Tips & Tricks für Profis

Haben Sie schon einmal daran gedacht, einen Stack in Basic zu simulieren? Oder die NMI-Routine umzuprogrammieren? Wir zeigen Ihnen neben diesen beiden Tricks und vielen weiteren auch eine Tabelle mit einigen interessanten Systemadressen.

Kleine Ursache, große Wirkung: Im Leserforum der Ausgabe 8/86 der 64'er fragte ein Leser nach der »größten Primzahl der Welt«. Seitdem treffen ständig mehr oder weniger kurze Listings ein, die sich mit Primzahlen auseinandersetzen. Ohne Zweifel, ein heißes Thema für Computerfreunde. Bleibt abzuwarten, wie weit diese Entwicklung geht. Vielleicht haben Sie ja auch einmal ein kniffliges Programm zum Thema Primzahlen geschrieben. Schicken Sie es an die 64'er! Interessante Lösungsmöglichkeiten werden wir veröffentlichen. (tr)

Star SG-10 und Bar-Codes

Nach dem Umschreiben auf Star SG-10 und anschließendem Probelauf sowie einen Vergleich mit tatsächlich vergebenen EAN-Codes, bin ich zu der Überzeugung gelangt, daß in Zeile 85 ein Fehler vorhanden ist. Meines Erachtens müßte der dritte DATA-Wert »1000100« lauten, statt wie angegeben, »1001000«. Dieses wäre schon allein durch die Tatsache erklärt, daß der Zeichensatz C das Spiegelbild des Zeichensatzes B, beziehungsweise die Invers-Darstellung von Zeichensatz A darstellt.

Übrigens läßt sich die Höhe der ausgedruckten Balken durch Ändern der Variablen RC in Zeile 52 reduzieren (zum Beispiel RC = 5), was für die meisten Anwendungen ausreichen dürfte und beim Ausdruck eine bedeutende Zeitersparnis bedeutet.

Bild 1 zeigt die Programm-Änderungen für den Drucker Star SG-10 sowie der berichtigten Zeile 85 und einem Probeausdruck. Da das von mir verwendete Star-Secus-Interface die Steuerzeichen im Klartext ausdruckt, ist statt des angegebenen »ESC« jeweils <CTRL+:> und statt »($1)« ist <CTRL+A> einzugeben, wie auch bereits im Artikel erwähnt.

(Markus Raute/tr)

```
50  :
51  OPEN1,4,4
52  RC=5 :IFEAN=8THENRC=5
53  FORC=1TOE:PRINT#1,"(ESC)@(ESC)M"CHR$(15);
54  FORCO=1TORC:FORC1=1TO2
55  PRINT#1,D$(0)D$(1)D$(2);
56  IFC1=1THENPRINT#1,"(ESC)3($1)"CHR$(13)CHR$(10);:
    GOTO58
57  IFCO<RCTHENPRINT#1,"(ESC)1"CHR$(13)CHR$(10);
58  NEXT:NEXT
59  :
60  PRINT#1,"(ESC)2"CHR$(13)CHR$(10)"(ESC)B"CHR$(2)
    "(ESC)G";
61  IFEAN=8THENPRINT#1," (ESC)W"CHR$(1)LEFT$(N$,4)
    " (ESC)W"CHR$(0)" (ESC)W"CHR$ (1)RIGHT$(N$,4);:
    GOTO64
62  PRINT#1,"(ESC)W"CHR$(1)LEFT$(N$,2)"(ESC)W"CHR$
    (0)" (ESC)W"CHR$(1)MID$(N$,3,5)"(ESC)W"CHR$
    (0)" (ESC)W"CHR(1);
63  PRINT#1,MID$(N$,8,5)"(ESC)W"
    CHR$(0)" (ESC)W"
    CHR$(1)RIGHT$(N$,1);
64  PRINT#1,CHR$(13)CHR$(10)CHR$(10);:NEXT:PRINT#1,
    "(ESC)@":CLOSE1:RUN
65  :

95  D$(0)=CHR$(27)+CHR$(103)+CHR$(3)+CHR$(58)+CHR$(2):
    D$(1)="":D$(2)=""

85  DATA"0111011","0010001","1000100"
```

Bild 1. Diese Zeilen müssen Sie im »Bar-Codes«-Listing ändern, um mit dem Star SG-10 arbeiten zu können

Ku-Ba-Graf und Epson-Drucker

Um das Programm Ku-Ba-Graf aus dem Grafik-Sonderheft 6/86 an einen (Epson-kompatiblen) Drucker anzupassen, der am seriellen Bus betrieben wird, sind nur ein paar Änderungen nötig.

