DATA-Erzeuger

Dieses Programm dient, wie der Name schon sagt, zum automatischen Erzeugen von DATA-Zeilen. Da das Programm in Maschinensprache geschrieben ist, ist die Ausführungszeit entsprechend kurz. Der Inhalt von 10000 Speicherzellen wird in weniger als 4 Sekunden in DATA-Zeilen umgewandelt!

Es kommt immer wieder vor, daß der Inhalt von Speicherzellen in DATA-Zeilen umgewandelt werden muß. Verschiedene Lösungen habe ich aus irgendwelchen Büchern oder Zeitschriften abgetippt. Da ich keine dieser Programme als befriedigend empfand, begann ich, einen DATA-Erzeuger in Basic zu programmieren. Auch dieses Programm war zu unflexibel und vor allem zu langsam. Da ich zwischenzeitlich verstärkt in Maschinensprache programmiere, bestand ein noch größerer Bedarf an einem leistungsfähigen DATA-Erzeuger. Also beschloß ich, einen solchen in Maschinensprache zu erstellen.

Leicht zu bedienen

Durch eine umfangreiche Fehlerprüfung ist das Programm sehr anwendungssicher. Die DATA-Zeilen können, falls gewünscht, an ein Basic-Programm angehängt werden, und das beliebig oft.

Eingabe-Format:

SYS 49152,ANFADR,ENDADR,ANFZNR,SCHRITTWEITE

ANFADR: Adresse der ersten Speicherzelle, aus deren Inhalt DATA-Zeilen erzeugt werden sollen.

ENDADR: Adresse der letzten Speicherzelle für die DATA-Zeilen-Erzeugung.

ANFZNR: Zeilennummer der ersten DATA-Zeile (muß größer sein als die letzte Zeilennummer im Basic-Programm, falls eines im Speicher ist).

SCHRITTWEITE: Um diese Zahl wird die Zeilennummer bei der DATA-Erzeugung jeweils erhöht.

Fehlermeldungen

Für ANFADR, ENDADR, ANFZNR dürfen nur ganze, positive Zahlen von 0-65536 eingegeben werden. Die SCHRITTWEITE darf zwischen 1 und 255 liegen. Werden Zahlen eingegeben, die außerhalb des oben genannten Bereichs liegen, so kommt es zur Fehlermeldung »? ILLEGAL QUANTITY ERROR«. Weitere Fehlermeldungen sind:

— ANFANGSADRESSE GROESSER ALS ENDADRESSE.
FEHLER! Logischerweise muß die Anfangsadresse kleiner sein als die Endadresse.

— 0 ALS SCHRITTWEITE NICHT MOEGLICH!
Würde 0 als Schrittweite akzeptiert werden, so hätten alle erzeugten DATA-Zeilen die gleiche Zeilennummer.

— ZEILENNUMMER GROESSER ALS 63999!
Bei der Erzeugung der DATA-Zeilen wurde eine Zeilennummer erzeugt, die größer als 63999 war. Da der C 64 nur Zeilennummern bis 63999 zuläßt, würden größere Zeilennummern Probleme der Art aufwerfen, daß sie sich nicht mehr löschen oder ändern ließen. Man sollte eine kleinere Anfangs-Zeilennummer oder Schrittweite wählen, um den Fehler zu beseitigen.

— FREIER SPEICHERPLATZ REICHT NICHT!
Bei der DATA-Erzeugung wurde festgestellt, daß der freie RAM-Bereich nicht ausreicht.

Beim Auftreten eines Fehlers wird die DATA-Erzeugung, falls schon begonnen, sofort abgebrochen. Die bis dahin erzeugten DATA-Zeilen werden wieder gelöscht. Die Programmzeiger werden wieder so eingestellt, wie es vor dem Aufruf des DATA-Erzeugers der Fall war.

War die DATA-Zeilen-Erzeugung erfolgreich, so erscheint die Meldung »DATA ERZEUGER FERTIG! (C) J. MATERNA«. Jetzt befinden sich die DATA-Zeilen und die Schleife zum Lesen und ZurückPOKE der DATAs im Speicher. Das Basic-Programm kann wie gewohnt gelistet, gestartet oder gespeichert werden.

