Tips, Tricks, PEEKs und POKEs zum C 128

Eine wahre Fundgrube für jeden C128-Anwender. Hier finden Sie interessante Listings und Informationen, die Ihnen helfen werden, Ihren C128 besser auszunutzen und kennenzulernen.

Eine wahre Fundgrube für jeden C128-Anwender und -Programmierer haben wir für Sie zusammengestellt. Tips & Tricks-Listings, PEEKs, POKEs und jede Menge Hintergrundinformationen, um Ihnen das Arbeiten mit Ihrem Computersystem zu erleichtern.

Datagenerator C128

Da der Monitor des C128 im Vergleich mit dem SMON keine Möglichkeit besitzt, Maschinenprogramme in DATA-Zeilen zu verwandeln, mußte der gequälte C128-Besitzer immer zwischen C64- und C128-Modus wechseln, nur um eine kleine Routine in Basic-Zeilen abgelegt zu bekommen.

Mit diesem Datagenerator (Listing 1, bitte mit dem neuen Checksummer 128 eingeben) können bequem alle Maschinensprache-Routinen in Basic-Zeilen übernommen werden.

Das Programm verlangt von Ihnen die Eingabe der Start- und Endadresse des zu wandelnden Bereiches. Diese kann sowohl dezimal oder auch, mit vorangestelltem »$«-Zeichen, hexadezimal erfolgen. Zu beachten ist nur, daß die Hex-Zahl vierstellig sein muß.

Haben Sie die beiden Adressen eingegeben, müssen Sie dem Computer noch die Startzeilennummer der DATA-Zeilen und deren Schrittweite mitteilen. Danach wird der Bildschirm gelöscht und die Einleseschleife übernommen. Wiederum löscht sich der Bildschirm und aus dem angegebenen Bereich werden die Werte in DATA-Zeilen abgelegt, und zwar in hexadezimaler Form. Dadurch sind die erzeugten Zeilen nicht nur schön untereinander formatiert, sondern ersparen den späteren Benutzern auch noch ein Drittel Tipparbeit. Letztendlich löscht sich der Datagenerator per DELETE -40 noch selbst, so daß nun beim LISTen ein speicherfertiges Programm auf dem Bildschirm steht.

```
2 REM *** DATAGEN 128 ***                                   <00P>
4 REM *** (C) T. FAHLE ***                                  <00Q>
6 :                                                         <01B>
8 SCNCLR                                                    <0FJ>
10 INPUT "STARTADR.";SA$: IF MID$(SA$,1,1)="
   $" THEN SA=DEC(RIGHT$(SA$,4)): ELSE SA=VA
   L(SA$)                                                   <S5Q>
12 INPUT "ENDADR.";EA$: IF MID$(EA$,1,1)="$"
    THEN EA=DEC(RIGHT$(EA$,4)): ELSE EA=VAL(
   EA$)                                                     <JAL>
14 INPUT "1.ZEILENNR.";ZN: INPUT "SCHRITTWEI
   TE";SW: IF ZN<38 THEN 14                                 <D09>
16 PRINT CHR$(147);ZN;"FORI="SA"TO"EA"                      <0MI>
18 ZN=ZN+SW: PRINT ZN;"READ A$:POKE I,DEC(A$
   )": ZN=ZN+SW: PRINT ZN;"NEXT I"                          <G4H>
20 ZN=ZN+SW: PRINT ZN;":"                                   <IS6>
22 PRINT "GOTO 26"                                          <CMN>
24 POKE 842,19: FOR I=1 TO 5: POKE 842+I,13:
    NEXT : POKE 208,6: END                                  <JV1>
26 ZN=ZN+SW: SCNCLR : PRINT ZN;"DATA";                      <8MF>
28 FOR I=SA TO SA+14                                        <IQT>
30 : PRINT RIGHT$(HEX$(PEEK(I)),2)+",";                     <O4D>
32 NEXT I: PRINT CHR$(20)                                   <8KP>
34 PRINT "SA="SA":EA="EA":SW="SW":ZN="ZN                    <ECK>
36 PRINT "{2DOWN}GOTO38": POKE 842,19: FOR T
   =1 TO 3: POKE 842+T,13: NEXT : POKE 208,5
   : END                                                    <RD1>
38 SA=SA+15: IF SA<EA THEN 26                               <UHL>
40 SCNCLR : PRINT "DELETE-40": POKE 842,19:
   POKE 843,13: POKE 208,2: END                             <LMA>
```

Listing 1. »DATAGEN 128« – erzeugt ein Basic-Ladeprogramm. Bitte verwenden Sie zur Eingabe den Checksummer 128 auf Seite 122.

```
1000 FORI= 4864 TO 4919
1010 READ A$:POKE I,DEC(A$)
1020 NEXT I
1030 :
1040 DATA78,A9,0E,8D,14,03,A9,13,8D,15,03,58,60,78,A5
1050 DATAD5,F0,25,C5,FC,F0,21,85,FC,A9,15,8D,18,D4,A0
1060 DATA09,A2,00,8C,05,D4,8E,06,D4,A9,30,8D,01,D4,A9
1070 DATA20,8D,04,D4,A9,21,8D,04,D4,58,4C,65,FA,A4,FB
```

Listing 2. So könnte ein mit DATAGEN 128 erzeugtes Listing aussehen

Das Programm »Tasten-Piep« (Listing 2) zeigt, wie ein durch »Datagen 128« bearbeitetes Programm aussieht. Die ersten vier Zeilen zeigen die vom Programm erzeugte Einleseschleife, danach folgen die formatierten DATA-Zeilen. Übrigens können Sie dieses kleine Programm ruhig abtippen, denn einen Piepston auf Tastendruck kann man immer gebrauchen, vor allem dann, wenn man mehr oder weniger blind DATA-Zeilen eintippt.

Wichtiger Hinweis:

Sollten Sie das Programm RENUMBERn, müssen Sie auch in den Zeilen 22, 36 und 40 die Werte entsprechend ändern, da der RENUMBER-Befehl GOTO und DELETE innerhalb einer PRINT-Zeile nicht ändert. (Torsten Fahle/dm)

Punkt-Komma-Tausch

Mit diesem kurzen Programm (Listing 3, bitte mit dem neuen Checksummer 128 eingeben) läßt sich der Dezimalpunkt auf der Zehnertastatur durch ein Komma ersetzen. Dies ist sehr hilfreich, falls lange DATA-Kolonnen mittels der Zehnertastatur eingetippt werden müssen.

Der Dezimalpunkt wird nur im ASCII-Modus mit dem Komma belegt. Es ist aber trotzdem möglich, den deutschen Zeichensatz zu nutzen, wenn man wie folgt vorgeht:

Den C128 ausschalten, auf DIN umstellen, das Programm laden und starten. Nach kurzer Zeit wird man aufgefordert, die Taste F8 zu betätigen.

Nun kann damit begonnen werden, Listings einzugeben. Will man nun den Dezimalpunkt als Komma benutzen, was ja meist in DATA-Zeilen der Fall ist (da hier ja sowieso keine deutschen Umlaute benötigt werden), schaltet man auf ASCII-Zeichensatz um und hat nun das Komma zur Verfügung. Soll wieder Text eingegeben werden, wählt man wieder den DIN-Zeichensatz. Der Dezimalpunkt auf der Zehnertastatur ist nun wieder Dezimalpunkt, kann aber durch erneutes Umschalten auf ASCII wieder als Komma genutzt werden.

Soll nur im ASCII-Modus gearbeitet werden, ist das ganze wesentlich weniger kompliziert. Einfach Programm laden und starten!

Vor dem Speichern eines so eingegebenen Programmes sollte erst einmal <RUN/STOP-RESTORE> betätigt werden, um den Ursprungszustand wieder herzustellen.

Programmbeschreibung

In Zeile 85 bis 105 werden die für das Programm notwendigen Adressen definiert. Die nun folgende FOR..NEXT-Schleife erstellt im RAM ab Adresse $1B00 die später benötigte Normaltabelle des ASCII-Codes, da sie dort einfach zu manipulieren ist. Diese Tabelle ist im Commodore-Handbuch des C128 in Anhang J sehr gut versteckt.

In Zeile 125 wird bei Adresse $0AC5 das siebte Bit gesetzt. Dies bewirkt, daß das High-Byte bei Adresse $033F immer mit dem gesetzten Modus (ASCII oder DIN) übereinstimmt, da die Adresse $033F andernfalls neu gesetzt würde.

Die Zeilen 130 bis 135 setzen nun noch die Adreßzeiger, und zwar $033E auf 0 und $033F auf 27.

Zeile 140 ist eigentlich der komplizierteste Teil. Hier findet der eigentliche Tausch der Tastaturbelegung statt. Sie erhalten nun die Methode gezeigt, mit der sich Tasten nahezu beliebig umbelegen lassen. Es folgt nun eine genaue Erklärung des S. Kohlitz-Tasttausch-Systems:

Man hat durch die FOR..NEXT-Schleife einen ASCII-Zeichensatz im RAM, der bei Adresse 6912 ($1B00) beginnt. Will man nun eine Taste umbelegen, so addiert man zu dem Wert 6912 zuerst den Platz-Code aus der Tastaturtabelle des zu ändernden Zeichens minus 1 (im Falle des Dezimalpunktes also 83-1). Warum denn minus 1? Ganz einfach: Die Platzcodes sind von 1 bis 89 fortlaufend durchnumeriert. Um aber sinnvoll damit arbeiten zu können, sollten sie von 0 bis 88 numeriert sein. Man sagt ja auch acht Bit, die aber von 0 bis 7 durchnumeriert sind.

Wir haben nun den Wert 6912+82 errechnet, also 6994. In diese Speicherstelle POKEt man nun nur noch den Code (nicht den Platzcode) des Zeichens, mit der die Taste belegt werden soll (im Falle des Kommas also eine 44). POKE 6994,44 und der Tausch ist perfekt.

Wollen Sie jetzt noch weitere Tasten umbelegen (wie wäre es mit SHIFT anstatt der Minus- oder Plus-Taste, um die Funktionstasten von der Zehnertastatur leichter zu erreichen?), so dürften der Ausführung keine Probleme mehr im Wege stehen.