Zuerst »Epson Support« laden:

```
LOAD"EPSON-SUPPORT",8,1
NEW
```

Durch die folgenden drei POKEs wird die Ausgaberoutine auf eine Betriebssystemroutine (CHROUT $FFD2) umgeleitet:

```
POKE 49649,76
POKE 49650,210
POKE 49651,255
```

Wer ein »Auto-Line-Feed« an seinem Drucker eingeschaltet lassen will, gibt noch folgenden POKE ein:

```
POKE 50326,0
```

Jetzt kann das geänderte »Epson-Support« (mit einem Monitor) gespeichert werden. Wem das zu kompliziert ist, kann die vier POKEs auch an den Anfang des Basic-Hauptprogramms stellen.

Um die Änderungen am Maschinenprogramm möglichst gering zu halten, muß nun noch das Hauptprogramm verändert werden:

```
547 OPEN1,4:CMD1:&PRINT:PRINT#1:CLOSE1
880 OPEN1,4:CMD1:&PRINT:PRINT#1:CLOSE1:RETURN
```

In Zeile 550 muß &PRINT, in Zeile 870 &PRINT und RETURN gelöscht werden.

Jetzt kann das geänderte Programm gespeichert werden.

Nach diesen Änderungen funktioniert Ku-Bu-Graf zumindest auf meinen NL-10 einwandfrei. (Stefan Blomen/tr)

Hohe Primzahlen

Zu dem Leserbrief in der 64'er 8/86 auf Seite 18 über Primzahlen habe ich mir Gedanken gemacht und ein kurzes Primzahlprogramm entwickelt, welches in relativ kurzer Rechenzeit viele Primzahlen auf dem Bildschirm ausdruckt. Wenn dieses Programm (Listing 1) compiliert wird, läuft es etwa um 50 % schneller. Es werden zwar keine solch hohen Primzahlen erreicht wie in dem genannten Beispiel, dazu wäre auch der Speicherplatz zu gering, dafür nutze ich aber die gesamte Rechengenauigkeit des C 64 aus.

Der Befehl »INT (A)« in Zeile 20 wird deswegen gebraucht, da der C 64 sonst mit seiner Rechenungenauigkeit immer das falsche Ergebnis herausbekommt.

Das Programm wurde kurz gehalten, damit es möglichst schnell ablaufen kann.

(Ingo Godau/tr)

```
20 Q=3500:C=0:A=0:E=0:F=0:D=0:PRINT 1,2,3,:DIM
   B(Q):B(0)=3:FOR A=3 TO 999999999 STEP 2        <180>
30 IF C<Q THEN IF INT(A)=INT(B(E)↑2)THEN E=E+1    <229>
40 FOR D=3 TO B(E)STEP 2:IF A/D=INT(A/D)THEN F=
   1:D=B(E)                                       <119>
50 NEXT:IF F=1 THEN F=0:NEXT                      <066>
60 PRINT A,:IF C<Q THEN C=C+1:B(C)=A:NEXT         <105>
70 NEXT                                           <080>

© 64'er
```

Listing 1. Schnelles Primzahlenprogramm in Basic

Stack in Basic

Im folgenden will ich eine Programmier-Strategie vorstellen, die die Einrichtung vieler Variablen überflüssig macht und die sogar die rekursive Programmierung bequem unterstützt.

Die Hauptschwierigkeit beim Programmieren in Basic besteht darin, daß für jede Kleinigkeit eine neue Variable eingerichtet werden muß. In vielen Fällen würde es reichen, einen Wert einfach auf einem Stack abzulegen und später wieder zurückzuholen. Basic hat aber keinen Datenstapel.

