Zeichen-Editor

Bei vielen Anwendungen ist es sinnvoll mit zwei verschiedenen Zeichensätzen zu arbeiten. Dieses Programm ermöglicht Ihnen, einen eigenen Zeichensatz zu erstellen, ohne den Original-Zeichensatz zu zerstören.

Programmiert man ein Videospiel, ein Textverarbeitungsprogramm oder will man einfach nur die üblichen Bildschirmzeichen etwas interessanter gestalten, bleibt einem nichts anderes übrig, als den normalen Zeichensatz aus dem ROM herauszuholen, ins RAM zu kopieren und dann in diesem kopierten Zeichensatz »herumzuPOKEn«.

Diese Tätigkeit ist aber — ähnlich wie bei der Konstruktion von Sprites — immer wieder eine mühsame Rechnerei. Deshalb habe ich mit einen komfortablen Zeichen-Editor zusammengestellt, der jede Rechnerei abnimmt. Mit seiner Hilfe ist das Definieren eigener Grafik-Zeichen ein Kinderspiel.

Nach dem Eintippen (und Abspeichern!) läßt man das Programm mit RUN starten. Nach einer kurzen Wartezeit, während der Großbuchstaben-Zeichensatz aus dem ROM ins RAM kopiert wird (und zwar in die Speicherzellen 51200 bis 53248, der Bildschirm beginnt dann bei Adresse 50176), erscheint das Menü mit einem Zeichenfeld links unten. (Wen es interessiert: das Maschinenprogramm, das den Zeichensatz ins RAM kopiert, beginnt bei der Adresse 828, also dem Anfang des Kassettenpuffers).

In diesem Zeichenfeld kann man nun mit den Cursortasten herumwandern, Sternchen (*) malen und gegebenenfalls mit der Space-Taste wieder löschen. Ein Sternchen im Zeichenfeld bedeutet einfach, daß hier ein Bit gesetzt wird, das dann später zur Berechnung des Zeichens dient. Jedes Graphik-Zeichen besteht ja aus 8 Bytes — und genau diese 8 Bytes stellt das Zeichenfeld symbolisch dar.

Taste 1: Berechnung des selbsterstellten Zeichens

Sobald man sein Zeichen gemalt hat, muß es berechnet werden. Hierfür ist im Menü der Programmpunkt 1 vorgesehen. Drückt man diese Taste, so wird gefragt: »Welche Taste?« Man drückt nun die Taste, der man das soeben erstellte Zeichen zuordnen möchte, und genau das Zeichen, das zu der gedrückten Taste gehört, wird nun durch das im Zeichenfeld definierte Zeichen ersetzt.

Ein Beispiel: Füllt man das Zeichenfeld ganz mit Sternchen aus, läßt das Feld berechnen und drückt dann die Taste »A«, so wird überall auf dem Bildschirm dort, wo eben noch ein »A« gestanden hat, ein reverses Quadrat erscheinen. Nette Spielchen kann man zum Beispiel mit der Space-Taste machen: dann wird nämlich überall da, wo ein Space auf dem Bildschirm ist (und das sind ja üblicherweise eine ganze Menge), das eben definierte Zeichen gedruckt. Auf diese Weise kann man den Bildschirm etwas interessanter gestalten.

Taste 2:

Sobald das neue Zeichen berechnet und ausgedruckt ist, kann man wieder beliebig im Zeichenfeld herumhantieren. Will man jedoch ein ganz anderes Zeichen konstruieren, so drückt man einfach Taste 2, und flugs ist das Zeichenfeld wieder »sauber« — das gerade definierte Zeichen wird dabei natürlich nicht gelöscht. Zusätzlich zum reinen Konstruieren von Grafik-Zeichen gibt es jodoch auch noch andere Programmfunktionen:

Taste 3: ROM-Zeichen auslesen

Mit Hilfe dieses Programmpunktes kann man sich ein beliebiges Zeichen aus dem ROM-Zeichensatz herholen. Hierbei wird das gewünschte Zeichen in das Zeichenfeld hineingePOKEt und die Daten dieses Zeichens rechts daneben ausgegeben. Selbst wenn Sie also den Buchstaben »A« als ein reverses Quadrat definiert haben sollten, so erscheint — sofern Sie Taste 3 und danach »A« drücken — auf dem Zeichenfeld das gute, alte »A« wieder (denn im ROM-Zeichensatz bleibt natürlich alles beim alten).