(Stephan Kohlitz/dm)

```
80 RESTORE                                          <79M>
85 Z1=DEC("0058"): Z2=DEC("1B00")                   <Q9Q>
90 Z3=DEC("0AC5"): Z4=DEC("0080")                   <8TV>
95 Z5=DEC("033E"): Z6=DEC("033F")                   <UHV>
100 Z7=DEC("001B"): Z8=DEC("1B52")                  <CAL>
105 Z9=DEC("002C")                                  <N1E>
110 FOR DC=0 TO Z1 : READ DA                        <NLK>
115 POKE Z2+DC, DA                                  <FN5>
120 NEXT DC                                         <5B5>
125 POKE Z3,PEEK(Z3) OR Z4                          <UN7>
130 POKE Z5,0                                       <KQE>
135 POKE Z6,Z7                                      <8V1>
140 POKE Z8,Z9                                      <A6U>
145 DATA 20,13,29,136,133,134,135,17                <0F9>
150 :                                               <043>
155 DATA 51,87,65,52,90,83,69,1                     <EIU>
160 :                                               <045>
165 DATA 53,82,68,54,67,70,84,88                    <1J4>
170 :                                               <047>
175 DATA 55,89,71,56,66,72,85,86                    <NQJ>
180 :                                               <04P>
185 DATA 57,73,74,48,77,75,79,78                    <17Q>
190 :                                               <04R>
195 DATA 43,80,76,45,46,58,64,44                    <OLP>
200 :                                               <045>
205 DATA 92,42,59,19,1,61,94,47                     <UIR>
210 :                                               <047>
215 DATA 49,95,4,50,32,2,81,3                       <EO7>
220 :                                               <041>
225 DATA 132,56,53,9,50,52,55,49                    <JVS>
230 :                                               <043>
235 DATA 27,43,45,10,13,54,57,51                    <D4P>
240 :                                               <04D>
245 DATA 8,48,46,145,17,157,29,255                  <GPP>
250 :                                               <04F>
255 DATA 255                                        <ALV>
260 KEY 8,"DELETE-300"+CHR$(13)                     <C2U>
270 PRINT CHR$(147);"PUNKT-->KOMMA TAUSCH           <NNF>
280 PRINT : PRINT "(C) STEPHAN KOHLITZ 1985"        <K4A>
290 PRINT : PRINT "VOR WEITEREN EINGABEN 'F8
    ' DRUCKEN"                                      <IDC>
```

Listing 3. Ein Programm zum Austauschen des Punktes mit einem Komma auf der Zehnertastatur

Get mit Cursor

Bei allen Commodore-Computern konnte man mit drei kleinen POKEs den Cursor beim GET-Befehl ein- und ausschalten. Beim C128 geht es im Prinzip auch, nur nicht im 80-Zeichen-Modus. Doch nichts ist unmöglich.

Im 80-Zeichen-Modus kann man dem VDC (Videocontroller) ein Zeichen mit einem »Blink-Attribut« versehen übergeben. Wie man anhand des Programms (Listing 4, bitte mit dem neuen Checksummer 128 eingeben) ausprobieren kann, blinkt der Cursor jetzt bei der GET-Routine, egal, ob im 40- oder 80-Zeichen-Modus.

Das Programmsegment für den 80-Zeichen-Modus dürfte durch die reichlichen Kommentare verständlich sein. Beim 40-Zeichen-Modus sind folgende zwei angesprochenen Speicherstellen von Interesse:

2599 $0A27 BLNSW Schalter für den Cursor (204 beim C64); 0 = Ein/1 = Aus

2600 $0A28 BLNCT Zähler für Cursorblinken (205 beim C64)

Den Cursorblinkzähler sollte man nach dem Holen eines Zeichens auf 1 stellen, da sonst hin und wieder ein inverser Block irgendwo auf dem Bildschirm stehenbleibt.

(Michael Bauer/dm)

```
100 REM ************************************ <02A>
110 REM *                                    * <02B>
120 REM *    GET MIT BLINKENDEM CURSOR       * <028>
130 REM *    FUER C128 MIT  40 ODER 80       * <029>
140 REM *         ZEICHEN-SCHIRM             * <02E>
150 REM *                                    * <02F>
160 REM ************************************ <02C>
200 PRINT "EINGABE ? ";                      <904>
210 GOSUB 310: IF X$<>"J" AND X$<>"N" THEN 2
    10                                       <M7J>
220 PRINT X$                                 <ATI>
230 END                                      <0KU>
250 REM ***************** SUBROUTINE GET      <029>
260 REM ***************** MIT BLINKENDEM      <02A>
270 REM ***************** CURSOR              <02B>
290 REM PRUEFE OB 40 ODER 80 ZEICHEN-SCHIRM  <025>
310 IF PEEK(215)=128 THEN BEGIN :   REM 80 Z
    EICHEN                                   <IBH>
330 : REM CHR$(15)  = ZEICHENBLINK AN        <047>
340 : REM CHR$(18)  = RVS ON                 <049>
350 : REM CHR$(146) = RVS OFF                <04B>
360 : REM CHR$(157) = CURSOR LINKS           <04D>
370 : REM CHR$(143) = ZEICHENBLINK AUS       <04F>
390 :  PRINT CHR$(15) CHR$(18);" ";CHR$(146)
     CHR$(157) CHR$(143);                    <DUV>
410 :  GET KEY X$:     REM WARTE AUF TASTEND
    RUCK                                     <09K>
430 BEND : ELSE BEGIN : REM 40 ZEICHEN       <V4R>
450 :  POKE 2599,0: REM SCHALTE BLINKEN EIN  <PSD>
470 :  GET KEY X$:  REM WARTE AUF TASTENDRUC
    K                                        <69K>
```

Listing 4. Nun blinkt der Cursor auch bei GET. Bitte beachten Sie den Checksummer 128 auf Seite 122.

Bildschirme verwalten

Dieses Programm macht es möglich, bis zu drei Bildschirme gleichzeitig im 80-Zeichen-Modus zu verwalten. Die einzelnen Bildschirme können nach dem Starten des Basic-Laders mit SYS 2883, Sichtbar, Bildschirmnummer, Attribut aufgerufen werden. Dabei haben die Parameter folgende Bedeutung:

- Sichtbar: Sichtbar kann zwei verschiedene Werte annehmen.

Name	Nummer	Adresse	Funktion
FSTMODE	01	$ff47	Fastimpuls an Bus
EAINIT	02	$ff4a	Standard E/A-Geräte setzen
C64MODE	03	$ff4d	Aktivierung des 64'er-Modus
DMA-CALL	04	$ff50	Externe RAM-Bausteine initialisieren
BOOT-CALL	05	$ff53	Booten der Diskette
PHOENIX	06	$ff56	Kaltstart
LKUPLA	07	$ff59	Suche nach logischer Filenummer
LKUPSA	08	$ff5c	Suche nach Sekundäradresse
SWAPPER	09	$ff5f	Umschalter zwischen 40 und 80 Zeichen
DLCHR	10	$ff62	Kopiert Zeichensatz ins VDC-RAM
PFKEY	11	$ff65	F-Tastenbelegung
SETBANK	12	$ff68	Speicherbank für LOAD/SAVE definieren
GETCONF	13	$ff6b	Holt Konfigurationsbyte
JSRFAR	14	$ff6e	JSR in beliebige Bank
JMPFAR	15	$ff71	JMP in beliebige Bank
FETCH	16	$ff74	LDA ( ),Y aus beliebiger Bank
STASH	17	$ff77	STA ( ),Y in beliebiger Bank
COMPARE	18	$ff7a	CMP ( ),Y in beliebiger Bank
PRIMM	19	$ff7d	Textausgabe
CINT	20	$ff81	Editor-Initialisierung
IOINIT	21	$ff84	Ein-/Ausgabe-Initialiesierung
RAMTAS	22	$ff87	Warmstart
RESTOR	23	$ff8a	Systemvektoren initialisieren
VECTOR	24	$ff8d	Systemvektoren kopieren
SETMSG	25	$ff90	Systemmeldungen ermöglichen/ verhindern
SECND	26	$ff93	Sekundäradresse nach LISTn
TKSA	27	$ff96	Sekundäradresse nach TALK
MEMTOP	28	$ff99	Lesen/setzen der Speicherobergrenze
MEMBOT	29	$ff9c	Lesen/setzen der Speicheruntergrenze
KEY	30	$ff9f	Tastaturabfrage
SETTMO	31	$ffa2	Setzen des Timeoutflags für IEEE
ACPTR	32	$ffa5	Holt Byte vom Bus
CIOUT	33	$ffa8	Legt Byte auf Bus
UNTLK	34	$ffab	Untalk an Bus senden
UNLSN	35	$ffae	Unlisten an Bus senden
LISTN	36	$ffb1	Listen an Bus senden
TALK	37	$ffb4	Talk an Bus senden
READST	38	$ffb7	Statusbyte lesen
SETLFS	39	$ffba	Setzen der Fileparameter
SETNAM	40	$ffbd	Setzen des Filenamens
OPEN	41	$ffc0	Datei öffnen
CLOSE	42	$ffc3	Datei schließen
CHKIN	43	$ffc6	Datei ist Eingabedatei
CKOUT	44	$ffc9	Datei ist Ausgabedatei
CLRCH	45	$ffcc	Ein-/Ausgabekanäle schließen
BASIN	46	$ffcf	Zeicheneingabe
BSOUT	47	$ffd2	Zeichenausgabe
LOAD	48	$ffd5	Laden einer Datei
SAVE	49	$ffd8	Speichern einer Datei
SETTIM	50	$ffdb	Setzen der internen Uhr
RDTIM	51	$ffde	Lesen der internen Uhr
STOP	52	$ffe1	Stop-Tasten-Abfrage
GETIN	53	$ffe4	Holt Zeichen aus Tastaturpuffer
CLALL	54	$ffe7	Schließen aller Dateien
UDTIM	55	$ffea	Interne Uhr hochzählen
SCRORG	56	$ffed	Fenstergröße feststellen
PLOT	57	$fff0	Cursorposition holen/setzen
IOBASE	58	$fff3	Holt die Basisadresse des I/O-Bereiches

Tabelle 1. Die wichtigsten Kernel-Routinen und ihre Funktion

Wert 0: Der vorherige Bildschirm wird angezeigt. Die Befehle PRINT und CHAR schreiben aber schon auf den neuen Bildschirm. Dadurch kann ein Bildschirm aufgebaut werden, während ein anderer angezeigt wird.

Wert 1: Der angewählte Bildschirm wird auch tatsächlich angezeigt.

- Bildschirm: Es stehen die Bildschirme 1 bis 5 zur Verfügung. Bitte lassen Sie sich nicht verwirren, die Bildschirme 1 und 3 sind die gleichen.

- Attribut: Auch für das Attribut gibt es zwei Werte:

Wert 0: Attribut wird ausgeschaltet. Das heißt, auf dem Bildschirm sind keine Farben, blinkende und unterstrichene Zeichen mehr sichtbar. Außerdem werden alle Zeichen im Grafikmodus angezeigt. Das alles gilt auch, wenn Sie den Bildschirm nicht sichtbar einschalten. Beim Beschreiben des Bildschirms wird aber auch das abgeschaltete Attribut-RAM beschrieben. Diese Funktion (Attribut aus) ist für die Bildschirme 3, 4 und 5 gedacht, für die nur ein Attribut-RAM zur Verfügung steht.

Wert 1: Attribut wird eingeschaltet.