Wenn man möchte, kann man noch beliebige weitere DATA-Zeilen erzeugen. Sie werden automatisch an die bereits vorhandenen angehängt. Wichtig: Die neue Anfangszeilennummer muß größer sein als die letzte im Programm verwendete. Am besten LISTet man das Programm erst auf, um die höchste Zeilennummer festzustellen. Möchte man die DATA-Blöcke als Unterprogramme aufrufen, muß natürlich noch am Ende des DATA-Blocks eine Zeilennummer mit »RETURN« eingegeben werden.

Tips und Beispiele

Bei allen Bereichen wird vorausgesetzt, daß sich der DATA-Erzeuger lauffähig im Speicher befindet, ebenso die Daten, aus denen DATA-Zeilen erzeugt werden sollen.

Sprite-Blöcke übernehmen

— Das Programm, das die Sprites erzeugt, laden und starten. Wenn die Sprites erzeugt sind, das Programm abbrechen und die Adressen der Sprites aufschreiben.

— Das Programm mit NEW löschen.

— Das eigene Programm, das die Sprites verwenden soll, laden und die höchste Zeilennummer feststellen.

— Beispielsweise SYS 49152, Anfangsadresse der Sprites, Endadresse der Sprites, höchste Zeilennummer des eigenen Programms plus 10, sowie beliebige Schrittweite eingeben.

— Den letzten Punkt beliebig oft wiederholen, bis alle Sprites in DATA-Zeilen umgewandelt sind. Dabei darf nicht vergessen werden, stets erneut die höchste Zeilennummer festzustellen.

— Es ist sinnvoll, am Ende der DATA-Blöcke jeweils eine Zeile mit »RETURN« einzugeben. Danach können die Sprites irgendwo vom Hauptprogramm aus mit einem GOSUB-Befehl aufgerufen werden.

— Programm testen und speichern.

Nach dem oben beschriebenen Verfahren kann man natürlich nicht nur Sprites übernehmen, sondern auch Maschinenprogramme und neue Zeichensätze. Insbesondere kann man auch sehr gut Maschinenprogramme, die von einem Assembler erzeugt wurden, an ein Basic-Programm anhängen.

Maschinenprogramm für Datenfernübertragung vorbereiten

Da die Daten im ASCII-Format übertragen werden, müssen sie vorher umgewandelt werden.

Beispiel: Das zu übertragende Programm ist 8000 Bytes lang und wird normal ab Basic-Anfang geladen.

— Das zu übertragende Programm laden.
— POKE 44,40 : POKE 40 * 256,0 : NEW
Damit ist das Programm vor dem Überschreiben geschützt. Wenn das Maschinenprogramm am Speicherende liegt, dann muß natürlich der Zeiger für das Basic-Ende herabgesetzt werden.
— SYS 49152,2049,10049,100,2

Nun muß das Programm nur noch als Datei gespeichert werden und kann dann von einem Terminal-Programm gesendet werden.

Ein Monitor-Programm ($9000 — $9FFF) in DATA-Zeilen verwandeln.

— Monitor laden.
— SYS 49152,36864,40959,100,2.
— Abspeichern
Das Programm kann nun normal geladen und gestartet werden. Danach kann der Monitor mit der entsprechenden SYS-Adresse aufgerufen werden.

Bildschirminhalt in DATA-Zeilen verwandeln

SYS 49152,1024,2023,100,2
Das Programm kann sofort gestartet werden.

Programmierte DATA-Erzeugung

Der DATA-Erzeuger läßt sich auch von Basic-Programmen aus aufrufen. Dabei dürfen die Parameter natürlich in Variablen übergeben werden. Doch Achtung: Wenn der DATA-Erzeuger fertig ist, wird ein CLR ausgeführt und zum Basic-Warmstart gesprungen. Dabei wird das Basic-Programm verlassen.

Um wieder ins Basic-Programm zurückzukommen, empfehle ich folgenden Trick, den ich am besten durch ein Beispiel demonstriere:

```
10 A=900:Z=100
20 A=A+100:Z=Z+20:IF Z=200 THEN END
30 PRINT CHR$(147);:PRINT CHR$(17):PRINT CHR$(17)
40 PRINT"A=";A;":Z=";Z;":GOTO 20";CHR$(19)
50 POKE 631,13:POKE 198,1
60 SYS 49152,A,A+100,Z,2:END
```

Dieses Programm hält in Zeile 60 an und startet sich so lange wieder automatisch, bis in Zeile 20 Z=200 ist. Dabei werden die Variablen, die in Zeile 40 auf dem Bildschirm ausgegeben werden, mit übernommen. Für den Auto-Start des Programms ist die Zeile 50 verantwortlich. Dort wird ein RETURN (POKE 631,13) in den Tastaturpuffer gePOKEt und die Länge des Tastaturpuffers (POKE 198,1) auf 1 gesetzt.