Taste 4: Eigene Zeichen auslesen

Natürlich können Sie nicht nur Zeichen aus dem ROM holen, sondern auch aus dem kopierten Zeichensatz, in dem Sie bisher munter herummanipuliert haben. Sie haben zum Beispiel aus dem »O« ein Smiley-Gesicht gemacht und wollen er gerne in vergrößerte Form wiedersehen, um etwa Korrekturen oder ähnliches vorzunehmen — bitte sehr! Drücken Sie die Taste 4, dann ein »O« und Ihr eigenes Zeichen steht im Zeichenfeld — samt den dazugehörigen Daten rechts nebenan.

Taste 8: Restore

Nun kann es vorkommen, daß Sie genug haben von Ihren eigenen Zeichen. Eigentlich wollten Sie jetzt ganz gern wieder die alten, »normalen« Zeichen anstelle der vielen Smiley-Gesichter und reversen Quadrate sehen.

Wenn Sie Taste 8 drücken, wird einfach der ROM-Zeichensatz wieder ins RAM kopiert; und da dieser Teil in Maschinensprache geschrieben ist, geht das ziemlich schnell vonstatten.

Will man übrigens nur ein einziges Zeichen wieder in den Ursprungszustand versetzen, so geht man am besten so vor: Man holt sich das (alte) Zeichen aus dem ROM (mit Taste 3) und läßt es mit Taste 1 wieder in den neuen Zeichensatz hineinkopieren. Mit dieser Methode kann man zum Beispiel auch einen Art Geheimcode entwickeln: Man tauscht einfach die Zeichen im ROM untereinander aus (statt eines A ein B, statt eines U ein X und so weiter). Nach einigem Umdefinieren bekommt man einen ganz eigenartigen Buchstabensalat auf dem Bildschirm.

Taste 9: Bild zeichnen

Nehmen wir an, Sie haben aus den etwas eintönigen Grafikzeichen des C 64 interessantere Zeichen zusammengebastelt: Leitern, Mauerwerke, Treppen und ähnliches. Sie würden aber ganz gerne sehen, wie diese Zeichen im Zusammenhang wirken; wie es etwa aussieht, wenn ein Leiterstück unter dem anderen steht, daneben ein Mauerwerk und so weiter. Dazu drücken Sie Taste 9 und können jetzt mit Ihren selbsterstellten Zeichen den ganzen Bildschirm vollmalen. Wenn Sie wieder ins Menü zurückwollen, drücken Sie einfach »Cursor Home« (steht auch auf dem Bildschirm).

Ein Hinweis zu diesem Programmpunkt: Der Cursor ist beim Bildmalen aus Gründen der Programmiervereinfachung nicht immer sehr gut sichtbar, manchmal »legt« er sich sogar ab, wenn er schnell über den Bildschirm bewegt wird. Das sollte Sie aber nicht weiter stören, schließlich ist diese Programmfunktion nur als Hilfe gedacht für einen schnellen Überblick.

Tasten 6 und 7: Abspeichern und Laden

Wenn Sie eigene Zeichen definiert haben, möchten Sie diesen Zeichensatz vielleicht abspeichern, um ihn später noch einmal verwenden zu können. Drücken Sie einfach Taste 6 und geben einen Dateinamen ein — der kopierte Zeichensatz mit Ihren selbsterstellten Zeichen wird nun auf Diskette gespeichert (nur der Großbuchstaben-Zeichensatz). Er nimmt dabei genau 44 Blöcke auf der Diskette ein. Mit Taste 7 können Sie schließlich einen abgespeicherten Zeichensatz wieder laden (der aktuelle Zeichensatz wird dabei überschrieben).

Sollten Sie einmal versehentlich 6 oder 7 gedrückt haben, so kommen Sie wieder ins Menü zurück, wenn Sie »N« + »RETURN« eingeben!

