Tips & Tricks für Profis

Hätten Sie gedacht, daß die rote LED an Ihrer Floppy ihre Helligkeit verändern kann? Oder daß ein kurzes Maschinenprogramm den Bildschirm auf den Kopf stellt? Auch diesmal haben wir für Sie einige Beiträge gesammelt, die selbst ausgefuchste Profis in Erstaunen versetzen werden.

Beginnen wollen wir diesmal mit einer wirklich außergewöhnlichen »Basic-Erweiterung«: Sie ermöglicht es, bis zu 252 Zeichen lange Basic-Zeilen einzugeben! Damit taucht unwillkürlich folgende Frage auf: Ist dies noch ein Einzeiler oder nicht?

Die Super-Einzeiler

Wer nicht will, daß sein Basic-Programm unnötig viel Speicher verbraucht, da er jedes freie Byte für Daten gebrauchen kann, oder wer will, daß sein Programm möglichst schnell ist, weil ihm jede Sekunde kostbar ist, für den ist es oft von Nutzen, mehrere Programmzeilen zu einer zusammenzufassen. Jede Zeile, die neu begonnen wird, verbraucht nämlich vier Byte mehr Speicherplatz, als wenn diese mit einem Doppelpunkt an die vorhergehende Zeile angehängt würde. Bei einem längeren Programm kann da schon einiges zusammenkommen. Außerdem kann der Interpreter Befehle in einer Zeile schneller abarbeiten, als wenn jeder Befehl in einer eigenen Zeile steht. Wenn diese sich dann noch in einer Schleife befinden, die vielleicht einige hundertmal durchlaufen wird, könnte man hier schon durch eine andere Anordnung Zeit sparen. Um das Programm allerdings nicht allzu unübersichtlich zu machen, sollte man nur solche Zeilen zusammenfassen, die auch logisch zusammengehören.

Wenn man dies nun beherzigen will, wird man allerdings durch den Basic-Editor ziemlich eingeschränkt, da dieser nur maximal 80 Zeichen in einer Zeile zuläßt. Der Interpreter würde aber auch längere Zeilen akzeptieren, nur kann man solche eben nicht eingeben.

Um dieses Manko zu beseitigen, habe ich EX-LINE geschrieben (siehe Listing 1). Mit diesem Programm ist es nun möglich, mit einer Länge von bis zu 252 Zeichen Basic-Zeilen einzugeben.

Nachdem man dieses Programm mit dem MSE eingegeben oder mit »LOAD ''EX-LINE'',8,1« von Diskette geladen hat, kann man es mit »SYS 49152« starten. Danach steht das Ausrufungszeichen als neuer Befehl zur Verfügung. Wenn man diesen Befehl eingibt, erscheint der Cursor in der linken oberen Ecke des Bildschirms. Nun kann man eine Programmzeile oder Befehle im Direktmodus eingeben. Dabei ist zu beachten, daß nach »RETURN« nur die ersten 6,5 Bildschirmzeilen übernommen werden, egal wo sich der Cursor dann befindet. Wenn man nach dem Ausrufungszeichen noch eine Zeilennummer angibt, wird vorher noch der Bildschirm gelöscht und die entsprechende Zeile gelistet, so daß man auch überlange Zeilen editieren kann.

Da sich das Programm nicht mit dem Checksummer verträgt, errechnet es noch eine Prüfsumme, die sich mit »PRINT PEEK(2)« auslesen läßt. Sie entsteht einfach durch Addition der ASC-Werte der eingegebenen Zeichen. In der Speicherzelle 2 steht dann das Low-Byte der Summe.

Wenn man die Befehle in abgekürzter Form eingibt, erhält man eine andere Prüfsumme, als wenn man sie ausschreibt. Die angegebenen Prüfsummen bei den folgenden Programmen beziehen sich immer auf die ausgeschriebene Form.

Programmbeschreibung

Das Programm ist komplett in Maschinensprache geschrieben und steht im Speicher ab $C000. Nach dem Start mit »SYS 49152« wird zuerst das Basic-ROM in den darunter liegenden RAM-Bereich kopiert und so verändert, daß der Eingabepuffer nicht mehr bei $0200, sondern bei $C200 liegt. Dies ist nötig, da dieser jetzt mehr Platz benötigt, als das Betriebssystem für ihn vorsieht.