Insgesamt können fünf Bildschirme erzeugt werden. Die verschiedenen Bildschirme werden durch Verschiebung des Video-RAMs (in ihm stehen die Bildschirmcodes der Zei-

```
10 : REM BILDSCHIRME                                          <02F>
20 :                                                          <029>
30 : REM MASCHINENSPRACHEDATEN EINLESEN                       <02B>
40 : BANK 0: FAST                                             <RQI>
50 : FOR I = DEC("B00") TO DEC ("B91")                        <58T>
60 :  READ DA$: POKE I, DEC(DA$)                              <IMQ>
70 :  SU = SU + DEC(DA$)                                      <LV5>
80 : NEXT I                                                   <204>
90 : IF SU <> 11931 THEN PRINT "{CTRL+O}FEHL
   ER IN DATAZEILEN": END : ELSE BEGIN                        <4UV>
100 :  PRINT "{CTRL+O}DATAZEILEN SIND RICHTI
    G.": SLEEP  : BEND                                        <V8P>
110 : REM DIE STEUERZEICHEN IN ZEILE70/80 SI
    ND < CONTROL O >                                          <04B>
120 BANK 15: SYS DEC("C07B"): REM BILDSCHIRM
     INITIALISIEREN                                           <CSF>
130 :                                                         <04F>
140 : REM ASSEMBLERDATEN                                      <041>
150 DATA 8E,00,D6,2C,00,D6,10,FB,8D,01,D6,60
    ,8E,00,D6,2C                                              <KL6>
160 DATA 00,D6,10,FB,AD,01,D6,60,A9,00,E0,01
    ,D0,07,A2,0C                                              <5IG>
170 DATA 20,00,0B,A2,01,8D,2E,0A,60,A9,08,E0
    ,01,D0,05,A2                                              <LOK>
180 DATA 14,20,00,0B,8D,2F,0A,60,20,0C,0B,09
    ,40,20,00,0B                                              <57F>
190 DATA 4C,53,0B,48,8A,48,A2,19,98,D0,ED,20
    ,0C,0B,29,BF                                              <MOG>
200 DATA 20,00,0B,68,C9,01,F0,04,C9,03,D0,08
    ,68,AA,20,18                                              <ASI>
210 DATA 0B,4C,29,0B,C9,02,D0,0C,68,AA,A9,10
    ,20,1A,0B,A9                                              <Q2L>
220 DATA 18,4C,2B,0B,C9,04,D0,0A,68,AA,A9,10
    ,20,1A,0B,4C                                              <DU2>
230 DATA 29,0B,C9,05,D0,0A,68,AA,A9,18,20,1A
    ,0B,4C,29,0B                                              <81Q>
240 DATA 68,60                                                <BFM>
250 :                                                         <04F>
260 : REM MIT SYS2883,SICHTBAR,BILDSCHIRM,AT
    TRIBUT WERDEN DIE BILDSCHIRME                             <049>
270 : REM EINGESCHALTET. SICHTBAR KANN DIE W
    ERTE 0 (NICHT SICHTBAR) UND 1 (SICHT-                     <04B>
280 : REM BAR) ANNEHMEN. ICHT SICHTBAR HEISS
    T, DASS FUER DIE BEFEHLE PRINT UND                        <04L>
290 : REM CHAR AUF DEN NEUEN BILDSCH. ZUGEGR
    IFFEN WIRD, ABER GLEICHZEIT. DER ALTE                     <04N>
300 : REM BILDSCH. ANGEZEIGT WIRD. BILDSCHIR
    M KANN DIE WERTE VON 1-5 ANNEHMEN.                        <041>
310 : REM ATTRIBUT KANN DIE WERTE 0 (AUS) UN
    D 1 (EIN) ANNEHMEN.                                       <043>
320 : REM DER BEFEHL IN ZEILE 120 ZUM INITIA
    LISIEREN DES BILDSCHIRMS IST NOTWEN-                      <045>
330 : REM DIG UM DAS PRG. BILDSCHIRME FEHLER
    FREI ABLAUFEN ZU LASSEN. ER MUSS EIN-                     <047>
340 : REM MAL NACH DEM POKEN DER MASCHINENSP
    RACHEDATEN AUSGEFUEHRT WERDEN.                            <049>
```

Listing 5. Mehrere Bildschirme austauschen

chen, die auf dem Bildschirm sichtbar sind), und durch Verschieben beziehungsweise Abschalten des Attribut-RAMs (in ihm werden die Farben und alternativen Darstellungen eines Zeichens eingetragen) des VDC-RAM (der VDC besitzt ein eigenes, 16 KByte großes RAM) realisiert.

Dabei stehen folgende Bildschirme zur Verfügung:

Bildschirm 1 Video-RAM von $0000 bis $0800, Attribut-RAM von $0800 bis $1600

Bildschirm 2 Video-RAM von $1600 bis $2400, Attribut-RAM von $2400 bis $3200

Bildschirm 3 Video-RAM von $0000 bis $0800, Attribut-RAM von $0800 bis $1600

Bildschirm 4 Video-RAM von $1600 bis $2400, Attribut-RAM von $0800 bis $1600

Bildschirm 5 Video-RAM von $2400 bis $3200, Attribut-RAM von $0800 bis $1600

Daraus ergibt sich die folgende Aufteilung: Bildschirm 1 und 2 oder Bildschirm 3, 4 und 5 können zusammen benutzt werden, ohne daß Verluste des Bildschirminhalts auftreten. Bei den Bildschirmen 3, 4 und 5 wird dasselbe Attribut-RAM benutzt. Für eben diese Schirme steht dann jedoch nur ein Attribut-RAM zur Verfügung.

Eingabehinweise

Das Programm (Listing 5) geben Sie bitte mit dem neuen Checksummer 128 ein und speichern es. Das Programm starten Sie durch RUN. Das zweite Programm (Listing 6) ist ein Demo-Programm, das nicht unbedingt abgetippt werden muß.

Der Befehl SYS DEC ("C07B") im Basic-Lader ist nötig, um den Bildschirm zu initialisieren. Wird dieser Befehl nicht ausgeführt, so kann es geschehen, daß sich der Computer nach dem Wechsel eines Bildschirms nicht mehr meldet.

```
10 FAST : EIN=1: AUS=0                                          <D98>
11 :                                                            <02E>
20 BS=2: SYS 2883,EIN,BS,EIN: PRINT "{CLR}":
          REM BILD 2 LOESCHEN U EINSCHALTEN                     <5K1>
30 BS=1: SYS 2883,AUS,BS,EIN: PRINT "{CLR}":
     REM BILD 1 LOESCHEN U NICHT SICHTB. EIN                    <9LM>
31 :                                                            <02A>
40 FOR I=1 TO 10                                                <JSV>
50 PRINT "{CTRL+B}CHRISTOPH FRANZEN BONIFATI
   USSTRASSE 70 4130 MOERS ASBERG BILDSCHIRM
    1                                                           <SCS>
60 NEXT                                                         <0P0>
61 :                                                            <020>
70 BS=1: SYS 2883,EIN,BS,EIN:
                 REM BILD I EINSCHALTEN                         <VAA>
80 INPUT "BITTE GEBE DEINEN NAMEN EIN"; NA$                     <TLH>
81 PRINT "{CLR,12SPACE}HALLO "NA$"{2DOWN}                       <195>
82 PRINT "ICH HOFFE ES GEHT DIR HEUTE SEHR G
   UT. JEDENFALLS WIRD JETZT GERADE WAEHREND                    <K8F>
83 PRINT "DU DIESEN TEXT LIESST DEIN NAME AU
   F BILDSCHIRM 2 BLINKEND GESCHRIEBEN.                         <K07>
84 PRINT "WENN DU GLEICH DIE AUFFORDERUNG BE
   KOMMST EINE TASTE ZU DRUECKEN, DANN IST                      <INK>
85 PRINT "DER AUFBAU DES BILDSCHIRMS 2 FERTI
   G UND DU KANNST IHN DIR ANSEHEN, INDEM DU                    <T1D>
86 PRINT "DER AUFFORDERUNG NACHKOMMST UND EI
   NE BELIEBIGE TASTE DEINES COMPUTERS                          <S7E>
87 PRINT "DRUECKST.                                             <8DS>
89 :                                                            <02K>
90 BS=2: SYS 2883,AUS,BS,EIN  :
         REM BILD 2 NICHT SICHTB. EIN                           <QEI>
100 PRINT "{CLR,CTRL+B}CHRISTOPH FRANZEN BON
    IFATIUSSTRASSE 70 4130 MOERS ASBERG BILD
    SCH.2                                                       <US5>
110 FOR I=0 TO 100: PRINT " {CTRL+O}" NA$ "
    "; : NEXT                                                   <BAR>
111 :                                                           <04A>
120 BS=1: SYS 2883,EIN,BS,EIN:
          REM BILD 1 WIEDER EINSCHALTEN                         <6IA>
130 CHAR 1,20,20, "BITTE EINE TASTE DRUECKEN
    "                                                           <IQ2>
140 GET KEY A$                                                  <KCD>
141 :                                                           <040>
150 BS=2: SYS 2883,EIN,BS,EIN:
       REM BILD 2 ZUM SEHEN EINSCHALTEN                         <7AI>
```

Listing 6. Ein Demo-Programm, das Ihnen die Fähigkeiten des Bildschirm-Vertauschens aufzeigt

Dann müssen Sie das Programm mit RUN/STOP-RESTORE unterbrechen. (Christoph Franzen/dm)

Programme verbinden (MERGE)

Mit diesem Programm (Listing 7, bitte mit dem neuen Checksummer 128 eingeben) lassen sich mehrere Programme zu einem einzigen verbinden.

Das Programm »Merge« ist so geschrieben, daß es an einen freien Speicherplatz gesetzt und aufgerufen werden kann. Um das Programm an eine andere Adresse zu verschieben, muß im Basic-Lader in Zeile 30 die Variable »AN« geändert werden.

```
10 REM MERGE                                                    <019>
20 :                                                            <029>
30 AN = 2816 : BANK 0                                           <3NT>
40 :                                                            <025>
50 REM MASCHINENSPRACHEDATEN EINLESEN                           <01D>
60 FOR I = AN TO AN + 30                                        <G9H>
70 READ DA$: POKE I, DEC(DA$)                                   <KAQ>
80 SU = SU + DEC(DA$)                                           <5VT>
90 NEXT                                                         <0UG>
100 IF SU <> 2733 THEN PRINT "FEHLER IN DATA
    ZEILEN": END : ELSE BEGIN                                   <L78>
110 PRINT "PROGRAMMSTART AN PROGRAMMENDE SET
    ZEN MIT SYS" AN                                             <7S3>
120 PRINT "PROGRAMME ZU EINEM PROGRAMM ZUSAM
    MENSETZEN MIT SYS" AN+22                                    <UE1>
130 :                                                           <04F>
140 REM MASCHINENSPRACHEDATEN                                   <02E>
150 DATA AD,10,12,38,E9,02,85,2D,AD,11,12,E9
    ,00,85,2E,A0                                                <PSH>
160 DATA 02,A9,00,91,2D,60,A9,01,85,2D,A9,1C
    ,85,2E,60                                                   <9LS>
170 :                                                           <047>
180 REM ENDE                                                    <022>
```

Listing 7. »Merge« – Verbinden mehrerer Basic-Programme

Die Bedienung der Routine:

1) Reset-Taster am Computer betätigen beziehungsweise Computer aus- und einschalten. Das ist wichtig, um sicherzugehen, daß der Basic-Start wirklich bei $1C00 beginnt.