Taste 5: Daten eingeben

Eine letzte Programmfunktion bietet die Eingabe von Daten, die ein Zeichen definieren. Dies ist eine Alternative zum Konstruieren eines Zeichens im Zeichenfeld. Zu beachten ist, daß nur Zahlen von 0 bis 255 eingegeben werden können (dies wird allerdings vom Programm sichergestellt; negative und Zahlen größer als 255 werden vom Programm nicht angenommen). Nach Eingabe des 8. Bytes erscheint dann die übliche Frage »Welche Taste!« und nach Eingabe dieser Taste das bekannte Sternchenbild auf dem Zeichenfeld.

Die Erklärung der Taste 0 (Programmende) erübrigt sich wohl. Zu erwähnen ist hierbei nur, daß der kopierte Zeichensatz auch nach Beendigung des Programms weiterhin zur Bildschirmgestaltung benutzt wird; auch der Bildschirm selbst sitzt weiterhin an der Adresse 50176 (und nicht wie üblich, bei 1024). Wollen Sie also wieder einen ganz gewöhnlichen Bildschirm mit ganz gewöhnlichen Grafik-Zeichen haben, so geht das am einfachsten nur durch Aus- und Anschalten des Computers (auch die Run/Stop-Restore-Taste wurde im Programm blockiert).

Eine kleine Bemerkung am Rande zum eigenen Programmieren: Sehr oft kommt es ja vor, daß man einen Satz von einem Programm aus mitten in dem Bildschirm hineinschreiben will. Üblicherweise geht man so vor, daß man die Cursor-Steuerzeichen in die Print-Zeile so oft einfügt, bis der gesuchte Platz gefunden ist. Für Programmlistings ist das etwas unübersichtlich. Es gibt aber noch eine andere Methode, die ich in diesem Programm benutzt habe (und zwar am Anfang fast jedes Programmpunktes): will man zum Beispiel einen Buchstaben in die 14. Zeile, 5. Spalte schreiben, so gibt man zunächst

POKE 214,13

ein, danach ein PRINT-Kommando und dann PRINTTAB(4)»X«.

In Adresse 214 steht nämlich die Zeilenposition des Cursors. Um diese richtig einzusetzen, muß man allerdings noch einen PRINT-Befehl nachschicken: Cursor in Zeile 13 + PRINT-Befehl ergibt dann Zeile 14!

(Volker Bühn / gk)

Programmablauf-Plan:

Zeile	Beschreibung
1-7	Programmname und Adresse des Autors
10-30	Einlesen und Aufruf der Maschinenroutine, die den Zeichensatz vom ROM ins RAM an die Adresse 51200 kopiert
40-140	Menü und Zeichenfeld werden gezeichnet
300-460	Abfrage der Tastatur: 1. Zeichnen im Zeichenfeld 2. Aufruf der einzelnen Programmfunktionen
500-520	Unterprogramm zum Printen des Zeichenfeldes
1000-1530	Die einzelnen Unterprogramme zur Steuerung des Cursors, der Space-Taste und des Sternchens, um im Zeichenfeld ein Zeichen zu erzeugen.
2000-8040	**Einzelne Programmfunktionen:**
2000-2120	Berechnung eines erstellten Zeichens; Zuordnung dieses neuen Zeichens in den Zeichensatz; Printen der Daten des Zeichens
3000-3150	Ausgabe eines Zeichens aus dem ROM-Zeichensatz; Printen der Daten dieses Zeichens
4000-4150	Ausgabe eines Zeichens aus dem kopierten Zeichensatz, Printen der Daten dieses Zeichens
5000-5050	Neuer Bildschirm zum Zeichen mit dem selbsterstellten Zeichensatz Die Zeile 5050 dient zur Cursor-Steuerung und Ausdrucken eines Zeichens auf dem Bildschirm
6000-6120	Abspeichern eines selbsterstellten Zeichensatzes auf Diskette
7000-7120	Laden eines Zeichensatzes von Diskette
8000-8040	Eingabe von 8 Bytes zur Definition eines Zeichens In 8030 wird sichergestellt, daß keine Zahl größer als 8 Bit ist
10000-10040	POKEn des Maschinenprogramms nach 828
10100-10150	DATA-Zeilen des Maschinenprogramms (in Hexcodes)

Wichtige Programmzeilen:

Erstellung eines eigenen Zeichens:

2050-2070	Hier werden die Bytes des selbsterstellten Zeichens aus dem Zeichenfeld gelesen, berechnet und in B(1) bis B(8) abgelegt.
2080	Die Bytes des Zeichens werden nun in die Stelle des Zeichensatzes gePOKEt, die der Benutzer durch die Tastenabfrage definiert hat.
	ROM-Zeichen:
3050	Verhindern von Interrupts
3060-3070	Im ROM-Zeichensatz wird das vom Benutzer festgelegte Zeichen in B(1) bis B(8) sowie C(1) bis C(8) abgelegt
3080	Zulassen von Interrupts
3090-3110	Die Bytes B(1) bis B(8) werden in das Zeichenfeld in Form von Sternchen (*) hinein gePOKEt

Eigenes Zeichen holen:

4060-4070	wie 3060-3070, nur daß im kopierten Zeichensatz gesucht wird
4080-4100	wie 3090-3110
	Bild zeichnen:
5050	In den Adressen 209,210 und 211 steht die aktuelle Cursorposition. Dies verhindert, daß der Cursur auf den Bildschirm wie ein normales Zeichen gePRINTet wird.

Liste der wichtigsten Variablen

Z	Position des anfänglichen Cursors im Zeichenfeld (ist immer 50776)
PU$, NO$	Strings, um Punkte beziehungsweise Leerstellen zu PRINTen
A	Variable, die die aktuelle Cursorposition im Zeichenfeld angibt
G$	Stringvariable zur Tastaturabfrage
B(1) bis B(8), C(1) bis C(8)	Arrays, die die 8 Bytes eines Zeichens enthalten
G	Bildschirmcode eines Zeichens
C	Cursorposition beim Bildzeichnen
N$	Name einer einzulesenden beziehungsweise abzuspeichernden Datei
A$	Hexcode aus den DATA-Zeilen
HN,LN	Hi-Nibble beziehungsweise Lo-Nibble von A$
I,J,K	Schleifenvariable

```
1 REM ************************                <128>
2 REM * ZEICHEN-EDITOR/C64    *               <187>
3 REM * VON     VOLKER BUEHN  *                <030>
4 REM *         B2,14         *                <236>
5 REM *         6800 MANNHEIM *                <003>
6 REM * AM      6.7.1984      *                <022>
7 REM ************************                <134>
10 PRINT"[CLEAR]":PRINT:PRINT"[SPACE2]BITTE
   [SPACE]WARTEN!":RESTORE:POKE 657,128
   :POKE 792,134:POKE 793,234                  <062>
20 GOSUB 10000:REM EINLESEN DER
                MASCHINENROUTINE               <069>
30 SYS 828     :REM AUFRUF DERSELBEN           <132>
40 POKE 53280,0:POKE 53281,0
   :PRINT"[CLEAR,SPACE8,RVSON]SELBSTERSTELLTE
   [SPACE]ZEICHEN[RVOFF]"                      <034>
50 PRINT:PRINT"1[SPACE]ZEICHEN[SPACE]BERECHNEN
   [SPACE]6[SPACE]ABSPEICHERN"                 <253>
60 PRINT"2[SPACE]NEUES[SPACE]ZEICHEN[SPACE5]7
   [SPACE]LADEN"                               <108>
70 PRINT"3[SPACE]ROM-ZEICHEN[SPACE7]8[SPACE]
   RESTORE"                                    <211>
80 PRINT"4[SPACE]ZEICHEN[SPACE]HOLEN[SPACE5]9
   [SPACE]BILD[SPACE]ZEICHNEN"                 <133>
90 PRINT"5[SPACE]DATAS[SPACE]EINGEBEN[SPACE4]0
   [SPACE]PROGRAMMENDE"                        <199>
100 Z=50776                                    <121>
110 PU$="........[SPACE31]"                    <157>
120 NO$="[SPACE28]"                            <047>
130 GOSUB 510:IF FL=1 THEN RETURN              <020>
140 A=Z:POKE A,PEEK(A)OR 128                   <182>
300 GET G$:IF G$=""THEN 300                    <153>
310 IF G$="[UP]"THEN GOSUB 1020                <171>
320 IF G$="[DOWN]"THEN GOSUB 1120              <054>
330 IF G$="[RIGHT]"THEN GOSUB 1220             <077>