Anhand dieses Beispiels sollte jeder in der Lage sein, dieses Problem zu lösen.

Ich hoffe, daß ich durch meine Beispiele genug Anregungen für den Einsatz des DATA-Erzeugers gegeben habe.

Die Programmeingabe

Nachdem das Programm in den C64 eingegeben ist, kann es bedenkenlos gestartet werden.

Falls dann die Meldung «DATA-FEHLER« erscheint, müssen die DATA-Zeilen noch einmal überprüft werden. Wenn die Meldung «PROGRAMM OK« erscheint, speichert man das Programm am besten ab, bevor man den DATA-Erzeuger aufruft. Vor dem Programmstart mit »SYS« löscht man am besten den Speicher mit »NEW« oder lädt das Programm, für das DATAs erzeugt werden sollen.

Möchte man den DATA-Erzeuger als Maschinenprogramm direkt speichern, so sind folgende Befehle einzugeben:

```
POKE 43,0:POKE 44,192:POKE 45,66:POKE 46,195:SAVE
"MP-DATA-ERZEUGER",8,1
```

Für die Datasette muß statt der 8 natürlich eine 1 eingegeben werden.

Das so gespeicherte Programm lädt man mit »LOAD "MP-DATA-ERZEUGER",8,1«.

Nachdem man NEW eingegeben hat, ist das Programm sofort einsatzbereit.