Schließlich wird der Vektor für »Basic-Befehl holen« bei $0308 auf die neue Routine gelenkt. Diese prüft, ob das erste Zeichen ein Ausrufungszeichen ist. Wenn dies der Fall ist und noch eine Zeilennummer folgt, wird zunächst der Bildschirm gelöscht und die entsprechende Zeile gelistet. (Wenn keine Zeile angegeben war, wird dieser Teil übersprungen.)

Nachdem dann die Eingabe erfolgt ist, wird das RAM bei $A000 aktiviert, der Text im oberen Teil des Bildschirms in den Puffer bei $C200 kopiert, in Interpretercode umgewandelt und ausgeführt. Das ROM wird bei der nächsten Eingabe wieder eingeschaltet, so daß der Eingabepuffer wieder wie üblich bei $0200 liegt.

```
programm : ex-line          c000 c12e
-------------------------------------

c000 : a0 00 a2 a0 84 f7 86 f8  71
c008 : a2 20 b1 f7 91 f7 c8 d0  c3
c010 : f9 e6 f8 e6 fa ca d0 f2  c6
c018 : a9 c2 8d f5 a4 8d 84 a5  58
c020 : 8d ba a5 8d e7 a5 8d 06  13
c028 : a6 a9 c1 8d 13 a5 8d 16  86
c030 : a5 8d 24 a5 8d cd a5 8d  53
c038 : d0 a5 8d f1 a5 8d 0b a6  bd
c040 : 8d d2 aa 8d fd c1 a2 87  1a
c048 : a0 c0 8e 08 03 8c 09 03  ac
c050 : a9 d0 a2 7a 8d 28 a5 8e  27
c058 : 2b a5 8c 2c a5 a9 20 a2  6c
c060 : 82 8d 80 a4 8e 81 a4 8c  fe
c068 : 82 a4 a2 00 bd 1d c1 f0  93
c070 : 06 20 d2 ff e8 d0 f5 6c  01
c078 : 02 a0 b9 fc c1 91 5f 4c  97
c080 : 59 a6 a9 37 85 01 60 20  a0
c088 : 73 00 c9 21 f0 03 4c e7  ba
c090 : a7 a2 36 86 01 20 73 00  c6
c098 : b0 21 a2 ae a0 c0 8e 00  ai
c0a0 : 03 8c 01 03 48 a9 93 20  ea
c0a8 : d2 ff 68 4c a4 a6 20 ea  f3
c0b0 : e8 a2 8b a0 e3 8e 00 03  99
c0b8 : 8c 01 03 a9 13 20 d2 ff  38
c0c0 : 20 cf ff ad 00 04 48 a2  04
c0c8 : 00 86 02 a5 d4 48 a9 00  77
c0d0 : 85 d4 a9 13 20 d2 ff a9  78
c0d8 : 27 85 d0 bd 00 04 8d 00  04
c0e0 : 04 68 85 d4 20 cf ff 9d  d0
c0e8 : 00 c2 18 65 02 85 02 e8  22
c0f0 : d0 d9 8a 8d ff c2 a2 fc  9c
c0f8 : bd 00 c2 c9 20 d0 04 ca  cd
c100 : b8 50 f5 e8 a9 00 9d 00  8c
c108 : c2 68 8d 00 04 a9 0d a2  69
c110 : 03 20 d2 ff ca d0 fa 20  37
c118 : cf aa 4c 86 a4 0d 45 58  99
c120 : 2d 4c 49 4e 45 20 42 59  a1
c128 : 20 48 43 45 0d 00 ff ff  b7
```

Listing 1. Die Basic-Erweiterung »EX-LINE«

Um zu zeigen, was man mit diesem Programm alles in einer Zeile unterbringen kann, habe ich folgende drei »Einzeiler« geschrieben:

1. SOFT FLASH:

Dieser Einzeiler (Listing 2) bewirkt, daß die rote LED am Diskettenlaufwerk scheinbar stufenlos ein- und ausgeschaltet wird. Wenn man das Programm mit Hilfe von »EX-LINE« eingegeben hat, kann man mit »PRINT PEEK(2)« die Prüfsumme abfragen. Diese sollte 180 betragen.