Wirklich nicht? Eigentlich besteht ein solcher Stapel doch nur aus einem eindimensionalen Datenfeld und einer Zeigervariablen: beides Elemente, die von Basic unterstützt werden. Warum machen wir uns nicht einfach selber einen?

Mit DIM STACK(255) wird das Array eingerichtet; die Tiefe 255 ist willkürlich gewählt. Der zugehörige Zeiger sei P=0.

Der Befehl push X (»lege den Wert von X auf dem Stack ab«) wird folgendermaßen nachgebildet:

```
P=P+1:STACK(P)=X;
```

der Befehl pop X (»hole den Wert für X vom Stack«) so:

```
X=STACK(P):P=P-1.
```

Der Datenstack bildet eine sehr einfache Möglichkeit, Parameter an Subroutinen zu übergeben, oder aber auch Funktionen mit echter Fallunterscheidung zu simulieren (die Parameter-Eingabe wie auch die -Ausgabe steht stets auf dem Stack!).

Listing 2 demonstriert anhand der allseits beliebten Fakultätsberechnung, daß mit Hilfe des Datenstacks in Basic sogar rekursiv programmiert werden kann. Nochmal ganz kurz zur Fakultätsdefinition:

```
10 DIM STACK(255):REM SOFT-STACK                  <158>
20 P=0                                            <237>
30 INPUT"FAKULTAET VON";X                         <114>
40 IF X>20 THEN 30                                <235>
50 STACK(P)=X:REM "PUSH"STACK                     <029>
60 GOSUB 100                                      <004>
70 Y=STACK(P):REM "PULL"STACK                     <131>
80 PRINT X"! ="Y                                  <119>
90 END                                            <092>
99 :                                              <075>
100 IF STACK(P)<2 THEN STACK(P)=1:RETURN          <038>
110 P=P+1:REM STACKPOINTER HOCHSTELLEN            <031>
120 STACK(P)=STACK(P-1)-1                         <003>
130 GOSUB 100:REM SELBSTAUFRUF. DESHALB NU
    R BIS !20                                     <164>
140 STACK(P-1)=STACK(P)*STACK(P-1)                <138>
150 P=P-1:REM POINTER WIEDER RUNTERSTELLEN        <203>
160 RETURN                                        <218>

© 64'er
```

Listing 2. Eine Fakultätsberechnung mit simuliertem Stack

Def fak(n): wenn n<2, dann übergib (1), andernfalls übergib (n*fak(n-1)).

Diese Strategie des selbstprogrammierten Stacks löst nun längst nicht alle Schwierigkeiten in Rauch auf: Beispielsweise bleibt es bei der begrenzten Anzahl von GOSUB-Ebenen (21). Aber es wirkt dem »Spaghetti-Charakter« vieler Programme entgegen. (W.D. Busch/tr)

Die (neue) NMI-Routine

Jeder kennt sie! Sie wird durch das Drücken der Tasten <RUN/STOP+RESTORE> aktiviert. Die NMI-Routine ist nichts Neues, auch nicht, daß sie bestimmte Vektoren wieder zurücksetzt, was nicht immer erwünscht ist.

Da man in jeder zweiten 64'er (Tips und Tricks) lesen kann »... und nach dem Drücken von <RUN/STOP+RESTORE> muß das Programm wieder mit SYS xxxxx aktiviert werden...«, gebe ich hier eine kleine Hilfestellung.

Den oben beschriebenen Effekt vermeide ich, indem ich den wichtigsten Teil der Original-NMI-Routine kopiere und an deren Schluß ein kleines Initialisierungsprogramm für meine »eigenen Vektoren« und sonstige Einstellungen setze. Ich erkläre es besser an einem Beispielprogramm, das bei $1000 beginnt (am besten gleich den SMON laden). Mit der folgenden Mini-Routine »verbiege« ich den sogenannten NMI-Vektor.

```
1000 lda#$0b   ;  lo-byte
1002 ldx#$10   ;  hi-byte
1004 sta $0318 ;  NMI-Vektor, lo
1007 stx $0319 ;  NMI-Vektor, hi
100a rts
```