2) Programm MERGE laden und starten. Bitte schreiben Sie sich die Adressen der SYS-Befehle auf.

3) Laden Sie Ihr Programm, das Sie mit einem anderen verbinden wollen, und bringen Sie das Programm eventuell durch RENUMBER in einen vernünftigen Zeilennummernbereich. Schreiben Sie sich bitte die höchste Zeilennummer des ersten Programmes auf.

4) Verschieben Sie den Basic-Start (SYS 2816).

5) Laden Sie das nächste Programm und bringen auch dieses durch RENUMBER in einen vernünftigen Zeilennummernbereich (die Zeilennummern des zweiten Programms müssen höher als die des ersten sein).

6) Verbinden Sie die Programme (SYS 2838).

7) Nun können Sie das neue Programm speichern.

Nach dem Verschieben des Basic-Starts sind alle Befehle erlaubt. Das heißt, Sie können die geladenen Programme mittels DELETE, RENUMBER oder sonstigen Befehlen verändern. (Christoph Franzen/dm)

Bankswitching in Assembler

Durch das Bankswitching wird die Assemblerprogrammierung oft sehr aufwendig und mühselig. Wer schon einmal versucht hat, aus BANK 1 in eine Kernel-Routine in BANK 15 zu springen, wunderte sich sicher, denn danach befand sich der Computer in einem (sehr) undefinierten Zustand.

Um diesem Problem zu begegnen, müßte ein Programm her, das vor Aufruf die gewünschte Konfiguration einschaltet und bei der Rückkehr die alte Speicherbelegung wiederherstellt.

Das Programm müßte zusätzlich in der Common-Area (jenem Speicherbereich, der für alle BANKs gleich ist) funktionieren. Es bietet sich der RS 232-Eingabepuffer von $C00 bis $D00 an.

Das Programm »Kernel« (Listing 8), das in diesem Bereich liegt, funktioniert so:

1) Die Nummer der Kernel-Routine wird in 996 gespeichert (die Nummern entnehmen Sie bitte Tabelle 1)

2) Akku, X- und Y-Register so mit ihren Werten laden, als würde direkt in die gewünschte Kernel-Routine verzweigt

3) JSR $C00

Manchmal kommt es auch vor, daß ein Unterprogramm im Basic-ROM von $F4000 bis $FAFFF oder im Monitor von $FB000 bis FBFFF aufgerufen werden soll. Solch ein Problem ist mit JSRFAR zu lösen:

1) Die anzuspringende BANK ist in $02 abzulegen
2) Die anzuspringende Adresse ist im High-/Low-Format in $03/$04 bereitzustellen
3) Akku in $06, X-Register in $07 und Y-Register in $08
4) JSR $C23

Soll kein Unterprogrammaufruf, sondern ein normaler Sprung stattfinden, so ist Punkt 4 durch JMP $CD0 zu ersetzen.

Diese seltsame Art der Parameterübergabe entspricht den original JMP- und JSRFAR-Routinen. Es dürfte also keine großen Probleme bei der Umrüstung bereits bestehender Programme geben.

»Kernel 128« muß vor dem Start noch initialisiert werden. Dies erfolgt durch JSR $C3E. Findet der Interpreter während der Ausführung eines Basic-Programms eine USR(X)-Anweisung, so wird nach $0F000 gesprungen.

(Frank Probst/dm)

Wichtiger Hinweis zum Abtippen

Bitte beachten Sie den neuen Checksummer 128. Mit diesem Programm können Sie alle C128-Basic-Listings eingeben. Die Prüfsumme am Ende jeder Basic-Zeile unterrichtet Sie sicher, ob Sie die Zeile richtig eingegeben haben. Den Checksummer 128 finden Sie in dieser Ausgabe.

```
name : kernel 128        0c00 0d10
-----------------------------------
0c00 : 85 06 86 07 84 08 a9 0f  58
0c08 : 85 02 a9 44 85 04 a9 ff  a0
0c10 : 85 03 ad e4 03 c9 14 90  0f
0c18 : 02 e6 04 0a 6d e4 03 65  a5
0c20 : 04 85 04 ad 00 ff 48 a9  12
0c28 : 00 8d 00 ff 20 cd 02 68  38
0c30 : 8d 00 ff a5 05 48 a5 06  a7
0c38 : a6 07 a4 08 28 60 a9 00  b8
0c40 : 8d 00 ff ad 06 d5 09 05  c0
0c48 : 8d 06 d5 a9 00 85 2f 85  77
0c50 : 31 a9 10 85 30 85 32 a9  56
0c58 : 72 8d 19 12 a9 0c 8d 1a  7f
0c60 : 12 a9 7f 8d 00 ff a0 37  c9
0c68 : b9 d8 0c 99 50 ff 88 10  0b
0c70 : f7 60 a9 3f 8d 00 ff 4c  5b
0c78 : 00 f0 8d 8d 0c 8d 92 0c  95
0c80 : ee 92 0c ad 00 ff 48 a9  e5
0c88 : 7e 8d 00 ff a5 00 8d 5a  12
0c90 : ff a5 00 8d 5b ff 20 50  ea
0c98 : ff 8d e4 03 68 8d 00 ff  ea
0ca0 : ad e4 03 60 8d e4 03 a9  ec
0ca8 : ff 8d bc 0c 8d c1 0c ee  13
0cb0 : c1 0c ad 00 ff 48 a9 7e  c9
0cb8 : 8d 00 ff a5 00 8d 7c ff  58
0cc0 : a5 00 8d 7d ff ad e4 03  7f
0cc8 : 20 6c ff 68 8d 00 ff 60  c5
0cd0 : a9 00 8d 00 ff 4c e3 02  d3
0cd8 : 78 ad 06 d5 29 f3 8d 06  d8
0ce0 : d5 b9 ff ff 8d f0 ff ad  4d
0ce8 : 06 d5 09 04 8d 06 d5 ad  57
0cf0 : f0 ff 58 60 8d f0 ff 78  53
0cf8 : ad 06 d5 29 f3 8d 06 d5  b2
0d00 : ad f0 ff 99 ff ff ad 06  1b
0d08 : d5 09 04 8d 06 d5 58 60  46
```

Listing 8. »Kernel 128« - Bitte mit dem MSE im C64-Modus eingeben

Boot-Sektoren selbst erstellen

Die Wirkung eines Boot-Sektors kennt wohl jeder Besitzer eines C128. Doch eine Diskette mit diesem Boot-Sektor (Listing 9) versehen, ist dagegen ganz etwas anderes.

Selbst mit dem »Auto-Boot-Creater« von der Test-Demo-Diskette kann leider nur ein Boot-Sektor mit Autostart für ein Maschinen- oder ein Basic-Programm erzeugt werden.

Oder man studiert die entsprechenden Kapitel in den entsprechenden Fachbüchern, und schreibt den Boot-Sektor mit Floppy-Befehlen oder einem Disketten-Monitor (Wer hat den schon für den C128?) auf die Diskette. Aber versuchen Sie dann einmal, das so geladene Basic-Programm im Speicher zu verändern! Oder lieber nicht, denn absolut geladene Basic-Programme lassen sich nicht mehr editieren. Doch das steht ja schon im »Commodore-128-Handbuch« von Markt & Technik.

Doch Gott sei Dank gibt es noch vernünftige Boot-Maker wie den hier vorgestellten.

Hiermit lassen sich leicht und locker 17 (in Worten: siebzehn) verschiedene Boot-Kombinationen erstellen!

Zunächst einmal kann man frei wählbaren Text während des Ladens auf den Bildschirm geben. Hierbei sind sogar Steuerzeichen möglich. Oder man gibt schlicht den Namen des Programms ein, das gerade geladen wird.

Das zu ladende Programm kann mit oder ohne Autostart sein. Natürlich kann man auch nur einen Begrüßungstext auf den Bildschirm ausgeben. Oder nur ein Maschinen- oder Basic-Programm laden.

Aus den drei Grundeinstellungen

1) Text nein / ja
2) Masch.Prg. nein / ja, mit Autostart
   ja, ohne Autostart
3) Basic-Prg. nein / ja, mit Autostart
   ja, ohne Autostart

ergeben sich die oben genannten 17 Boot-Versionen.

»Boot-Sektor« wurde auf einem 80-Zeichen-Montior erstellt, aber auch auf einem 40-Zeichen-Monitor bleibt die Übersichtlichkeit erhalten. Hierfür wird vom Programm automatisch festgestellt, welcher der Bildschirme gerade eingeschaltet ist.

Selbstverständlich läuft das Programm nicht nur mit den Floppy-Laufwerken 1570/1571, sondern auch mit der guten, alten 1541.

Zum Ausprobieren nimmt man am besten eine frisch formatierte Diskette mit je einem Testprogramm in Basic und Maschinensprache. Daß die Floppy 1571 beidseitig formatierte Disketten schneller liest, hat sich sicher schon herumgesprochen; es entfallen dann nämlich einige vergebliche Leseversuche. Aber das nur nebenbei.

Doch nun zur eigentlichen Beschreibung des Programms (Listing 9), das Sie übrigens mit dem neuen Checksummer 128 eingeben müssen:

Zeile 100 löscht den Bildschirm und aktiviert die Fehlerfalle. Hiermit soll vor allem ein Programmabbruch durch versehentliches Drücken der <STOP>-Taste oder falsche HEX-Eingabe (in Zeile 530) vermieden werden.

Es folgen in den Zeilen 120 bis 150 Initialisierung und Test auf Schreibschutz.

Mit Zeile 160 bis 200 wird der Sektor 0 in Spur 1 anhand der ersten 3 Byte geprüft. Je nach Ergebnis kann nach einer Information laut Zeile 210 oder Zeile 250 das Programm in Zeile 290/320 abgebrochen oder fortgesetzt werden. Hier kommen übrigens die sehr praktischen, für C64-Aufsteiger neuen Basic-Anweisungen GETKEY und IF..THEN..ELSE zur Anwendung.

Haben die ersten 3 Byte den Wert 00, ist der Block frei und es wird von Zeile 190 direkt nach Zeile 340 gesprungen.

```
10 REM : : :
```

Zeile 400: Soll ein Maschinen-Programm geladen werden, kann jetzt der Name des Programms eingegeben werden. Kein Maschinen-Programm? <RETURN> drücken!