Programmbeschreibung:

In der FOR-NEXT-Schleife wird ein Maschinenprogramm mittels »Memory-Write« in das RAM der Floppy ab $0500 geschrieben. Der Floppy-Befehl »UC« bewirkt dann, daß dieses gestartet wird. Das Maschinenprogramm schaltet die LED so schnell an und aus, daß dies für das Auge nicht sichtbar ist. Dabei ändert sich die Länge der Hell- und Dunkelphase, so

daß es scheint, als ob die Lampe langsam hell und dunkel würde.

Weil das Programm als Endlosschleife geschrieben ist, läßt sich das Laufwerk nicht mehr ansprechen. Wenn man die Floppy weiter verwenden will, muß man sie vorher aus- und anschalten.

```
1 OPEN 1,8,15:FOR I=0 TO 44:READ A:PRINT#1
  ,"M-W"CHR$(I)CHR$(5)CHR$(1)CHR$(A):NEXT:
  PRINT#1,"UC":DATA 169,254,170,32,21,5,20
  2,224,1,208,248,32,21,5,232,224,255,208,
  248,240,235,138,72,73,255,168,169,248,14
  1,0,28,202,208,248,169,240,141,,28,136,2
  08,248,104,170,96
```

64'er

Listing 2. »Soft Flash« läßt die rote LED der Floppy stufenlos an- und ausgehen. Zur Eingabe bitte Listing 1 verwenden.

2. STRICH-CURSOR:

Dieser Einzeiler (Listing 3) verwandelt den Cursor in einen Strich, der unter den Zeichen blinkt, so wie man es oft bei größeren Computern sieht. Man kann diese Routine verwenden, um seinen Programmen ein professionelles Aussehen zu verleihen (Prüfsumme: 186).

Programmbeschreibung:

Die ersten drei POKE-Befehle bewirken, daß der Bildschirm nach $CC00 verlegt wird und der Zeichensatz aus dem RAM ab $D000 gelesen wird. Dies ist notwendig, da kein Basic-Speicher verloren gehen soll und der VIC nur 16 KByte auf einmal adressieren kann.

Bevor man nun den Originalzeichensatz bei $D000 mit »POKE 1,3« lesbar machen kann, muß noch der Interrupt mit »POKE 56333,1« ausgeschaltet werden. Die erste FOR-NEXT-Schleife kopiert den Bildschirm an seine neue Position. Die zweite Schleife kopiert den Zeichensatz ins RAM. Dabei werden die reversen Zeichen so verändert, daß nur die unterste Reihe revers erscheint. Die folgenden POKEs bewirken schließlich, daß der Interrupt eingeschaltet wird und der I/O-Bereich bei $D000 wieder ansprechbar ist.

```
1 W=56333:Q=53248:Z=41↑5:POKE Q+24,52:POKE
  56576,0:POKE 648,204:POKE W,1:POKE 1,3:F
  OR I=0 TO 999:POKE 52224+I,PEEK(Z+I):NEX
  T:FOR I=0 TO Z:A=Q+I:B=A+2*Z:L=Z*((I AND
   7)=7):POKE A,PEEK(A):POKE B,PEEK(B):POK
  E A+Z,PEEK(A-L):POKE B+Z,PEEK(B-L):NEXT:
  POKE 1,7:POKE W,129
```

64'er

Listing 3. »Strich-Cursor«. Verleiht Ihrem Cursor ein äußerst professionelles Aussehen

3. UPSIDE-DOWN:

Dieses Programm habe ich in zwei Versionen geschrieben, da es als Einzeiler (Listing 4) sehr schwer abzutippen ist. Die zweite Version (Listing 5) wird mit dem MSE eingegeben (die Prüfsumme für den Einzeiler lautet 76). Beide Versionen können einfach mit »RUN« gestartet werden. Danach erscheint der ganze Bildschirm »auf den Kopf gestellt«. Dies bezieht sich auf *alle* Ein- und Ausgaben. Auch Zeichen, die mit »POKE« auf den Bildschirm gebracht werden, erscheinen an der entsprechenden anderen Stelle auf dem Kopf. Die meisten Programme vertragen sich damit gut, da nur der IRQ-Vektor verändert wird. Nach einem Reset oder RUN/STOP-RESTORE kann das Programm mit »SYS 300« wieder aktiviert werden.