Die Hauptroutine sieht so aus:

```
100b pha
100c txa
100d pha
100e tya
100f pha
1010 lda #$7f
1012 sta $dd0d
1015 jsr $f6bc
1018 jsr $ffe1
101b jsr $fd15
101e jsr $fda3
1021 jsr $e518
1024 jsr $1000 ; den schon wieder verstellten NMI-Vektor
                 reininitialisieren
1027 jsr $102d ; neue Einstellungen
102a jmp ($a002)
```

Hier endet der kopierte Teil.

```
102d lda#$00 ;hier steht der eigene Teil
102f sta $d020 ;der NMI-Routine
1032 sta $d021
1035 rts
```

Unser Beispiel färbt den Hintergrund und den Rahmen schwarz. Probieren Sie es aus: SYS 4096 und dann <RUN/STOP+RESTORE> drücken!

Eine neue NMI-Routine ist auch dann notwendig, wenn Funktionstasten, die über den IRQ-Vektor programmiert sind, nach <RUN/STOP+RESTORE> ohne »aktivieren durch SYS xxxxx« noch funktionieren sollen.

Die »Einstellroutine« kann dann so aussehen:

```
102d lda#$38
102f ldx#$10
1031 sta $0314
1034 stx $0315
1037 rts
```

```
1038 lda $cb   ; Code der gerade gedrückten Taste
103a cmp #$40  ; $40 = keine Taste gedrückt
103c bne $1043 ; irgendeine Taste gedrückt, dann weiter
103e sta $02   ; merken, um Repeat-Funktion zu verhindern
1040 jmp $ea31 ; zur normalen IRQ-Routine
1043 cmp #$04  ; (F1) gedrückt ?
1045 bne $103e ; nein, normaler IRQ
1047 cmp $02   ; dieselbe Taste wie vorher ?
1049 beq $103e ; ja, dann normaler IRQ
104b sta $02   ; merken, um Repeat-Funktion zu verhindern
104d lda #$52  ; "R"
104f ldx #$55  ; "U"
1051 ldy #$4e  ; "N"
1053 sta $0277
1056 stx $0278 ; in Tastaturpuffer schreiben
1059 sty $0279
105c lda #$0d   ; "Return"
105e ldx #$04   ; vier Zeichen im Tastaturpuffer
1060 sta $027a
1063 stx $c6
1065 jmp $ea31 ; normal weitermachen
```

Auf die Programmierung der Funktionstasten will ich hier nicht weiter eingehen, da ich damit ja nur ein sinnvolles Beispiel für eine neue NMI-Routine geben wollte.

Das Herz jeder neuen NMI-Routine ist das Programm von $100b bis $102c. (Gerhard Muth/tr)

Tips & Tricks Mischmasch

Zuerst einige Tips für Basic-Programmierer, die auch für Assemblerfans interessant sind

- SYS 58778	Initialisiert den Videochip neu
- SYS 64931	Initialisiert den gesamten Interrupt
- SYS 59626	Scrollt den Bildschirminhalt um eine Zeile nach oben
-SYS 58592	wartet 8,5 Sekunden, bei Druck auf die Commodoretaste wird abgebrochen (wird benutzt beim Laden von Datasette (FOUND...)
- SYS 63544	Gibt »PRESS RECORD & PLAY ON TAPE« aus und wartet, bis die <Play>-Taste der Datasette gedrückt ist
- SYS 63511	Wie SYS 63544, jedoch wird nur auf die Play>-Taste gewartet (gibt »PRESS PLAY ON TAPE« aus)
- SYS 64767	Bewirkt einen Basic Kaltstart
- POKE 214,x: POKE211,y: SYS 58732	Setzt den Cursor auf Zeile x und Spalte y
- ?PEEK(174) +PEEK(175) *256	Gibt nach dem Laden eines Programmes die Endadresse des Programmes aus
- ?PEEK(187) +PEEK(188) *256	Gibt die Anfangsadresse des zuletzt benutzten Filenamens aus
- ?PEEK(648) *256	Gibt die Anfangsadresse des Bildschirmspeichers aus
- POKE 649,0	Schaltet die Tastatur »aus« (Setzen der Tastaturpuffergröße auf Null)
- POKE 198,0	Löscht den Tastaturpuffer
- POKE 157,128 (0)	Blockiert (erlaubt) die Ausgabe der Meldungen »SEARCHING FOR« etc.
- A=PEEK(10)	Falls A=0, dann war der zuletzt benutzte Befehl, der die Floppy oder Datasette anspricht, der LOAD-Befehl, andernfalls der VERIFY-Befehl (SAVE wird nicht berücksichtigt)
- POKE 781,x: SYS 59903	Löscht Bildschirmzeile x
- POKE 808,230	Schaltet die <RUN/STOP>-Taste aus, wobei die Funktion von TI$ und TI erhalten bleibt