340 IF G$="[LEFT]"THEN GOSUB 1320              <216>
350 IF G$="[SPACE]"THEN GOSUB 1420             <070>
360 IF G$="*"THEN GOSUB 1520                   <123>
370 IF G$="1"THEN POKE A,PEEK(A)AND 127
   :GOTO 2000                                  <093>
380 IF G$="2"THEN RUN 100                      <093>
390 IF G$="3"THEN 3000                         <016>
400 IF G$="4"THEN 4000                         <028>
405 IF G$="5"THEN 8000                         <038>
410 IF G$="6"THEN 6000                         <042>
420 IF G$="7"THEN 7000                         <054>
430 IF G$="8"THEN 30                           <221>
440 IF G$="9"THEN 5000                         <074>
450 IF G$="0"THEN PRINT"[CLEAR]":END           <176>
460 GOTO 300                                   <233>
500 REM FELD ZEICHNEN                          <243>
510 PRINT"[HOME]":FOR I=1 TO 14:PRINT:NEXT     <110>
520 FOR I=1 TO 8:PRINT PU$:NEXT:RETURN         <179>
1000 :                                         <037>
1010 REM CURSOR NACH OBEN                      <160>
1020 IF PEEK(A-40)=32 THEN RETURN              <057>
1030 POKE A,PEEK(A)AND 127:A=A-40
   :POKE A,PEEK(A)OR 128:RETURN                <205>
1100 :                                         <138>
1110 REM CURSOR NACH UNTEN                     <107>
1120 IF PEEK(A+40)=32 THEN RETURN              <157>
1130 POKE A,PEEK(A)AND 127:A=A+40
   :POKE A,PEEK(A)OR 128:RETURN                <048>
1200 :                                         <238>

1210 REM CURSOR NACH RECHTS                    <014>
1220 IF PEEK(A+1)=32 THEN RETURN               <206>
1230 POKE A,PEEK(A)AND 127:A=A+1
   :POKE A,PEEK(A)OR 128:RETURN                <097>
1300 :                                         <083>
1310 REM CURSOR NACH LINKS                     <043>
1320 IF PEEK(A-1)=32 THEN RETURN               <052>
1330 POKE A,PEEK(A)AND 127:A=A-1
   :POKE A,PEEK(A)OR 128:RETURN                <199>
1400 :                                         <183>
1410 REM SPACE-TASTE IM ZEICHENFELD            <231>
1420 IF PEEK(A+1)=32 THEN POKE A,174:RETURN
                                               <113>
1430 POKE A,46:A=A+1:POKE A,PEEK(A)OR 128:RETURN
                                               <247>
1500 :                                         <027>
1510 REM "*" IM ZEICHENFELD                    <159>
1520 POKE A,42:IF PEEK(A+1)=32 THEN POKE A,
   PEEK(A)OR 128:RETURN                        <124>
1530 A=A+1:POKE A,PEEK(A)OR 128:RETURN         <179>
2000 REM ********************                 <132>
2001 REM *ZEICHEN BERECHNEN*                   <075>
2002 REM ********************                 <134>
2010 POKE 214,14:PRINT:PRINT TAB(14)"WELCHE
   [SPACE]TASTE?[SPACE]";                      <110>
2020 GET G$:IF G$=""THEN 2020                  <136>
2030 PRINT G$:G=PEEK(Z+28)*8:K=0:A=Z           <127>
2040 PRINT TAB(10);NO$:PRINT TAB(10);NO$
   :PRINT"[UP4]"                               <024>
2050 FOR J=1 TO 8:FOR I=A+7 TO A STEP-1        <214>
2060 IF PEEK(I)=42 THEN B(J)=B(J)+2↑K          <251>
2070 K=K+1:NEXT I:A=A+40:K=0:NEXT J            <047>
2080 FOR I=51200+G TO 51207+G:K=K+1:POKE I,B(K)
   :NEXT                                       <062>
2090 PRINT TAB(10);NO$                         <244>