Programmbeschreibung:

Das Maschinenprogramm, das bei dem Einzeiler mittels »PRINT« auf den Bildschirm gebracht wird, kopiert zunächst eine Interruptroutine in den Stack ab Adresse 300. (Dieser Bereich ist besonders geeignet, da er nach einem Reset nicht gelöscht wird und ein Programm dort nicht stört.) Dann wird der gesamte Zeichensatz auf den Kopf gestellt und in das RAM bei $D000 kopiert. Der VIC wird veranlaßt, den Zeichensatz aus $D000 und den Bildschirm aus $E000 zu lesen, und schließlich wird die Interruptroutine aktiviert, die 60mal in der Sekunde den Bildschirm umgekehrt nach $E000 kopiert. (H. C. Edelmann/tr)

```
1 PRINT"{HOME,RVSON,SPACE,RVOFF}*{RVSON}9{
  RVOFF}CD{RVSON}Y{RVOFF},A{RVSON}H{RVOFF}
  P{RVSON}▽{RVOFF}▽{RVSON}){RVOFF}3{RVSON}
  E{RVOFF}A{RVSON}){RVOFF}@{RVSON}E▽)PE▽{S
  PACE,RVOFF}@{RVSON}1▽.9#{RVOFF}J&B{RVSON
  }JP@X{RVOFF}IG{RVSON}(%{RVOFF}B{RVSON}Q▽
  X{RVOFF}IG{RVSON}(HP++▽%▽I{SHIFT-SPACE}P
  ↑){RVOFF}7{RVSON}E{RVOFF}AL,A▽{RVSON}}DM
  {RVOFF}X{RVSON}P){RVOFF}@{RVSON}M{RVOFF}
  @{RVSON}_){RVOFF}C{RVSON}M{RVOFF}TC{RVSO
  N}){RVOFF}A{RVSON}M{RVOFF}UCX{SHIFT-SPAC
  E,RVSON}){RVOFF}@{RVSON}*(H={RVOFF}@D{RV
  SON}Y?I={RVOFF}@E{RVSON}Y?I={RVOFF}@F{RV
  SON}Y?{SHIFT-SPACE}={RVOFF}@G{RVSON}Y?H
  ?P?{RVOFF}L1{RVSON}▽{RVOFF}":SYS 1024
```

64'er

Listing 4. »Upside-Down«. Dreht den Bildschirm um 180 Grad! Beachten Sie bitte die Eingabehinweise im Text.

```
programm : upside-down v2  0801 0896
------------------------------------

0801 : 11 08 00 00 9e 32 30 37  c1
0809 : 32 20 ab 20 48 43 45 00  ee
0811 : 00 00 00 00 00 00 00 a0  53
0819 : 40 b9 5b 08 99 2c 01 88  1e
0821 : 10 f7 78 a9 33 85 01 a9  37
0829 : 00 85 f7 a9 d0 85 f8 a0  7d
0831 : 00 b1 f7 ae b9 a3 4a 26  0c
0839 : 02 ca d0 fa 98 49 07 a8  75
0841 : a5 02 91 f7 98 49 07 a8  8c
0849 : c8 d0 e6 e6 f8 a5 f8 c9  44
0851 : e0 d0 de a9 37 85 01 4c  c2
0859 : 2c 01 78 a9 84 8d 18 d0  10
0861 : a9 00 8d 00 dd a9 43 8d  c1
0869 : 14 03 a9 01 8d 15 03 58  c8
0871 : 60 a9 00 aa a8 88 bd 00  c1
0879 : 04 99 e8 e2 bd 00 05 99  03
0881 : e8 e1 bd 00 06 99 e8 e0  5c
0889 : bd 00 07 99 e8 df 88 e8  bd
0891 : d0 e4 4c 31 ea 20 8e b4  60
```

Listing 5. »Upside-Down V2«. Listing 4 als MSE-Dump

Print Shop und CP 80X

Probleme traten bei Print Shop und angeschlossenem Drucker (Melchers) CP 80X auf. Das lag daran, daß Print Shop nach jeder Druckzeile ein $0A sendet. Dies wird aber vom CP 80X laut Handbuch ignoriert. Abhilfe: Im Programmteil »Syslib« ($8800-$A0C0) die Adresse $9B31 von $0A auf $0D (= 13, entspricht Carriage Return + Linefeed) abändern und beim Setup den Druckertyp »Mannesmann Tally Spirit 80« anwählen. Seit dieser Änderung läuft Print Shop mit CP 80X einwandfrei. Die gesuchte Adresse befindet sich auf der (Original-) Disk im 20. Block, Adresse 5B von »Syslib«.