Nun folgen ein paar Tips für die Assemblerfreaks:

- lda #01 sta $c7	<REVERS ON>
- jsr $e544	Löscht den Bildschirm
- jsr $e566	Setzt den Cursor in die linke obere Ecke (<HOME>)
- lda #l ldy #h	Gibt den Text aus, der im ASCII-Code Adresse steht, die in Low- und Highbyte angegeben ist.
jsr $ab1e	Der Computer bricht ab, wenn er auf ein Nullbyte oder ein Gänsefüßchen stößt.
- ldx #l lda #h jsr $bdcd	Gibt eine 16-Bit-Zahl dezimal auf dem Bildschirm aus, die in Low- und Highbyte angegeben werden muß
- ldx #a ldy #b	Setzt den Cursor an Zeile a und Spalte b
jsr $e50c	
- jmp $febc	Alle Register werden vom Stapel geholt, mit anschließendem RTI. Gut zu gebrauchen für eigene Interrupt-Programme (Rasterinterrupt etc.)
- ldx #a jsr $e9f0	Holt die Anfangsadresse der Bildschirmzeile a im Bildschirmspeicher und legt diese in $D1/$D2 ab. Bei Einsprung in Adresse $EA24 erfolgt das gleiche wie oben, aber diesmal für den Farbspeicher, die Werte werden in $F3/$F4 abgelegt.

(Christian Seiler/tr)

Exzesse über den Filenamen

Es ist wie mit den meisten Schutztricks. Einmal veröffentlicht, schon sind sie nicht mehr effizient, weil sie nun jeder kennt. So auch mit dem »LOAD-Schutz einmal anders« aus der 64'er 6/86. Stefan Bültenas Frage kann ich daher nur mit »ja« beantworten! Diesen Trick findet man heute in fast jeder Computerzeitschrift. Niemand aber erwähnt, daß es auch noch eine ganze Palette von anderen Möglichkeiten gibt, einem File den Namen »" "« zu geben. Bis heute habe ich 74 ohne, und weit über 100 Arten mit optischer Veränderung des Directorys ausprobiert. Die Wahrscheinlichkeit, jetzt den richtigen Namen herauszufinden, ist nun um ein Vielfaches geringer. Mit etwas Ausdauer lassen sich bestimmt noch mehr Wege finden, zwei Anführungszeichen als Namen zu benutzen. Nun aber zu den angekündigten Varianten:

1. Die ersten 15:
SAVE CHR$(34)+"{1 bis 15 Space}",8
2. Die zweiten 15:
SAVE"<SHIFT+SPACE>{1 bis 15 Space}",8
3. Weitere 10 Stück:
a) SAVE CHR$(34)+"<SPACE>",8
b) +"<SHIFT+SPACE>",8
c) +"<CRSR>-links",8
d) +"<CRSR>-rechts",8
e) +<SHIFT+SPACE>"<CRSR>-links",8
f) +<SHIFT+SPACE>"<CRSR>-rechts",8
g) +"<SPACE><SHIFT+SPACE><SPACE>",8
h) "<SHIFT+SPACE><CRSR>-links",8
i) "<SHIFT+SPACE><CRSR>-rechts",8
j) "<SPACE><SHIFT+SPACE><SPACE>",8
4. Die bekanntesten:
a) SAVE CHR$(34),8
b) SAVE"<SHIFT+SPACE>",8