2100 PRINT TAB(15);B(1);B(2);B(3);B(4)         <009>
2110 PRINT TAB(15);B(5);B(6);B(7);B(8)         <035>
2120 POKE(Z+11),G/8:FOR I=1 TO 8:B(I)=0:NEXT
   :FL=0:GOTO 140                              <115>
3000 REM **************                        <158>
3001 REM * ROM-ZEICHEN*                       <200>
3002 REM **************                        <160>
3010 GOSUB 510                                 <240>
3020 POKE 214,14:PRINT:PRINT TAB(14)"WELCHE
   [SPACE]TASTE?[SPACE]";                      <100>
3030 GET G$:IF G$=""THEN 3030                  <128>
3040 K=0:A=Z:PRINT G$:G=PEEK(Z+28)*8           <117>
3050 POKE 56334,PEEK(56334)AND 254
   :POKE 1,PEEK(1)AND 251                      <216>
3060 FOR I=53248+G TO 53255+G                  <011>
3070 K=K+1:B(K)=PEEK(I):C(K)=B(K):NEXT:K=0     <030>
3080 POKE 1,PEEK(1)OR 4:POKE 56334,
   PEEK(56334)OR 1                             <043>
3090 FOR J=1 TO 8:FOR I=7 TO 0 STEP-1          <238>
3100 IF B(J)>=2↑I THEN B(J)=B(J)-2↑I
   :POKE(A+7-I),42                             <205>
3110 NEXT I:A=A+40:NEXT J                      <127>
3120 PRINT"[UP]"TAB(10);NO$                    <211>
3130 PRINT TAB(15);C(1);C(2);C(3);C(4)         <023>
3140 PRINT TAB(15);C(5);C(6);C(7);C(8)         <049>
3150 FOR I=1 TO 8:B(I)=0:NEXT:GOTO 140         <239>
4000 REM ************************              <088>
4001 REM * EIGENES ZEICHEN HOLEN *             <015>
4002 REM ************************              <090>
4010 GOSUB 510                                 <220>
4020 POKE 214,14:PRINT:PRINT TAB(14)"WELCHE
   [SPACE]TASTE?[SPACE]";                      <080>
4030 GET G$:IF G$=""THEN 4030                  <109>
4040 PRINT G$:G=PEEK(Z+28)*8:K=0:A=Z           <097>
4050 IF FL=1 THEN 4080                         <084>
4060 FOR I=51200+G TO 51207+G                  <228>
4070 K=K+1:B(K)=PEEK(I):C(K)=B(K):NEXT:K=0     <010>

4080 FOR J=1 TO 8:FOR I=7 TO 0 STEP-1          <207>
4090 IF C(J)>=2↑I THEN C(J)=C(J)-2↑I
   :POKE(A+7-I),42                             <177>
4100 NEXT I:A=A+40:NEXT J                      <097>
4110 PRINT"[UP]"TAB(10);NO$                    <181>
4120 IF FL=1 THEN 2080                         <153>
4130 PRINT TAB(15);B(1);B(2);B(3);B(4)         <255>
4140 PRINT TAB(15);B(5);B(6);B(7);B(8)         <025>
4150 POKE(Z+11),G/8:FOR I=1 TO 8:B(I)=0:NEXT
   :GOTO 140                                   <187>
5000 REM ****************                      <202>
5001 REM *  BILD MALEN  *                      <007>
5002 REM ****************                      <204>
5010 PRINT"[CLEAR]";:POKE 53280,0:POKE 53281,1
   :POKE 646,0:POKE 650,128                    <176>
5020 PRINT"[HOME=ZURUECK]"                     <211>
5030 POKE 204,0:GET G$:IF G$=""THEN 5030       <029>
5040 IF G$="[HOME]"THEN POKE 204,1:POKE 646,6
   :POKE 650,0:GOTO 40                         <181>
5050 C=PEEK(209)+PEEK(210)*256+PEEK(211)
   :POKE C,PEEK(C)AND 127:PRINT G$;:GOTO 5030
                                               <061>
6000 REM *************                         <056>
6001 REM * SPEICHERN *                         <012>
6002 REM *************                         <058>
6010 PRINT"[CLEAR]":PRINT:PRINT"[SPACE7]