(Jürgen Römer/tr)

Disk-Optimizer

Ein nützliches Utility für Sparsame. Das Programm läuft auf allen Commodore-Computern, wenn man den DIM-Befehl in Zeile 60 dem verfügbaren Speicherplatz des jeweiligen Computers anpaßt.

Das Programm (Listing 6) ermöglicht die optimale Ausnutzung der Disketten. Man kann seine Programmsammlung auf

die kleinstmögliche Anzahl von Disketten verteilen und belastet damit den schmalen Geldbeutel vieler Computerfreaks nicht so stark. Ein Anwenderprogramm dieser Art ist bis jetzt nicht nur einzigartig, sondern nützt auch wirklich etwas — wird seinem Namen (Utility) also gerecht!

Anwendungsbeispiel:

Wir haben 13 Programme zur Verfügung, die auf möglichst wenige Disketten verteilt werden sollen. Sie haben folgende Blocklängen: 116,153,182,6,33,191,38,84,156,5,86,186,113. Nach dem Programmstart mit »RUN« geben wir die freien Blöcke einer Diskette ein (meistens 664) und anschließend die Blocklängen der 13 Programme in beliebiger Reihenfolge. Vertippt man sich, so kann man mit einer »0« oder einfach RETURN die jeweils letzte Eingabe berichtigen. Zum Abschluß der Eingabe muß die Zahl »999« folgen.

Jetzt sucht das Programm die Blöcke heraus, die addiert die vorher eingegebene Summe ergeben. Damit keine gleichen Zahlenfolgen auf dem Bildschirm erscheinen, sortiert das Programm solche Zahlenfolgen, die bereits schon einmal gefunden wurden, aus. Nach einer Weile sehen wir, daß es mehrere Kombinationen gibt, diese Programme platzsparend auf wenigen Disketten unterzubringen. Wir suchen uns die passenden aus und erhalten durch geschicktes Umkopieren der Programme auf eine oder mehrere Disketten brauchbaren Speicherplatz. Hätten Sie gedacht, daß man mit diesen 13 Programmen so viele verschiedene Möglichkeiten hat, eine Diskette randvoll zu füllen? Die Bedeutung der »Versuche«-Anzeige im Fenster oben links wird in der Programmbeschreibung erklärt. Das Programm sucht so lange nach neuen Folgen, bis Sie durch Drücken der START-/STOP-Taste »S« neu beginnen.