5. Die Supernamen (unlöschbar!):
a) SAVE CHR$(34)+CHR$(58)+CHR$(34) ,8
b) SAVE"<SHIFT+SPACE>"+CHR$(58)+"<SHIFT+SPACE>",8
6. Die drei gleichen:
a) SAVE CHR$(34)+CHR$(44)+CHR$(34) ,8
b) SAVE CHR$(34)+CHR$(44)+"<SHIFT+SPACE>" ,8
c) siehe 5a!
7. Nochmal 13 Arten:
SAVE"<SHIFT+SPACE>"+CHR$(58)+"<SHIFT+SPACE>{1 bis 15 Space}",8
8. Diese drei gibt's auch:
a) SAVE"<SHIFT+SPACE>"+CHR$(44)+"<SHIFT+SPACE>" ,8
b) SAVE CHR$(34)+CHR$(58)+CHR$(34)+CHR$(44)+CHR$(34)" ,8
c) bei b) das mittlere CHR$(34) durch <SHIFT+SPACE> ersetzen
9. Die letzten 14:
SAVE CHR$(34)+"<SHIFT+SPACE>{1 bis 14 Space}" ,8
Bemerkungen zu den Speicherarten:

Zu Punkt 6: Hat man einen dieser Namen benutzt, so lassen sich die anderen nicht mehr anwenden! Zum Laden läßt sich statt CHR$(44) auch CHR$(58) verwenden, und umgekehrt! Beim Speichern ist es genau das gleiche Spielchen.

Zu den optischen Veränderungen benutzt man die <DEL>-Taste. Aber auch <SHIFT+SPACE> in Verbindung mit den Funktionstasten tut das seine dazu.

Ich wünsche noch viel Spaß beim Ausprobieren und Weiterknobeln! (Marco Richter/tr)

Betrifft: »Die schnellste Grafikerweiterung für den C 64«

Im 64'er Sonderheft 6/86 wurde auf Seite 96 die schnellste Grafikerweiterung für den C 64 vorgestellt. Dieses Programm erlaubt die Programmierung von wirklich schneller HiRes-Grafik in Basic.

Ein Leser teilte uns jedoch mit, daß in der abgedruckten Basic-Erweiterung »gefährliche« Zeropage-Adressen verwendet werden, die bei komplexen Funktionen als Parameter für die Grafik-Routinen zu einem »Absturz« des Computers führen können.

Im folgenden ist deshalb ein Programm abgedruckt, das in mühevoller (7minütiger) Kleinarbeit alle diese Zeropage-Adressen durch ungefährliche Werte ersetzt. Laut Autor läuft »High-Speed-Grafik« danach einwandfrei.

Zur Bedienung:

Laden Sie die Grafik-Erweiterung aus dem Sonderheft und tippen Sie »NEW« ein. Danach laden Sie das Listing 3 und starten es mit »RUN«. Legen Sie außerdem eine Diskette in das Diskettenlaufwerk, damit die korrigierte Version gespeichert werden kann. Nach zirka sieben Minuten sind 237 Adressen geändert worden (wird vom Programm angezeigt). Die korrigierte Version von »High-Speed-Grafik« wird danach automatisch mit dem Namen »HSG2« auf die eingelegte Diskette geschrieben.