Wurde ein Name eingegeben (MP$), muß nun die Frage »Autostart?« mit J(a) oder N(ein) beantwortet werden (Zeilen 420 bis 480).

Zeile 470: Die Variable RU ist für die Einsprungadresse zum eventuellen Laden eines Maschinen-Programms wichtig, und ergibt sich aus der Länge des Textes (TX$), der Länge des Maschinenprogramm-Namens (MP$) und dem Wert von JN=19 plus 3 für einen JSR-Befehl, falls Autostart gewählt wurde.

In Zeile 490/500 wird wieder für die PRINT-Anweisung der aktuelle Bildschirm abgefragt. Falls das Maschinen-Programm nicht automatisch gestartet werden soll, geht es von Zeile 490 direkt nach 650 zur Frage nach einem Basic-Programm. Ansonsten muß nun erst noch die Startadresse genannt werden (Zeilen 530 bis 630). Wer die Adresse nicht in HEX-Werten weiß, kann selbstverständlich auch die SYS-Adresse in dezimal angeben; man braucht dann nur N(ein) mit <RETURN> zu bestätigen. Wer unmögliche Werte eingibt, TRAPt in die Fehlerfalle und darf die Eingabe wiederholen.

Die Zeilen 620/630 zerlegen die Adresse in Low- (JL) und High-(JH)Byte.

Nun wird (Zeile 650) nach einem Basic-Programm gefragt. Soll kein Basic-Programm geladen werden, nur <RETURN> drücken. Andernfalls muß der Programmname eingegeben werden (BP$).

Entsprechend der Entscheidung: Autostart – ja oder nein wird der Variablen BS$ der Wert RUN für Laden und Starten oder DLOAD für Laden ohne Starten übergeben (Zeilen 670 bis 710).

Sollte weder ein Text noch ein Programmname eingegeben worden sein, wäre ein Boot-Sektor überflüssig und das Programm wird beendet.

Im Abschnitt der Zeilen 770 bis 830 wird mit GETKEY OK$ angehalten und die Eingaben können in aller Ruhe überprüft werden. Durch X bricht das Programm ab, mit N können laut Zeile 340 nach Löschung aller Variablen (CLR) alle Eingaben erneuert werden.

Ist alles in Ordnung, muß nur noch J gedrückt werden. Nachdem den Variablen CB$ und NB$ die Werte CBM und 00 zugewiesen wurden, läuft der eigentliche Schreibvorgang des Boot-Sektors in den Zeilen 880 bis 1010 ab. In Zeile 1010 wird mit TRAP die Fehlerfalle desaktiviert. Zeile 1020 schafft mit ESCAPE <Klammeraffe> Platz für die Floppy-Meldung DS$.

Fehlermeldungen von der Floppy werden in den Zeilen 1040 bis 1060 und Basic-Fehler in den Zeilen 1080 bis 1120 verarbeitet. (Hans-Georg Schmidt/dm)

Variablen- und Array-Inhalte anzeigen

1): DUMPV 128

Dieses Programm (Listing 10, bitte mit dem neuen Checksummer 128 eingeben) ermöglicht die Ausgabe der Werte der nichtindizierten Variablen eines Basic-Programms.

Nach Start des Ladeprogramms mit RUN erfolgt der Aufruf mit SYS DEC ("0B00"). Es werden für jede Variable deren Name (maximal die ersten beiden Buchstaben), deren Typ (SPACE für Real, % für Integer und $ für String) sowie deren Werte ausgegeben.

2): DUMPA 128

Dieses Programm (Listing 11, bitt ebenfalls mit dem neuen Checksummer 128 eingeben) funktioniert ähnlich wie DUMPV 128, nur daß die Werte der indizierten Variablen (Arrays) eines Basic-Programms ausgegeben werden. Nach Start des Ladeprogramms mit RUN kann es mit SYS DEC

```
110 REM                                              <02B>
120 REM          DUMPV 128                           <028>
130 REM                                              <029>
140 REM PROGRAMM ZUR AUTOMATISCHEN AUSGABE D
    ER NICHTINDIZIERTEN VARIABLEN                    <02E>
150 REM EINES BASIC 7.0 PROGRAMMES FUER DEN
    COMMODORE 128                                    <02F>
160 REM                                              <02C>
250 SCREENMODE=PEEK(DEC("00D7"))                     <PPQ>
260 IF SCREENMODE=0 THEN SP=0: SX=15: SY=10:
     ELSE SP=5: SX=35: SY=10                         <HF6>
270 SCNCLR SP                                        <K5N>
280 COLOR 0,1: COLOR 6,1: COLOR 4,1: COLOR 5
    ,2                                               <CJU>
290 CHAR 1,SX,SY,"DUMPV 128",1                       <IET>
300 REM DATA-ZEILEN LESEN                            <02E>
310 S=0: FOR I=DEC("0B00") TO DEC("0C1D"): R
    EAD X: S=S+X: POKE I,X: NEXT I                   <UCC>
320 DATA 169,0,141,0,255,165,47,133,252,165,
    48,133,253,197,50,208,12,165,252                 <50M>
330 DATA 197,49,208,6,169,63,141,0,255,96,16
    0,0,32,230,11,16,6,32,78,11,76                   <8GN>
340 DATA 57,11,200,32,230,11,16,6,32,147,11,
    76,57,11,32,115,11,169,13,32,45                  <UPQ>
350 DATA 199,24,165,252,105,7,133,252,165,25
    3,105,0,133,253,76,13,11,32,200                  <7PC>
360 DATA 11,169,37,32,45,199,169,32,32,45,19
    9,32,230,11,133,100,200,32,230                   <860>
370 DATA 11,133,101,162,144,56,32,112,140,32
    ,66,142,32,226,85,96,32,200,11                   <E2E>
380 DATA 169,32,32,45,199,169,32,32,45,199,1
    66,252,164,253,232,208,1,200,232                 <HAM>
390 DATA 208,1,200,138,32,246,11,76,108,11,3
    2,200,11,169,36,32,45,199,169                    <ROI>
400 DATA 32,32,45,199,169,32,32,45,199,32,23
    0,11,133,254,240,27,200,32,230                   <NND>
410 DATA 11,72,200,32,230,11,133,251,104,133
    ,250,160,0,32,238,11,32,45,199                   <1U7>
420 DATA 200,198,254,208,245,96,169,32,32,45
    ,199,160,0,32,230,11,41,127,32                   <E2E>
430 DATA 45,199,200,32,230,11,41,127,208,2,1
    69,32,32,45,199,200,96,165,252                   <TOF>
440 DATA 133,250,165,253,133,251,169,250,162
    ,1,32,116,255,96,133,250,132,251                 <5LV>
450 DATA 160,4,32,238,11,133,103,136,32,238,
    11,133,102,136,32,238,11,133,101                 <4UK>
460 DATA 136,32,238,11,133,104,9,128,133,100
    ,136,32,238,11,133,99,96                         <A4I>
470 SCNCLR SP                                        <N5N>
480 IF S=32209 THEN 510                              <MA5>
490 CHAR 1,SX,SY,"FEHLER IN DATAZEILEN !!!"          <K2M>
500 CHAR 1,SX,SY+2,"DIFFERENZ = "+STR$(32209
    -S)                                              <HUN>
510 END                                              <0NU>
```

Listing 10. »DumpV 128« – Ausgabe nichtindizierter Variablen

("0C1E") aufgerufen werden. Die Ausgabe der Namen und Felder erfolgt analog zur Ausgabe der Variablennamen bei DUMPV 128. Die Werte eines Arrays werden, getrennt durch Kommata, in der Reihenfolge, in der sie im Speicher stehen, ausgegeben.

Da beide Programme verschiedene RAM-Bereiche belegen, können sie gleichzeitig im Speicher stehen und unabhängig voneinander verwendet werden. Die Werte aller Variablen (also einfache und indizierte Variable) können mit SYS DEC ("0DE6") ausgegeben werden, wenn vorher folgendes Programm gestartet wird:

```
10 FOR I=DEC("0DE6") TO DEC("0DEC"): READ X
20 POKE I,X: NEXT I
30 DATA 32,0,11,32,30,12,96
```

(Lothar Glässer/dm)

Aus dem Directory laden

Wird ein Programm mit »DSAVE"NAME[SHIFT+SPACE]:"« gespeichert dann findet man im Directory »"Name":« vor. Es kann nun nach <F3> einfach durch Anfahren mit dem Cursor und <F2> + <RETURN> geladen werden. Mit dem RENAME-Befehl lassen sich natürlich schon vorhandene Programme entsprechend umbenennen.

