 2. »Demo« HiRes-Scrolling (Schluß)

```

10 DIM H(75):FOR I=0 TO 9 <139>
20 H(48+I)=I:H(65+I)=I+10:NEXT <067>
30 FOR I=679 TO 764:READ A$ <072>
40 H=ASC(A$):L=ASC(RIGHT$(A$,1)) <132>
50 D=H(H)*16+H(L):POKE I,D <165>
60 NEXT:POKE 794,167:POKE 795,2 <080>
100 DATA A5,B8,B5,FB,A5,BA,C9,04,F0,03,4C, <241>
    4A,F3,A9,03,85,B9,A9,06,85
110 DATA BA,A9,7F,85,B8,20,4A,F3,A6,B8,20, <091>
    C9,FF,A9,00,20,D2,FF,A5,B8
120 DATA 20,C3,FF,A9,66,C5,B9,F0,15,85,B9, <039>
    A9,7E,85,B8,AD,18,D0,29,02
130 DATA 4A,18,69,30,8D,C9,02,4C,C0,02,A9, <241>
    00,85,B9,A9,00,8D,C9,02,A5
140 DATA FB,85,B8,4C,4A,F3 <041>

```

© 64'er

Listing 3. Der 1520-Plotter als vollwertiger Drucker.