Da das Basic-Programm relativ langsam arbeitet, empfiehlt es sich, es zu compilieren. Erst bei 3- bis 4mal schnellerer Bearbeitungsgeschwindigkeit wird man es gern einsetzen. Die Programmservice-Diskette enthält neben der Basic- auch die compilierte Version. (Martin Boels/tr)

```
40 AP=200                                                    <068>
50 DL=20                                                     <034>
60 DIM B(AP),E(AP),Z(AP),GS(AP,DL)                           <222>
80 GOSUB 140                                                 <088>
90 GOSUB 210                                                 <058>
100 GOSUB 300                                                <060>
110 GOSUB 350                                                <150>
120 GOTO 90                                                  <106>
140 PRINT"{CLR,DOWN,SPACE}DISK-OPTIMIZER:"
    :PRINT" ßßßßßßßßßßßßßßß":INPUT"{DOWN,S
    PACE}SUMME =";S                                          <023>
150 PRINT"{DOWN}BEI FEHLERN '0' EINGEBEN."
    :PRINT"ENDE: '999'":PRINT                                <149>
160 AZ=AZ+1:PRINT" ZAHL"AZ;:INPUT B(AZ)                      <076>
170 IF B(AZ)=0 THEN AZ=AZ-2:PRINT"{2UP}";:
    GOTO 160                                                 <067>
180 IF B(AZ)<>999 GOTO 160                                   <230>
190 AZ=AZ-1                                                  <132>
200 PRINT"{CLR}SUMME{3SPACE}:"S:PRINT"ZAHL
    EN{2SPACE}:";AZ:RETURN                                   <154>
210 V=V+1:PRINT"{HOME,2DOWN}VERSUCH :"V                      <177>
220 GET B$:IF B$="S"THEN RUN                                 <099>
230 FOR I=1 TO T:B(Z(I))=E(I):NEXT:T=0:BS=
    0                                                        <235>
240 Z%=RND(1)*AZ+1:IF B(Z%)=0 GOTO 240                       <120>
250 BS=BS+B(Z%):T=T+1:E(T)=B(Z%):B(Z%)=0:Z
    (T)=Z%                                                   <169>
260 IF BS>S GOTO 210                                         <157>
270 IF BS=S THEN RETURN                                      <016>
280 IF T=AZ THEN PRINT"{2DOWN}SUMME ALLER
    ZAHLEN IST KLEINER ALS";S:END                            <197>
290 GOTO 240                                                 <036>
300 G=T-1:FOR X=G TO 1 STEP-1                                <024>
310 F=0:FOR Y=1 TO G:IF E(Y)<=E(Y+1)GOTO 3
    30                                                       <034>
320 F=Y:M=E(Y):E(Y)=E(Y+1):E(Y+1)=M                          <088>
330 NEXT Y:G=F:IF F=0 THEN RETURN                            <019>
340 NEXT X:RETURN                                            <150>
350 FOR A=1 TO GS:FOR B=1 TO T                               <016>
```

Listing 6. »Disk-Optimizer«. Nützen Sie die vorhandenen Diskettenkapazitäten voll aus!

```
360 IF GS(A,B)<>E(B)THEN B=T:NEXT B,A:GOTO
    380                                                      <011>
370 NEXT B:RETURN                                            <004>
380 PRINT CHR$(19):GS=GS+1:FOR I=1 TO RZ+3
    :PRINT:NEXT                                              <140>
390 FOR I=1 TO T:GS(GS,I)=E(I):PRINT E(I);
    :NEXT                                                    <243>
400 RZ=RZ+1:IF RZ<20 THEN RETURN                             <245>
410 GET A$:IF A$=""GOTO 410                                  <219>
420 RZ=0:GOTO 200                                            <150>

© 64'er
```

Listing 6. Schluß

Programmbeschreibung

Zeile	
40— 60:	Die vorgegebene Variablendimensionierung erfordert mindestens den Speicherplatz des C 64. Für kleinere Commodore-Computer muß knapper kalkuliert werden. AP gibt die größtmögliche Anzahl der zu verarbeitenden Programme an. DL gibt die größtmögliche Anzahl von Files auf einer Diskette an. Berechnung siehe im Listing.
80—120:	Hauptschleife mit Unterprogrammen
140—200:	Eingabe der Zahlen (Blocklängen) in B(AZ), AZ = Anzahl
210:	Anzeige der Versuche, die gewünschte Summe (664) zu erhalten.
220:	Abbruchmöglichkeit
230:	Initialisierung
240—250:	Zufälliges Aussuchen einer Blocklänge, doppelte ignorieren
260:	Diskette »überfüllt«, neu suchen
270:	Summe genau erreicht, Diskette voll
280:	Falls alle Blöcke addiert keine Diskette füllen: Abbruch
300—390:	Vergleich vorheriger Zahlenfolgen mit der jetzt gefundenen: falls neu: Bildschirmausgabe; sonst weitersuchen
400—420:	Scrollstop für den Bildschirm, falls unteres Ende erreicht; weiter mit Tastendruck

LoRes zu HiRes

Diese geniale Maschinenroutine (Listing 7) konvertiert den normalen Low-Resolution-Bildschirm, also den Bildschirm, der nach dem Einschalten aktiv ist, zu einem hochauflösenden Grafikbild und speichert dieses auf Diskette. Es kann sowohl die niedrig auflösende Grafik (Pixel-Grafik) als auch der normale Textbildschirm umgewandelt und gespeichert werden.