(Gerhard Seebauer/dm)

```
110 REM                                          <02B>
120 REM          DUMPA 128                       <028>
130 REM                                          <029>
140 REM PROGRAMM ZUR AUTOMATISCHEN AUSGABE D
    ER INDIZIERTEN VARIABLEN                     <02E>
150 REM EINES BASIC 7.0 PROGRAMMES FUER DEN
    COMMODORE 128                                <02F>
160 REM                                          <02C>
250 SCREENMODE=PEEK(DEC("00D7"))                 <PPQ>
260 IF SCREENMODE=0 THEN SP=0: SX=15: SY=10:
     ELSE SP=5: SX=35: SY=10                     <HF6>
270 SCNCLR SP                                    <K5N>
280 COLOR 0,1: COLOR 6,1: COLOR 4,1: COLOR 5
    ,2                                           <CJU>
290 CHAR 1,SX,SY,"DUMPA 128",1                   <J0T>
300 REM DATA-ZEILEN LESEN                        <02E>
310 S=0: FOR I=DEC("0C1E") TO DEC("0DE5"): R
    EAD X: S=S+X: POKE I,X: NEXT I               <8K8>
320 DATA 169,0,141,0,255,165,49,133,252,165,
    50,133,253,197,52,208,12,165,252             <DON>
330 DATA 197,51,208,6,169,63,141,0,255,96,16
    0,0,32,163,13,16,6,32,97,12,76               <22H>
340 DATA 87,12,200,32,163,13,16,6,32,233,12,
    76,87,12,32,171,12,169,13,32,45              <JV0>
350 DATA 199,165,253,76,43,12,32,67,13,169,3
    7,32,45,199,169,32,32,45,199,160             <6NI>
360 DATA 0,32,174,13,133,100,200,32,174,13,1
    33,101,162,144,56,32,112,140,32              <MK0>
370 DATA 66,142,32,226,85,24,165,22,105,2,13
    3,22,165,23,105,0,133,23,197,253             <5LK>
380 DATA 240,4,176,16,144,6,165,22,197,252,1
    76,8,169,44,32,45,199,76,110,12              <N7T>
390 DATA 96,32,67,13,169,32,32,45,199,169,32
    ,32,45,199,165,22,164,23,32,190              <JTJ>
400 DATA 13,32,66,142,32,226,85,24,165,22,10
    5,5,133,22,165,23,105,0,133,23               <DVC>
410 DATA 197,253,240,4,176,16,144,6,165,22,1
    97,252,176,8,169,44,32,45,199                <892>
420 DATA 76,184,12,96,32,67,13,169,36,32,45,
    199,169,32,32,45,199,169,32,32               <3G5>
430 DATA 45,199,160,0,32,174,13,133,254,240,
    27,200,32,174,13,72,200,32,174               <CQK>
440 DATA 13,133,251,104,133,250,160,0,32,182
    ,13,32,45,199,200,198,254,208                <9JH>
450 DATA 245,24,165,22,105,3,133,22,165,23,1
    05,0,133,23,197,253,240,4,176                <EK6>
460 DATA 16,144,6,165,22,197,252,176,8,169,4
    4,32,45,199,76,246,12,96,169,32              <F9F>
470 DATA 32,45,199,160,0,32,163,13,41,127,32
    ,45,199,200,32,163,13,41,127,208             <7OH>
480 DATA 2,169,32,32,45,199,160,4,24,165,252
    ,105,5,133,22,165,253,105,0,133              <76H>
490 DATA 23,32,163,13,24,101,22,133,22,169,0
    ,101,23,133,23,32,163,13,24,101              <LR2>
500 DATA 22,133,22,169,0,101,23,133,23,160,2
    ,32,163,13,24,101,252,8,133,254              <HFV>
510 DATA 200,32,163,13,40,101,253,133,253,16
    5,254,133,252,96,165,252,133,250             <2BL>
520 DATA 165,253,133,251,76,182,13,165,22,13
    3,250,165,23,133,251,169,250,162             <KDR>
530 DATA 1,32,116,255,96,133,250,132,251,160
    ,4,32,182,13,133,103,136,32,182              <Q76>
540 DATA 13,133,102,136,32,182,13,133,101,13
    6,32,182,13,133,104,9,128,133                <EA3>
550 DATA 100,136,32,182,13,133,99,96             <RM1>
560 SCNCLR SP                                    <NTN>
570 IF S=47090 THEN 600                          <ISF>
580 CHAR 1,SX,SY,"FEHLER IN DATAZEILEN !!!"      <K4M>
590 CHAR 1,SX,SY+2,"DIFFERENZ = "+STR$(47090
    -S)                                          <ONR>
600 END                                          <0M6>
```

Listing 11. »DumpA 128« – Ausgabe indizierter Variablen

Den Zehnerblock sinnvoll belegen

Das nächste Programm (Listing 12) demonstriert den Gebrauch der Interruptroutine und der Tastaturbelegung des C128 anhand einer Neubelegung des Zehnerblocks bei gedrückter SHIFT-Taste.

Will man DATA-Werte eingeben, so kommt einem doch der Zehnerblock des C128 sehr gelegen. Um diesen Vorteil auch bei der Assemblerprogrammierung nutzen zu können, fehlt es jedoch an den Buchstaben A bis F. Weiterhin ist zu bemängeln, daß es beim eingebauten Monitor zu Problemen kommt, wenn man einen Teil disassembliert und anschließend verändern will. Stehen hinter dem neu Eingegebenen noch Zeichen, so akzeptiert der Monitor die Eingabe nicht und gibt statt dessen ein mit der Zeit nervendes Fragezeichen aus. Eine Verbesserung der Belegung der Zehnerblock-Tasten, besonders, wenn man mit dem Monitor arbeitet, bewirkt das Programm »10'ER BLOCKER«. Dabei wird lediglich die SHIFT-Version der 14 Zehnerblock-Tasten neu belegt. Die Belegung ohne SHIFT bleibt gleich (Bild 1 zeigt die neue Belegung des Zehnerblocks, gedrückt mit SHIFT).

Untergebracht sind die Buchstaben A bis F, das Komma sowie die folgenden Zeichen: »$«, »%«, »&«, »/« und »*«. Die Null-Taste ergibt, gedrückt mit SHIFT, zwei Nullen (00). Der RETURN-Taste wird in diesem Fall ein Doppelpunkt vorgeschoben. Befindet man sich im Monitor und verändert ein Assemblerlisting, so werden, falls man SHIFT und RETURN gleichzeitig drückt, die Zeichen hinter dem Cursor nicht beachtet.

```
15 REM  * * *  10'ER BLOCKER  * * *              <01B>
20 REM                                           <01A>
40 REM  NEUBELEGUNG DER SHIFT/10'ER              <01C>
50 REM  BLOCK - TASTEN ; DURCH VER-              <01D>
60 REM  AENDERN DER INTERRUPTROUTI-              <01E>
70 REM  NE UND DER TASTATURBELEGUNG              <01F>
80 REM                                           <010>
100 BANK 15: POKE 53280,0: POKE 53281,0          <MCK>
110 PRINT CHR$(147);CHR$(30)                     <DLE>
120 PRINT "AN WELCHE ADRESSE (IN HEX) SOLL D
    IE": PRINT                                   <KJB>
130 PRINT "ROUTINE? PLATZBEDARF:1 BLOCK (256
     BYTES)": PRINT                              <72J>
140 PRINT : INPUT "ADRESSE ($)";A$: FAST         <LEJ>
150 A=DEC(A$): IF A=0 THEN A=5888: REM $1700     <7ER>
155 REM 1.SCHLEIFE FUER DATA-ZAHLEN              <02D>
160 FOR N=A TO A+139: READ Z: POKE N,Z: S=S+
    Z: NEXT                                      <11T>
170 IF S<>17017 THEN PRINT "FEHLER IN DATA'S
    !!!": STOP                                   <B3P>
175 REM 2.SCHLEIFE FUER DATA-ZEICHEN             <02F>
180 FOR N=A+140 TO A+165: READ Z$: Z=VAL(Z$)
    : IF Z=0 THEN Z=ASC(Z$)                      <VFH>
190 POKE N,Z: NEXT                               <51L>
200 PRINT CHR$(147);"INITIALISIEREN MIT:": P
    RINT : PRINT : PRINT                         <7BK>
210 PRINT "BANK15:SYS"A;CHR$(145);CHR$(145);
    CHR$(145)                                    <J92>
215 REM ANPASSEN AN DIE ADRESSE A                <02F>
220 L=(A+166) AND 255: H=INT((A+166)/256)        <912>
230 POKE(A+2) ,(A+13) AND 255                    <UI2>
240 POKE(A+7) ,INT((A+13)/256)                   <QOD>
250 POKE(A+67),L: POKE(A+68),H                   <GMU>
260 POKE(A+80),(A+148) AND 255                   <13T>
270 POKE(A+81),INT((A+148)/256)                  <6U0>
280 POKE(A+83),(A+231) AND 255                   <I5T>
290 POKE(A+84),INT((A+231)/256)                  <QLA>
300 POKE(A+91),L: POKE(A+93),H                   <J46>
310 SLOW : END                                   <L06>
320 DATA 120,169,13,141,20,3,169,23,141,21,3
    ,88,96,216,32,36                             <UOG>
330 DATA 192,144,18,32,248,245,32,208,238,17
    3,13,220,173,4,10,74                         <OT3>
340 DATA 144,3,32,6,64,173,65,3,201,240,144,
    56,174,0,255,169                             <UGV>
350 DATA 15,141,0,255,172,64,3,132,254,172,6
    5,3,132,255,160,0                            <1JH>
360 DATA 177,254,153,166,23,200,192,89,208,2
    46,142,0,255,160,0,185                       <KML>
370 DATA 148,23,153,231,23,200,192,18,208,24
    5,162,166,160,23,142,64                      <QK2>
380 DATA 3,140,65,3,165,211,106,144,22,166,2
    08,240,18,189,73,3                           <72M>
390 DATA 201,48,208,14,236,32,10,176,6,232,1
    57,73,3,134,208,76                           <2E1>
400 DATA 51,255,201,141,208,249,236,32,10,17
    6,244,169,58,157,73,3                        <4SJ>
410 DATA 169,13,208,229,"%","E",24,"B","D","
    $","A",27,"*","/"                            <A9Q>
420 DATA 10,141,"F","&","C",8,48,","             <IR8>
```

Listing 12. Neubelegung der geSHIFTeten Tasten des Zehnerblocks

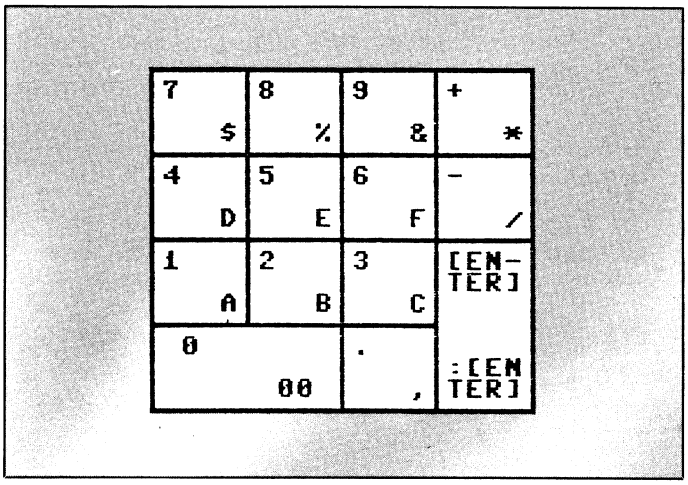


Bild 1. Die Tastenbelegung der geSHIFTeten Tasten des Zehnerblocks

Die Tastaturbelegung

Zunächst muß man wissen, daß der C128 mehrere Tastatur-Decodiertabellen besitzt. Diese Tabellen sind für die verschiedenen Tastenkombinationen mit SHIFT-, CONTROL- und COMMODORE-Tasten vorhanden. Außerdem gibt es weitere Tabellen für die DIN-Belegung, die sich von der ASCII-Version unterscheidet.

Die Tabellen, die uns interessieren, sind die, die in Verbindung mit der SHIFT-Taste gelten. Der Vektor, der auf diese Tabelle zeigt, liegt bei $340/$341. Man transferiert also die Tabelle, auf die dieser Vektor zeigt ($FAD9 bis $FB31), ins RAM, verändert die entsprechenden Werte und verbiegt den Vektor auf die neue Tabelle. Dies funktioniert jedoch nur, bis die ASCII/DIN-Taste gedrückt wird, da dann der Vektor wieder ins ROM auf die DIN-Decodiertabelle ($FD82 bis $FDDA) gestellt wird.

Auch eine Belegung der Null- und RETURN-Taste mit zwei Zeichen gleichzeitig ist so nicht zu realisieren. Die einzige vernünftige Lösung scheint über den Interrupt zu laufen.