   SPEICHERN[SPACE]DES[SPACE]ZEICHENSATZES"
                                               <072>
6020 PRINT:PRINT"(INPUT[SPACE]'N'[SPACE],[SPACE]
   WENN[SPACE]ZURUECK)"                        <085>
6030 PRINT:PRINT:PRINT"[SPACE2]NAME[SPACE]DER
   [SPACE]DATEI?"                              <202>
6040 INPUT N$                                  <166>
6050 IF N$="N"THEN 40                          <005>
6060 OPEN 2,8,2,N$+",S,W"                      <228>
6070 CLOSE 1:OPEN 1,8,15:INPUT#1,OA
   :IF OA<>0 THEN 6010                         <160>
6080 FOR I=51200 TO 53248                      <245>
```

```
6090 S%=PEEK(I)                                    <103>
6100 PRINT#2,S%                                    <089>
6110 NEXT I                                        <192>
6120 CLOSE 2:GOTO 40                               <248>
7000 REM *********                                 <124>
7001 REM * LADEN *                                 <187>
7002 REM *********                                 <126>
7010 PRINT"[CLEAR]":PRINT:PRINT"[SPACE7]LADEN
   [SPACE]EINS[SPACE]ZEICHENSATZES"                <074>
7020 PRINT:PRINT"(INPUT[SPACE]'N',[SPACE]WENN
   [SPACE]ZURUECK)"                                <065>
7030 PRINT:PRINT:PRINT"[SPACE2]NAME[SPACE]DER
   [SPACE]DATEI?"                                  <182>
7040 INPUT N$                                      <146>
7050 IF N$="N"THEN 40                              <241>
7060 OPEN 2,8,2,N$+",S,R"                          <203>
7070 CLOSE 1:OPEN 1,8,15:INPUT#1,OA
   :IF OA<>0 THEN 7010                             <141>
7080 FOR I=51200 TO 53248                          <225>
7090 INPUT#2,S%                                    <039>
7100 POKE I,S%                                     <091>
7110 NEXT I                                        <172>
7120 CLOSE 2:GOTO 40                               <228>
8000 REM ******************                        <226>
8001 REM * DATAS EINGEBEN *                        <237>
8002 REM ******************                        <228>
8010 PRINT"[CLEAR]":PRINT:PRINT"[SPACE]DATEN
   [SPACE]FUER[SPACE]EIN[SPACE]ZEICHEN[SPACE]
   EINGEBEN"                                       <128>
8020 FOR I=1 TO 8:PRINT I".[SPACE]BYTE"
   :INPUT B(I):C(I)=B(I)                           <254>
8030 IF B(I)>255 OR B(I)<0 THEN I=I-1              <103>
8040 NEXT:FL=1:GOSUB 40:GOTO 4010                  <107>
10000 REM ***************************              <094>
10001 REM * MASCHINENROUTINE-EINGABE *             <239>
10002 REM ***************************              <096>
10010 FOR I=0 TO 58:READ A$                        <036>
10020 HN=(ASC(LEFT$(A$,1))-48)*16
   :IF HN>144 THEN HN=HN-112                       <105>
10030 LN=ASC(RIGHT$(A$,1))-48
   :IF LN>9 THEN LN=LN-7                           <099>
10040 HN=HN+LN:POKE(828+I),HN:NEXT:RETURN          <126>
10100 DATA 78,A9,31,85,01,A9,00,85,64,A9           <240>
10110 DATA D0,85,63,A9,C8,85,65,A2,10,A0           <016>
10120 DATA 00,B1,62,91,64,C8,D0,F9,E6,63           <031>
10130 DATA E6,65,CA,D0,F2,A9,37,85,01,58           <061>
10140 DATA A9,12,8D,18,D0,A9,94,8D,00,DD           <083>
10150 DATA A9,C4,8D,88,02,20,44,E5,60              <153>
```