Bedienung

Nach dem Laden des Programms mit »LOAD''LORES ZU HIRES'',8,1« wird die Maschinenroutine mit »SYS 828,''BILDNAME'',8« aufgerufen. Die Parameter nach dem »SYS 828,« entsprechen genau den Angaben hinter dem normalen SAVE-Befehl. Da das Programm frei im Speicher verschiebbar ist, muß man nur die SYS-Startadresse auf den benutzten Speicherbereich anpassen. Ursprünglich liegt die Routine im Kassettenpuffer, wird also von keinem Basic-Programm gestört. Der Aufruf »SYS 828,''NAME'',8« kann sowohl im Programm- als auch im Direktmodus erfolgen. Im Direktmodus jedoch wird dann auch der eingegebene SYS-Befehl, der sich noch auf dem Bildschirm befindet, als Bestandteil des Textbildschirms gespeichert.

Anwendung:

Wird von einem LoRes-Grafikbild eine Hardcopy gemacht, so erscheint bei fast allen Druckern zwischen den Bildschirmzeilen ein weißer Streifen. Das ist beim Text ja auch sehr sinnvoll — wie sollte man ihn sonst auch lesen? Bilder werden durch diese Art des Druckes aber leider zerstört.

Nun hilft die »LoRes zu HiRes«-Routine: Man speichert sein Bild als hochauflösende Grafik, lädt es mit Hi-Eddi oder einem sonstigen Malprogramm wieder und kann nun mit die-

sem Programm eine »heile« Hardcopy erstellen. Wenn diese Hardcopy-Routine Bilder doppelt groß drucken kann (Hi-Eddi!), hat man erstmalig die Möglichkeit, einen LoRes-Bildschirm im Riesenformat auf Papier zu bringen.

Zweites Beispiel:

Wer schon einmal ein schönes Grafikbild mit Hi-Eddi erstellt hat, weiß, wieviel Arbeit das macht. Oft kann man die grobe, geometrische Struktur mit einem Maskengenerator (64'er, Ausgabe 8/85) erstellen. Da man sein Werk jetzt in Hi-Res übertragen kann, ist für die Feinarbeit mit Hi-Eddi schon gute Vorarbeit geleistet.

Drittens: Man kann sogar ein LoRes-Bild, das mit einem anderen Zeichensatz erstellt wurde, weiterverarbeiten! Der Konvertierungsroutine muß nur die Basisadresse des neuen Zeichensatzes mitgeteilt werden. Das High-Byte wird in die Speicherstelle 828+55 geschrieben, das Low-Byte muß $00 sein, braucht also nicht extra geändert zu werden.

Das Basic-Programm (Listing 8) ist äquivalent zu dem Maschinen. Interessant ist der Zeitvergleich: In Basic dauert die Hi-Res-Übertragung 2:13 min, in Maschinensprache 0,4 s! Die Basic-Version ist daher weniger zum Abtippen, sondern vielmehr zum leichten Verstehen des Algorithmus gedacht.

(Martin Boels/tr)

```
programm : lores zu hires  033c 03ba
-------------------------------------

033c : a9 00 8d 0e dc a9 33 85  fd
0344 : 01 a9 04 85 f8 a9 00 85  b3
034c : f7 85 f9 a9 20 85 fa a2  19
0354 : 04 a0 00 8a 48 98 48 a9  b7
035c : 00 85 fc b1 f7 85 fb 06  3c
0364 : fb 26 fc 06 fb 26 fc 06  63
036c : fb 26 fc 18 a5 fc 69 d0  46
0374 : 85 fc a2 00 a0 07 b1 fb  21
037c : 91 f9 88 10 f9 18 a5 f9  19
0384 : 69 08 85 f9 90 02 e6 fa  3c
038c : 68 a8 68 aa c8 d0 c4 e6  ac
0394 : f8 ca d0 bd a9 37 85 01  49
039c : a9 01 8d 0e dc 20 fd ae  0f
03a4 : 20 d4 e1 a9 20 85 fa a9  49
03ac : 00 85 f9 aa a9 f9 a0 40  b0
03b4 : 20 d8 ff 60 01 08 00 00  9d
```

Listing 7. »LoRes zu HiRes«. Verwandelt den Textbildschirm in eine HiRes-Grafik

```
10 POKE 56334,0:POKE 1,51                          <109>
20 FOR I=1024 TO 2023:BA=53248+PEEK(I)*8           <072>
30 FOR J=0 TO 7:POKE 8192+Z,PEEK(BA+J):Z=Z
   +1:NEXT J,I                                     <207>
40 POKE 1,55:POKE 56334,1                          <175>
```

Listing 8. Der grundlegende Algorithmus von Listing 7

Tips & Tricks zum C 128

Hauptsächlich neue POKEs und interessante Programmiertricks möchten wir Ihnen diesmal anbieten. Auch die Nutzung des kompletten deutschen Zeichensatzes im C 64-Modus ist nicht zu verachten.