Der Interrupt

Beim C128 gibt es, ähnlich wie beim C64, einen Interrupt-Vektor. Dieser Vektor liegt bei $314/$315 und zeigt auf die Routine bei $FA65. Diese Routine geht bis $FA7C und springt anschließend nach $FF33. Will man eigene Routinen in den System-Interrupt einbauen, so kopiert man zunächst diese ROM-Routine ins RAM (etwa nach $1300) und hängt die eigene Routine hinten an. Am Ende der eigenen Routine springt man nach $FF33, um den Interrupt mit dem Wiederherstellen der Register zu beenden. Aktiviert wird diese neue Routine, indem man den Interrupt-Vektor auf ihren Anfang lenkt.

Das Programm

Der Hauptteil dieses Programms läuft im Interrupt ab. Nachdem der Interrupt-Vektor ($314/$315) durch SYS DEC ("1700") auf die Hauptroutine bei $1700 gestellt wurde, werden dort zunächst die gleichen Unterroutinen aufgerufen, wie es die Interrupt-Routine bei $FA65 auch machen würde. Anschließend wird kontrolliert, ob der Zeiger auf die Tastatur-Decodiertabelle ins ROM zeigt. Ist dies der Fall, wird diese Tabelle nach $17A6 kopiert und die Neuerungen im Zehnerblock-Bereich vorgenommen. Nun muß nur noch der Vektor bei $340/$341 auf die neue Tabelle gestellt werden.

Als nächstes prüft das Programm, ob die SHIFT-Taste gedrückt ist. Dazu existiert in der Zeropage ein Flag. Ist die SHIFT-Taste gedrückt, so ist das 0. Bit von $D3 gesetzt. Trifft dies zu, so wird das letzte Zeichen aus dem Tastaturpuffer gelesen (Anzahl der Zeichen im Tastaturpuffer in $D0, Tastaturpuffer ab $349, maximale Länge des Tastaturpuffers bei $A20). Handelt es sich bei dem Zeichen um eine Null, so wird diese nochmals in den Tastaturpuffer geschoben. Ist dies jedoch die RETURN-Taste, so wird zunächst ein Doppelpunkt und anschließend ein RETURN in den Tastaturpuffer geschrieben.

Der Basic-Lader (Listing 12, bitte mit dem neuen Checksummer 128 eingeben) kann das 255 Byte lange Programm an jede beliebige Stelle speichern. Es empfehlen sich folgende Bereiche für das Maschinenprogramm: $0B00 bis $0DFF (Kassetten- und RS232-Ein- und Ausgabepuffer)

```
************** Interruptvektor veraendern
1700 sei          ; Interrupt verbieten
1701 lda #$0d     ; Interruptvektor bei $0314/$0315
1703 sta $0314    ; auf $170d stellen
1706 lda #$17     ; (normal: $fa65)
1708 sta $0315    ;
170b cli          ; Interrupt wieder erlauben
170c rts          ; und zurueck zur Einsprungsadresse

************** Interrupt Einsprung
170d cld          ;
170e jsr $c024    ;
1711 bcc $1725    ;
1713 jsr $f5f8    ; Kopie der ROM-Interruptroutine
1716 jsr $eed0    ; von $fa65 bis $fa7c
1719 lda $dc0d    ;
171c lda $0a04    ;
171f lsr          ;
1720 bcc $1725    ;
1722 jsr $4006    ;

************** Neuer Teil der Interruptroutine
1725 lda $0341    ; Hi-Byte der aktuellen Tast.dec.tabelle
1728 cmp #$f0     ; zeigt in den ROM-Bereich ? ($fa o. $fd)
172a bcc $1764    ; Nein: weiter                (ASCII  DIN)
172c ldx $ff00    ; aktuelles Konfig.-register speichern
172f lda #$0f     ; neue Konfiguration:
1731 sta $ff00    ; Kernal-ROM einschalten
1734 ldy $0340    ; aktuellen Zeiger auf Tastaturdecodier-
1737 sty $fe      ; tabelle 2 in die Zeropage
1739 ldy $0341    ; ($0340/$0341) nach ($fe/$ff)
173c sty $ff      ;
173e ldy #$00     ; Schleife zum transferieren der
1740 lda ($fe),y  ; aktuellen Tastaturdecodiertabelle 2
1742 sta $17a6,y  ; (ASCII oder DIN) an das Ende dieser
1745 iny          ; Routine ($17a6)
1746 cpy #$59     ; laenge der Tabelle
1748 bne $1740    ;
174a stx $ff00    ; alte Speicherkonfig. wieder herstellen
174d ldy #$00     ;
174f lda $1794,y  ; Schleife zum Einbauen der neuen
1752 sta $17e7,y  ; Zeichen in die neue Tastaturdecodier-
1755 iny          ; tabelle 2
1756 cpy #$12     ;
1758 bne $174f    ;
175a ldx #$a6     ; Zeiger ($0340/$0341) auf neue
175c ldy #$17     ; Tastaturdec.tabelle ($17a6)
175e stx $0340    ;
1761 sty $0341    ;
************** Pruefen auf SHIFT/RETURN o. -/"0"
1764 lda $d3      ; Pruefen auf SHIFT-Taste
1766 ror          ; (0.Bit gesetzt=SHIFT gedrueckt)
1767 bcc $177f    ; nicht gedrueckt: Ende
1769 ldx $d0      ; Anzahl Zeichen im Tastaturpuffer
176b beq $177f    ; leer: Ende
176d lda $0349,x  ; Zeichen aus Tastaturpuffer holen
1770 cmp #$30     ; gleich ASCII "0" ?
1772 bne $1782    ; nein: Testen auf RETURN
1774 cpx $0a20    ; Tastaturpuffer voll ?
1777 bcs $177f    ; ja: Ende
1779 inx          ; Anzahl der Zeichen im Puffer um 1 erhoehen
177a sta $0349,x  ; Zeichen in Tastaturpuffer schreiben
177d stx $d0      ; Anzahl der Zeichen im Puffer abspeichern
177f jmp $ff33    ; ****** Ende ****** Ruecksprung zum ROM-IRQ
1782 cmp #$8d     ; Zeichen = SHIFT/RETURN ?
1784 bne $177f    ; Nein: Ende
1786 cpx $0a20    ; Noch Platz im Tastaturpuffer ?
1789 bcs $177f    ; Nein: Ende
178b lda #$3a     ; ASCII ":"
178d sta $0349,x  ; ":" in den Tastaturpuffer
1790 lda #$0d     ; ASCII RETURN
1792 bne $1779    ; Unbedingter Sprung

 1794 .byte 25 45 18 42 44 24 41 1b ; neue Zeichen der Tastatur-
 179c .byte 2a 2f 0a 8d 46 26 43 08 ; decodiertabelle 2
 17a4 .byte 30 2c                   ;

 17a6 - 17ff      ; Platz fuer neue Tastaturdecodiertabelle
```

Listing 13. Der kommentierte Quellcode zum 10'er-Blocker

sowie das freie RAM von $1300 bis $1F77. Gibt man keine Adresse an, so wird es automatisch in den Bereich von $1700 bis $17FF geschrieben.

Ebenfalls ist es möglich, die geSHIFTeten Zehnerblock-Tasten beliebig zu belegen. Dazu müssen lediglich in den DATA-Zeilen 410 und 420 des Listings 12 die entsprechenden ASCII-Zeichen verändert werden.

Beispiel: Will man die SHIFT-Kombination der Taste »9« im Zehnerblock mit dem Zeichen » # « anstelle des »&« belegen, so braucht man nur in der Zeile 420 das »&«-Zeichen durch das » # « zu ersetzen.

Für Maschinensprache-Experten ist in Listing 13 der kommentierte Quell-Code abgedruckt. (Olaf Pfeiffer/dm)

Fehlermeldungen, die nicht im Handbuch stehen

Auch das überraschend dicke Commodore-Handbuch zum C 128 vermag nicht alles zu beantworten. So hat wohl schon so mancher, dem zum Beispiel die Fehlermeldung UNRESOLVED REFERENCE ERROR unterkam, nachgeschlagen, gesucht und doch nicht gefunden...

Insgesamt existieren fünf Fehlermeldungen, die nicht im Handbuch zu finden sind und auch sonst nirgendwo erwähnt werden. Dies sind die Meldungen mit den Nummern 37 bis 41, die Sie auch mit der Funktion ERR$(ER) ansehen können. Hier erfahren Sie nun endlich, was die Meldungen im einzelnen bedeuten:

ER 37: BEND NOT FOUND

Dieser Fehler tritt auf, wenn der Interpreter zu einem BEGIN kein BEND findet. Hat man zum Beispiel eine Befehlssequenz wie

```
IF (Bedingung) THEN BEGIN
```

und die Bedingung ist nicht erfüllt, so sucht der Interpreter nach dem BEND. Ist im nachfolgenden Programm vor dem nächsten BEGIN kein BEND mehr vertreten, so erhalten Sie die oben genannte Fehlermeldung mit der Zeilennummer der letzten Basic-Zeile. Drückt man die HELP-Taste, so wird die ganze Zeile als falsch angegeben. Sehr ärgerlich kann der Fehler sein, wenn in einem Programm mehrere BEGIN und BEND existieren. Dann müssen Sie Ihr Programm von Hand durchsuchen. Als Beispiel sehen Sie Listing 14, das die Wurzel einer Zahl berechnen soll. Die Wurzel kann aber nur von einer positiven Zahl gezogen werden. Also muß bei negativen Zahlen auf die Berechnung verzichtet werden. In diesem Fall soll zum END verzweigt werden. Ihnen ist sicher schon aufgefallen, daß in Zeile 50 das BEND fehlt. Lassen Sie das Programm einmal laufen. Geben Sie eine positive Zahl ein, so arbeitet die Routine richtig. Ist Ihre Zahl negativ, so erhalten Sie ein BEND NOT FOUND ERROR IN 540, also der letzten REM-Zeile. Fügen Sie das BEND ein, so arbeitet das Programm ordnungsgemäß.

Geben Sie nun einmal

```
RENUMBER 10000,10000
```

ein. Damit sind wir schon beim nächsten Fehler.

ER 38: LINE NUMBER TO LARGE

Auf den ersten Blick könnte man vermuten, daß dieser Fehler auftritt, wenn man eine zu große Zeilennummer eingibt. Dies ist aber nicht der Fall. Der Interpreter würde hier einen SYNTAX ERROR ausgeben. Die oben genannte Fehlermeldung tritt beim RENUMBER-Befehl auf, wenn bei der Umnumerierung Zeilennummern auftreten würden, die zu hoch sind. Die höchste erlaubte Zeilennummer ist 63999. Falls Sie bei Listing 14

```
RENUMBER 10000,10000
```

eingeben, so würde in der siebten Programmzeile schon die Zeilennummer 70000 auftreten.