(Jörg Materna / ev)

```
10 REM ********************************    <217>
15 REM *                              *    <242>
20 REM *          DATA-MAKER          *    <174>
30 REM *                              *    <001>
40 REM *        JOERG MATERNA         *    <138>
50 REM *        BECKUMER STR.91       *    <244>
60 REM *        4730 AHLEN            *    <085>
65 REM *                              *    <036>
70 REM ********************************    <021>
75 REM                                      <218>
80 PRINT CHR$(147)                          <157>
100 FOR A=49152 TO 49986:READ X:POKE A,X:SU=SU+X
   :NEXT                                    <062>
105 IF SU<>102090 THEN PRINT "DATA-FEHLER"  <173>
106 PRINT "PROGRAMM[SPACE]OK"               <070>
110 DATA 169,0,141,65,195,32,233,193,132,251,
   133,252,32,233,193,140,54,195,141,55     <212>
120 DATA 195,32,233,193,140,59,195,141,60,195,
   32,253,174,32,158,183,142,63,195,138     <042>
130 DATA 208,3,76,58,194,165,252,205,55,195,144,
   12,208,7,165,251,205,54,195,144          <156>
140 DATA 3,76,68,194,56,165,45,233,2,141,61,195,
   133,253,165,46,233,0,141,62,195          <157>
150 DATA 133,254,160,2,173,59,195,145,253,200,
   173,60,195,145,253,200,169,129,145       <214>
160 DATA 253,200,169,65,145,253,200,169,178,145,
   253,200,152,72,165,251,166,252,32        <014>
170 DATA 243,193,141,64,195,104,168,32,7,194,
   169,164,145,253,200,152,72,173,54,195    <089>
180 DATA 174,55,195,32,243,193,141,64,195,104,
   168,32,7,194,169,58,145,253,200,169      <059>
190 DATA 135,145,253,200,169,88,145,253,200,169,
   58,145,253,200,169,151,145,253,200       <094>
200 DATA 169,65,145,253,200,169,44,145,253,200,
   169,88,145,253,200,169,58,145,253        <025>
210 DATA 200,169,130,145,253,76,52,193,160,2,
   173,59,195,145,253,200,173,60,195,145    <117>
220 DATA 253,200,169,131,145,253,200,152,72,160,
   0,177,251,32,160,193,104,168,173         <005>
230 DATA 56,195,201,48,240,3,145,253,200,173,57,
   195,201,48,208,10,173,56,195,201         <029>
240 DATA 48,240,6,173,57,195,145,253,200,173,58,
   195,145,253,165,251,205,54,195,208       <163>
250 DATA 13,165,252,205,55,195,208,6,238,65,195,
   76,52,193,230,251,208,2,230,252          <009>
260 DATA 192,74,176,8,200,169,44,145,253,76,223,
   192,200,169,0,145,253,200,140,64         <062>
270 DATA 195,160,0,24,165,253,109,64,195,145,
   253,141,64,195,165,254,105,0,200,145     <119>
280 DATA 253,133,254,197,56,208,3,76,45,194,173,
   64,195,133,253,173,65,195,208,28         <119>
290 DATA 24,173,59,195,109,63,195,141,59,195,
   144,13,238,60,195,173,60,195,201,250     <166>
300 DATA 208,3,76,32,194,76,205,192,24,165,253,
   105,2,133,45,165,254,133,46,144,2        <100>
310 DATA 230,46,160,0,152,145,253,200,145,253,
   32,78,194,32,96,166,76,134,227,162       <114>
320 DATA 0,142,56,195,142,57,195,142,58,195,201,
   100,144,8,233,100,238,56,195,76          <080>
330 DATA 171,193,201,10,144,8,233,10,238,57,195,
   76,183,193,201,0,240,8,238,58,195        <186>
```

```
340 DATA 233,1,76,195,193,24,169,48,109,56,195,
   141,56,195,169,48,109,57,195,141          <139>
350 DATA 57,195,169,48,109,58,195,141,58,195,96,
   32,253,174,32,138,173,32,247,183          <201>
360 DATA 96,133,99,134,98,162,144,56,32,73,188,
   32,223,189,32,135,180,32,166,182          <144>
370 DATA 96,162,0,165,34,141,20,194,165,35,141,
   21,194,189,0,1,145,253,200,232,206        <202>
380 DATA 64,195,208,244,96,32,86,194,169,120,
   160,194,32,30,171,76,154,193,32,86        <161>
390 DATA 194,169,155,160,194,32,30,171,76,154,
   193,169,192,160,194,32,30,171,76,154      <058>
400 DATA 193,169,229,160,194,32,30,171,76,154,
   193,169,11,160,195,32,30,171,96,173       <015>
410 DATA 61,195,133,253,173,62,195,133,254,160,
   0,152,145,253,200,145,253,24,173          <155>
420 DATA 61,195,105,2,133,45,173,62,195,105,0,
   133,46,96,18,90,69,73,76,69,78,78         <207>
430 DATA 85,77,77,69,82,32,71,82,79,69,83,83,69,
   82,32,65,76,83,32,54,51,57,57,57          <045>
440 DATA 33,146,0,18,70,82,69,73,69,82,32,83,80,
   69,73,67,72,69,82,80,76,65,84,90          <034>
450 DATA 32,82,69,73,67,72,84,32,78,73,67,72,84,
   33,146,0,18,48,32,65,76,83,32,83          <034>
460 DATA 67,72,82,73,84,84,87,69,73,84,69,32,78,
   73,67,72,84,32,77,79,69,71,76,73          <090>
470 DATA 67,72,33,146,0,18,65,78,70,65,78,71,83,
   65,68,82,69,83,83,69,32,62,32,69          <068>
480 DATA 78,68,65,68,82,69,83,83,69,46,70,69,72,
   76,69,82,33,146,0,18,42,32,68,65          <089>
490 DATA 84,65,32,69,82,90,69,85,71,69,82,32,70,
   69,82,84,73,71,33,32,40,67,41,32          <070>
500 DATA 74,46,77,65,84,69,82,78,65,46,32,42,
   146,0,66,195,49,52,50,244,1,1,8,10        <018>
510 DATA 48,0,0                              <166>
```

Single-Step für VC 20

»Single-Step« soll bei Maschinenprogrammen zum Aufdecken von Fehlern beitragen. Anfängern erlaubt es ein schnelleres Einfühlen in die Wirkungsweise der Maschinenbefehle.

Das Programm ist für den Commodore VC 20 mit mindestens 8 KByte Speichererweiterung geschrieben (ein entsprechendes Programm für den C64 finden Sie im 64'er Stammagazin). Es ist ein DATA-Listing einer Maschinenspracheroutine. Über Zeile 9 des Listings läßt sich die gewünschte Startadresse festlegen; die Basic-Schleife ab Zeile 300 korrigiert absolute Sprungadressen. Hierzu wurden bei der Programmerstellung die absoluten Zieladressen so eingestzt, als stünde das Programm ab Adresse 0 im Speicher. Der Basic-Teil sorgt dafür, daß die Startadresse (ss) als Offset berücksichtigt wird.

Nach einmaligem Starten läßt sich die Maschinenspracheroutine mit SYS ss starten, der Basic-Teil kann demnach gelöscht werden, sollte aber sicherheitshalber vorher abgespeichert werden.