Der C 128 ist im Kommen. Sein hervorragendes Basic macht ihn zu einem immer beliebteren Aufsteigercomputer. Aber gerade seine (im Vergleich zum C 64) komplexe, interne Struktur mit unter anderem drei Mikroprozessoren macht den C 128 zu einer Fundgrube für die Tips & Tricks-Ecke. Übrigens: Wenn Sie etwas Neues herausgefunden haben, schreiben Sie uns! Jeder brauchbare Trick wird veröffentlicht!

Programmieren in Z80-Assembler

Einige frischgebackene C 128-Besitzer möchten jetzt auch einmal in Z80-Assembler programmieren. Um den Z80-Mikroprozessor einzuschalten, muß das Bit 0 in Speicherstelle $fd505 auf 1 gesetzt werden. Doch wo beginnt der Prozessor seine Arbeit und was ist sonst noch zu beachten?

Zunächst müssen die System-IRQs gesperrt werden. Weiterhin sollte man sich in Bank 0 befinden (dort wird später das Z80-ROM von $0000 bis $1000 eingeblendet). Nachdem dies geschehen ist, kann man endlich das Bit 0 in $fd505 auf »High« legen. Nach dem Abfallen des Systemtaktes der 8502-CPU beginnt nun endlich der Z80 seine Arbeit ab der Adresse $0ffed. Hinter dem dort stehenden »NOP« findet sie einen Sprungbefehl nach $0008 in das nun eingeschaltene ROM. Das dort stehende Programm bootet normalerweise die CP/M-System-Diskette.

Um in eigene Programme zu gelangen, muß man vor dem Einschalten den Sprungbefehl verbiegen. Dazu schreibt man in die Speicherstelle $0ffee ein $c3 (entspricht dem »JP«-Befehl) und in die folgenden beiden Stellen die Startadresse im Low/High-Format.

Listing 1 zeigt ein Programm, das die oben genannten Punkte, außer dem Verändern des Sprungbefehls, für Sie ausführt. Nach dem Abarbeiten des Z80-Programms sollte man nach $0ffe0 springen. Dort wird die 8502-CPU reaktiviert (Näheres in Listing 2). Beide Programme stehen sofort nach Einschalten des Computers zur Verfügung.

Zum Schluß möchte ich noch auf einen Fehler im Betriebssystem hinweisen: Betätigt man, während der 40-Zeichen-Bildschirm gescrollt wird, die ASCII/DIN-Taste, so erscheint sehr oft in etwa der Mitte des Bildschirms ein Zeichensalat. Disassembliert ergeben die ASCII-Werte ein Programm, das auch in der erweiterten Zeropage zu finden ist.

(Frank Probst/tr)

»PRINT AT« oder »CHAR«?

Zunächst möchte ich zu dem Artikel »Entdeckungsreise durch den C 128« (64'er, Ausgabe 12/85) Stellung nehmen: Ein Simulieren des »PRINT AT«-Befehls, wie in diesem Artikel beschrieben, ist nicht nötig: Er ist im Basic 7.0 im Prinzip enthalten. Er wird mit dem »CHAR«-Befehl angesprochen, der nicht nur im Grafik-Modus wirksam ist. Format: char, Spalte, Zeile, Beispiel:

```
10 char,5,5,"hallo" : char,15,15,"....",1
15 char,20,20 : input "eingabe";a
```

Der eingebaute Sprite-Editor des C 128 besitzt außer den im Handbuch aufgeführten Anweisungen noch den Befehl »COPY«. Drückt man in der SPRDEF-Ebene die Taste »C«, so erscheint unten die Frage »COPY FROM ?«. Es wird dann die Nummer des Sprite angegeben (1-8), dessen Bitmuster in das momentan bearbeitete Raster kopiert werden soll. So lassen sich an verschiedenen Sprites geringfügige Änderungen