ER 39: UNRESOLVED REFERENCE

Diese Fehlermeldung bezieht sich auch nur auf den RENUMBER-Befehl. Es handelt sich hierbei um das Gegenstück zum UNDEF'D STATEMENT ERROR. Es wird die Nummer der Zeile ausgegeben, in der sich ein Sprungbefehl befindet (GOTO, GOSUB), der sich auf eine Zeile bezieht, die nicht existiert. Fügen Sie beispielsweise

```
55 GOSUB 1000
```

in Listing 14 ein. Wollen Sie das Programm jetzt umnumerieren, erhalten Sie einen UNRESOLVED REFERENCE ERROR IN 55.

```
10 INPUT "{2SPACE}GEBEN SIE EINE ZAHL EIN ";
   A                                          <G4K>
20 IF A>=0 THEN BEGIN                         <JDU>
30 W=SQR(A)                                   <FSL>
40 PRINT "{2SPACE}DIE WURZEL VON";A;"IST";W   <GT6>
60 END                                        <0CM>
70 :                                          <023>
80 :                                          <02T>
500 REM  WENN DIE ZAHL GROESSER ODER          <022>
510 REM  GLEICH 0 IST, SOLL DIE WURZEL        <023>
520 REM  BERECHNET WERDEN. IST DIE ZAHL       <020>
530 REM  KLEINER 0 SO SOLL NICHTS GE-         <021>
540 REM  MACHT WERDEN.                        <026>
```

Listing 14. Ein Versuchs-Programm, um Fehlermeldungen zu erklären (siehe Text)

Allerdings kann man sich diesen Fehler zunutze machen. Haben Sie beispielsweise ein langes Basic-Programm, aus dem Sie jetzt eine Zeile löschen müssen, besteht die Gefahr, daß diese Zeile über GOTO oder GOSUB angesprungen werden könnte. Lassen Sie das Programm laufen, so erhalten Sie einen UNDEF'D STATEMENT ERROR. Angenommen, Listing 14 enthielte ein Unterprogramm ab Zeile 1000, das Sie jetzt gelöscht haben. Sie wissen aber nicht, von wo es angesprungen wurde. Eigentlich hätte man im komfortableren Basic 7.0 einen FIND-Befehl erwarten können. Leider ist das nicht der Fall. Um nun die Zeilen zu finden, die die Zeile 1000 anspringen, fügen Sie an das Listing 14 beispielsweise die Zeile

```
63000 GOTO 63001
```

an, wobei die Zeile 63001 nicht existieren darf. Geben Sie jetzt einfach RENUMBER ein. Daraufhin erhalten Sie einen UNRESOLVED REFERENCE ERROR IN 55. Drücken Sie die HELP-Taste, so haben Sie die erste Stelle gefunden, von der aus das Unterprogramm angesprungen wurde. Ändern Sie diese Zeile und geben Sie wieder RENUMBER ein, bis Sie einen UNRESOLVED REFERENCE ERROR IN 63001 erhalten. Dann sind alle Stellen korrigiert. Vielleicht haben Sie sich schon gefragt, was die Zeile 63000 bewirken soll? Nun, solange das Programm nicht einwandfrei ist, beginnt der Interpreter nicht mit der eigentlichen Umnumerierung.

Die Fehlermeldungen 40 und 41 (UNIMPLEMENTED COMMAND ERROR und FILE READ ERROR) konnten noch nicht genau entschlüsselt werden, da die nähere Funktion vom Namen her vielleicht erkennbar ist, aber noch keine stichhaltigen Beweise für Vermutungen vorliegen. Wenn Sie diese Fehlermeldungen schon entlarvt haben, sind wir jederzeit für Tips dankbar. (Thomas Hansch/dm)

Lissajous-Figuren

Ein kleines Programm, das das Grundgerüst zum Berechnen und Zeichnen von Lissajous-Figuren ermöglicht (Listing 15, bitte mit dem neuen Checksummer 128 eingeben). Durch Verändern verschiedener Werte (siehe REM-Zeilen) können Sie alle möglichen Figuren erzeugen.

(Matthias Jäger/dm)

```
10 GRAPHIC 1,1                                  <LUI>
20 WIDTH 2                                      <32H>
30 X0=160: Y0=100: A=150: B=95                  <QOU>
40 D=5: N=3: M=4: T0=π/180                      <CQ3>
50 COLOR 0,1: COLOR 4,2: COLOR 1,2              <LSI>
60 FOR W=0 TO 180*D STEP D                      <NFK>
70 T=W*T0                                       <3HC>
80 X=X0+A*COS(N*T)                              <JJ6>
90 Y=Y0+B*SIN(M*T)                              <D3Q>
100 IF T<>0 THEN DRAW 1,X1,Y1 TO X,Y            <PU2>
110 X1=X: Y1=Y                                  <D57>
120 NEXT W                                      <2LB>
130 COLOR 1,3                                   <LS7>
140 PAINT 1,0,0                                 <LPD>
150 REM FIGUR 1 - D=5  , N=3 , M=4              <02F>
160 REM FIGUR 2 - D=77 , N=1 , M=2              <02C>
170 REM FIGUR 3 - D=77 , N=2 , M=1              <02D>
180 REM FIGUR 4 - D=77 , N=2 , M=3              <022>
190 REM FIGUR 5 - D=77 , N=3 , M=2              <023>
200 REM FIGUR 6 - D=94 , N=1 , M=1              <02C>
210 REM FIGUR 7 - D=94 , N=2 , M=3              <02D>
220 REM FIGUR 8 - D=94 , N=1 , M=2              <02E>
230 REM FIGUR 9 - D=94 , N=1 , M=3              <02F>
```

Listing 15. Mit diesem kleinen Programm können Sie bezaubernde Lissajous-Figuren erzeugen. Bitte benutzen Sie zur Eingabe den Checksummer 128 auf Seite 122.

Noch ein paar PEEKs und POKEs

POKE 160,255 - Zurückstellen der internen Uhr (TI$ = 000000)

PEEK (199) - zeigt an, in welcher BANK ein File abgelegt ist

POKE 216,0 - ersetzt GRAPHIC 0,1

POKE 216,1 - ersetzt GRAPHIC 1,1

POKE 216,64 - ersetzt GRAPHIC 2,1

POKE 216,128 - ersetzt GRAPHIC 3,1

POKE 216,254 - ersetzt GRAPHIC 4,1

POKE 241,X - Ändern der Schriftfarbe (X = 0 bis 15)

POKE 243,1 - Einschalten des Revers-Modus

POKE 243,0 - Ausschalten des Revers-Modus

POKE 246,255 - Aktivieren des automatischen Einfügemodus (ESC-A)

POKE 246,1 - Desaktivieren des automatischen Einfügemodus (ESC-C)

POKE 247,64 - Schaltet NO-SCROLL-Taste ab

POKE 247,0 - NO-SCROLL-Taste wieder normal

POKE 247,255 - Unterbinden der Groß-/Kleinschrift-Umschaltung mit COMMODORE/SHIFT

POKE 248,254 - Der Bildschirm rollt sich beim LISTen auf (ESC-M)

POKE 249,255 - Schaltet Klingelzeichen (CTRL-G) ab

POKE 249,0 - Schaltet das Klingelzeichen wieder ein

POKE 774,102:POKE 775,224 - Listschutz

POKE 774,81 :POKE 775,81 - Hebt den Listschutz wieder auf

POKE 820,189 - ESC wird im Quote-Modus sichtbar

POKE 820,185 - ESC reagiert wieder normal

POKE 2594,0 - Keine Tasten-Wiederholung bis auf DEL, INST und CRSR

POKE 2594,64 - Alle Tasten haben Auto-Repeat

POKE 2594,128 - Normalzustand Tasten-Repeat

POKE 2598,64 - bringt den Cursor zum Stillstand

POKE 2598,0 - Cursor blinkt wieder

POKE 2603,128 - Fester Cursor im 80-Zeichen-Modus

POKE 2603,96 - Blinkender Cursor im 80-Zeichen-Modus

POKE 2603,32 - Abgeschalteter Cursor im 80-Zeichen-Modus

POKE 2603,64 - Schnell blinkender Cursor im 80-Zeichen-Modus

SYS 45096 - Monitor wird aktiviert

(Michael Bartels/Helmut Schottmüller/dm)

Laufschrift mit Sprites

Diese auf Spritebasis arbeitende Laufschrift, die jeder Anwender in seine Programme einbauen kann, läßt sich ohne viel Aufwand realisieren. Durch die Interruptsteuerung der Sprites wird der Programmablauf nicht beeinflußt. Dazu ist nur folgendes kleine Programm nötig:

```
10 INPUT"FARBNUMMER (1-16)";F
20 INPUT"GESCHWINDIGKEIT (1-6)";G
30 INPUT"TEXT (BIS 24 ZEICHEN)";I$:SCNCLR
40 GRAPHIC 1,1:CHAR 1,0,0,I$
50 SSHAPE A$,0,0,16,20:SPRSAV A$,1
60 SSHAPE B$,23,0,39,20:SPRSAV B$,2
70 SSHAPE C$,47,0,63,20:SPRSAV C$,3
80 SSHAPE D$,71,0,87,20:SPRSAV D$,4
90 SSHAPE E$,95,0,111,20:SPRSAV E$,5
100 SSHAPE F$,119,0,135,20:SPRSAV F$,6
110 SSHAPE G$,143,0,159,20:SPRSAV G$,7
120 SSHAPE H$,167,0,183,20:SPRSAV H$,8
130 SCNCLR:B=0:FOR A=1TO8:SPRITE A,1,F,1,1,0:B=B+40
140 MOVSPR A,B,51:MOVSPR A,270#G:NEXT
```

(Karl-Heinz Montag/dm)

Bildschirmhardcopy 80-Zeichen-Schirm

Der Video-Controller für den 80-Zeichen-Schirm verwaltet einen eigenen 16 KByte großen Speicher, der außerhalb des normalen Adreßbereiches des Prozessors liegt. Das bedeutet, daß der Video-Speicher nur indirekt angesprochen werden kann, hat aber den Vorteil, daß er keinen Arbeitsspeicher belegt (normalerweise liegt der Bildschirmspeicher innerhalb des VDC-RAMs von $0000 bis $07CF).

Diese kleine Routine erlaubt eine Hardcopy des 80-Zeichen-Bildschirms.

Arnulf Bechmann/dm)

```
10 CLOSE 1:OPEN 4,4,1
20 A=DEC("D0600"):D=A+1
30 FOR J=0 TO 1999 STEP 80
40 FOR Y=0 TO 79
50 BY=J+Y:HB=INT(BY/256):LB=BY*256
60 POKE A,18:POKE D,HB
70 POKE A,19:POKE D,LB
80 POKE A,31:B=PEEK(D)
90 IF B=>1 AND B<=26 THEN B=B+96
100 PRINT#,CHR$(B)
110 NEXT Y,J
120 PRINT#4:CLOSE4:END
```