Für die Bedienung des Programmes stehen die Funktionstasten F1, F3 und F5 zur Verfügung. Ein Beispiel soll den Ablauf verdeutlichen (Tabelle):

Zunächst startet man »Single-Step« mit SYS ss, es erscheint dann die Überschrift mit Bezeichnung der einzelnen Bildschirmspalten und der ersten, übersetzten Zeile. Nun kann man über F3 in den Editier-Modus gelangen, es lassen sich sämtliche dargestellte Register und Flags ändern. Dazu fährt man mit dem Cursor einfach an die entsprechende Stelle und tippt die gewünschten Werte ein. Im Editier-Modus wird nun der PC (Programm-Counter) so geändert, daß er auf die Startadresse der eigenen Routine zeigt (hier im Beispiel ist dies die Adresse $ 4000). Durch Betätigen der Taste F1 wird der hier abgelegte Befehl ausgeführt. Der PC zeigt dann auf die Startadresse des nächsten Befehles, PSW (Programm Status Word, auch Flag-Register genannt),A,X und Y sind entsprechend verändert. (Im Beispiel wurde der Akku mit den Daten #00 geladen). Jede weitere Betätigung der Funktionstaste F1 bringt einen weiteren Befehl zur Ausführung.

Jetzt noch eine Anmerkung zu Manipulationen im PSW (Flag-Register): Durch Ändern einzelner Flags lassen sich zum Beispiel Schleifen vorzeitig beenden oder, wie im Beispiel in Zeile 16 geschehen, auch verlängern; das Zero-Bit wurde zurückgesetzt, was die CPU durch einen weiteren relativen Sprung auf Adresse $ 4000 quittieren mußte.

Weiterhin lassen sich unter Benutzung des PSW, genauer des I-Flags, Breakpoints im Programm setzen. Es ist nicht immer nötig das Programm schrittweise von Anfang bis Ende zu untersuchen. In solchen Fällen setzt man zu Beginn des inter-

**** Beispiel ****

```
$4000        lda #0      a900     ;sum gleich null
$4002        tax         aa       ;index gleich null
$4003        sed         f8       ;dezimale addition
$4004  sum   clc         18       ;übertrag nicht berücksichtigen
$4005        adc $2,x    7502     ;sum = sum+daten
$4007        inx         e8
$4008        cpx $0      e400     ;alle elemente ?
$400a        bne sum     d0f8     ;nein, dann weiter
$400c        sta $1      8501     ;lege Ergebnis ab
$400e        rts         60

$0000        hex         02       ;anzahl elemente
$0001                    00       ;platz für summe reservieren
$0002                    793327   ;zu add. elemente in packed bcd
```

Ablauf mit Single-Step:

```
00 :** single  —  step **  :
01 :p c  nv-bdizc a  x  y  :
02 :                       :
03 :e14e-00-00011-00-00-2c: ;ändern in →
04 :4000-           - ff - ff -  : ;next step führt zu →
05 :4002-00-00011-00- ff -2c:
06 :4003-00-00011-00-00-2c: ;next step = set decimal flag →
07 :4004-00-01011-00-00-2c: ;clear carry →
08 :4005-00-01010-00-00-2c:
09 :4007-00-01000-79-00-2c: ;inkrementiere x-reg. →
10 :4008-00-01000-79-01-2c: ;vergleiche, setze n, z, c →
11 :400a-10-01000-79-01-2c: ;nicht null also sprung →
12 :4004-10-01000-79-01-2c:
13 :4005-10-01000-79-01-2c: ;addiere, setze n,v,z,c →
14 :4007-11-01001-12-01-2c: ;inkrementiere x-reg. →
15 :4008-01-01001-12-02-2c: ;vergleiche, setze n,z,c →
16 :400a-01-01011-12-02-2c: ;null also kein sprung →
17 :400c-01-01011-12-02-2c: ;ergebnis ablegen ...
```

mit Änderung:

```
16  :400a-01-01011-12-02-2c: ;ändern →
16' :400a-          0        : ;erzwingt sprung →
17' :4004-01-01001-12-02-2c:
18' :4005-01-01000-12-02-2c: ;addiere →
19' :4007-00-01000-39-02-2c: ;inkrementiere x-reg. →
20' :4008-00-01000-39-03-2c:
```

Beispielgrogramm und Ablauf mit »Single Step«