(W. Lohwasser/ks/tr)

```
100 REM  HSG-W  MODIFIZIERUNG              <040>
110 REM  AENDERT VERWENDETE ZERO-PAGE      <187>
120 REM  ADRESSIERUNG                      <195>
130 REM                                    <192>
140 REM  ERMOEGLICHT VERWENDUNG VON        <114>
150 REM  FUNKTIONEN ALS ARGUMENTE          <092>
160 REM  DER BEFEHLE.                      <060>
170 REM                                    <232>
180 REM  ACHTUNG ! DAUER CA 7 MIN          <026>
190 REM                                    <252>
1000 FOR I=36864 TO 40032                  <055>
1010 Q=PEEK(I)                             <233>
1020 IF Q=164 OR Q=165 OR Q=166 THEN 3000  <005>
1050 IF Q=132 OR Q=133 OR Q=134 THEN 3000  <215>
1080 IF Q=197 OR Q=229 OR Q=230 THEN 3000  <148>
1110 IF Q=101 OR Q=141 OR Q=177 THEN 3000  <202>
1140 IF Q= 49 OR Q=145 OR Q=17 THEN 3000   <167>
2000 NEXT                                  <232>
2005 :                                     <203>
2010 FOR I=1 TO 21                         <252>
2020 READ W,Q:POKE W,Q                     <003>
2030 NEXT                                  <006>
2100 :                                     <044>
2200 OPEN 1,8,1,"HSG2,P,W"                 <149>
2210 PRINT#1,CHR$(0);CHR$(144);            <237>
2220 FOR I=36864 TO 40032                  <005>
2230 PRINT#1,CHR$(PEEK(I));                <048>
2240 NEXT                                  <218>
2250 CLOSE 1                               <229>
2500 END                                   <216>
3000 :                                     <182>
3010 W=PEEK(I+1)                           <037>
3020 IF W=87 THEN POKE I+1,164:GOSUB 4000  <054>
3030 IF W=88 THEN POKE I+1,165:GOSUB 4000  <001>
3040 IF W=89 THEN POKE I+1,166:GOSUB 4000  <204>
3050 IF W=90 THEN POKE I+1,180:GOSUB 4000  <050>
3060 IF W=91 THEN POKE I+1,181:GOSUB 4000  <252>
3070 IF W=92 THEN POKE I+1,173:GOSUB 4000  <006>
3080 IF W=93 THEN POKE I+1,176:GOSUB 4000  <212>
3090 IF W=94 THEN POKE I+1,177:GOSUB 4000  <159>
3100 IF W=95 THEN POKE I+1,178:GOSUB 4000  <106>
3110 IF W=96 THEN POKE I+1,179:GOSUB 4000  <052>
3850 GOTO 2000                             <244>
3900 :                                     <066>
4000 C=C+1                                 <053>
4010 PRINT C:PRINT"{UP}";                  <006>
4020 RETURN                                <012>
5000 :                                     <150>
5010 DATA 37205,96,37264,95,37989,180,3831
     7,92,38323,93,38336,96,38379,87       <061>
5020 DATA 38388,89,38394,91,38420,95,38425
     ,94,39047,87,39060,88,39063,89        <083>
5030 DATA 39066,90,39069,91,39103,92,39106
     ,93,39114,96,39127,94,39130,95        <128>
```

Listing 3. Das Änderungsprogramm für die »High-Speed-Graphic«

Checksummer und MSE

Der Checksummer und der MSE sind Eingabehilfen für unsere Listings.

Der Checksummer zeigt für jede eingegebene Basic-Zeile eine Prüfsumme auf dem Bildschirm, die mit der in der 64'er abgedruckten Zahl (am Zeilenende) übereinstimmen muß. Diese Zahlen dürfen Sie beim Eintippen nicht mit eingeben. Unterstrichene Zeichen sind zusammen mit der<SHIFT>-Taste, überstrichene zusammen mit der <C=>-Taste einzugeben. Wenn im Listing geschweifte Klammern ({CLR}) auftauchen, dürfen Sie das, was innerhalb der Klammern steht, nicht eintippen, sondern müssen die entsprechenden Tasten drücken (zum Beispiel <CLR>).

Der MSE dient zur Eingabe von Maschinenspracheprogrammen. Auch er erzeugt zu jeder eingegebenen Zeile eine Prüfsumme. Diese »MSE-Listings« können Sie auch mit einem normalen Maschinensprache-Monitor eingeben. Dabei müssen Sie jedoch die letzte Spalte (Prüfsumme) weglassen.

Der Checksummer und der MSE wurden zuletzt in der Ausgabe 10/86 auf Seite 76 veröffentlicht. Beide sind auch auf jeder Programmservice-Diskette enthalten. Gegen Einsendung eines mit 1,80 Mark frankierten Rückumschlages (Format DIN A4) senden wir Ihnen die Listings mit Beschreibung auch gerne zu. (tr)