Listing »Zeichen-Editor«

Super Line — 80 Zeichen für den C 64

Ein kleiner Basic-Lader realisiert, wofür man sonst viel Geld ausgeben muß: 80 Zeichen pro Zeile

Es werden 4 neue Befehle definiert, die das Darstellen von 80 Zeichen möglich machen. Und das, ohne daß man lange programmieren muß.

Beginnen wir mit dem Einfachsten: dem Eingeben. Dies dürfte keine Schwierigkeiten bereiten. Bevor man das Quellenprogramm jedoch startet, sollte man es abspeichern, da es sich, vorausgesetzt, man hat keinen Fehler gemacht, selbst löscht. Anschließend kann man es mit RUN laufen lassen und wenn die Prüfsumme stimmt, erscheint nach einigen Sekunden einfach READY. Die neuen Befehle sind nun definiert und können angewendet werden.

Diese Befehle lauten:

O (für »ON«):	dieser Befehl bewirkt ein Einschalten des 80-Zeichen Modus. Dabei wird auf hochauflösende Grafik umgeschaltet.
C (für »CLEAR«):	der 80-Zeichen Bildschirm wird gelöscht.
W x,y,a$ (für »WRITE«):	der String A$ wird an Spalte x Zeile y geschrieben. x geht von 0 bis 79, y geht von 0 bis 24.
F (für »OFF«):	Abschalten des 80-Zeichen Modus.

So bewirkt zum Beispiel das kurze Programm:

```
10 O
20 C
30 W 0,0,"64'ER DAS MAGAZIN FÜR COMPUTER-FANS"
```

daß der in Anführungsstrichen stehende Satz in die linke obere Ecke geschrieben wird. Anschließend rührt sich nichts mehr und man kann durch die 'blinde' Eingabe von 'F' wieder zum normalen Bildschirm zurückkehren.

A$ läßt sich auch durch einzelne Stringvariable ersetzen, die mit + verknüpft werden, oder man kann auch eine normale numerische Variable verwenden. Allerdings dürfen keine Variable mit dem Namen O,C,W oder F verwendet werden. So ist zum Beispiel O$,CG,WR$ oder ähnliches verboten.

Nun einige detaillierte Angaben zum Programm selbst. Das Maschinenprogramm liegt im Bereich von $ 9000 bis $ 928F. Wer im Besitz eines Monitors ist, kann es direkt abspeichern und von der Diskette mit ,8,1 laden (danach »NEW« eingeben). Gestartet wird es dann mit SYS 36864. Wer das nicht will, der lädt einfach den Basic-Lader, der das Maschinenprogramm in den Speicher 'POKEt'. Zwischen $ 9000 und $ 902D wird zunächst der Basic-Vektor umgesetzt und der Anfang des Basic-Speichers hochgelegt. Zwischen $ 902E und $ 9044 beginnt nun die Befehlsdecodierung. Bei Erkennen eines Befehls wird verzweigt, ansonsten in die normale Interpreterroutine gesprungen. Bei dem Befehl 'O' wird zunächst in der Unterroutine von $ 9233 bis 928E der Zeichensatz aus dem verdeckten Bereich $ D000 in den offenen Bereich $ C000 übertragen. Der Bereich von $ 0400 bis $ 0800 wird mit dem Code für die Hintergrundfarbe gefüllt. Außerdem wird bei $ 9057 das Register # 648 umgesetzt, damit es auf dem Bildschirm kein farbliches Durcheinander gibt. Weiterhin wird selbstverständlich der hochauflösende Grafik-Modus eingeschalten. Die Routine für den Befehl 'C' liegt zwischen $ 9081 und $ 90A3. Der Bereich der Bit-Map wird einfach mit 00 gefüllt.

Der Befehl 'F' wird zwischen $ 906C und $ 907E ausgeführt. Das Register 648 wird zurückgesetzt, der hochauflösende Grafik-Modus ausgeschaltet und der normale Bildschirm gelöscht.

Der Befehl, dessen Routine am längsten ist, ist der Befehl 'W'. Er wird zwischen $ 90A6 und $ 9230 bearbeitet. Zunächst werden die beiden Koordinaten x und y geholt und aus ihnen die Adresse der Bit Map berechnet, an der das erste Byte gesetzt wird. Dies geschieht zwischen $ 90A6 und 9135. Dann werden die einzelnen Zeichen des zu schreibenden Satzes geholt und ihr Code wird so umgerechnet, daß er mit der Stelle übereinstimmt, an der das jeweilige Zeichen in dem nach $ C000 verschobenen Zeichen ROM steht. Anschließend durchläuft jedes der 8 Bytes, aus denen ein Zeichen definiert ist, die gleiche Prozedur. Das Byte wird geholt, jedes zweite Bit ausgefiltert, und die verbliebenen 4 Bits zusammengeschoben. Das Zeichen ist jetzt nur noch durch 4 x 8 Punkte definiert. Jetzt müssen die entstandenen Nibbles noch in die Bytes der Bit Map gebracht werden. Dies geschieht mit einer EXOR-Verknüpfung. Dabei steuert ein Flag, das in $ 9300 steht, ob das Nibble in die linke oder die rechte Hälfte des Bytes geschrieben wird.

(Andreas Zell / rg)