fehle zu verändern: Sie möchten aus einem AND #04 nachträglich ORA #04 machen? Aus JSR FCE2 soll INC 2321 werden? Alles kein Problem. Einzige Bedingung: Der alte Befehl muß im Speicher die selbe Länge haben wie der neue. Aus einem RTS, das im Speicher ein Byte belegt, können Sie durch einfache Modifizierung kein JSR FCE2 machen, weil dieser Befehl drei Bytes, also zwei zuviel benötigt. Durch Ändern der impliziten Befehle (aus INX wird DEX) könnte man in einer Schleife die Zählrichtung umkehren.

Als letztes Beispiel zum Thema Selbstmodifikation zeigen wir nun, wie man die Abarbeitung eines Unterprogramms verhindert. Wir werden dazu drei Lösungen entwickeln: Erst eine klassische, dann zwei, die auf Modifikation beruhen.

Das Unterprogramm beginnt bei dem Label UP. Im ersten Fall verwenden wir ein "Signal", ein "Flag", das dem Programm anzeigt, ob die Abarbeitung von UP gewünscht ist oder nicht. Der Wert dieses Signals sei 0, wenn das Unterprogramm gesperrt sein soll, bei jedem anderen Wert soll der Computer in das Unterprogramm springen. Das ist nicht weiter schwer: Wir schalten das Unterprogramm erst frei, indem ein anderer Wert als Null in das Flag geschrieben wird, dann folgt evtl. weiterer Programmtext, und dann prüfen wir durch Auslesen von FLAG, ob das Unterprogramm gestartet werden soll oder nicht.

```
FLAG    = 2     ; unbenutzte Speicherzelle
        LDA #1  ; ungleich 0
        STA FLAG        ; merken
        ...     ; evtl. weitere Befehle
        LDA FLAG        ; Unterprogramm starten?
        BEQ NEIN        ; Null, dann nicht
        JSR UP  ; sonst UP starten
NEIN    ...     ; weiter
```

Das Flag könnte auch zu Beginn des Unterprogramms abgefragt werden. Bei Feststellen einer Null müßte dieses dann sofort wieder verlassen werden (mit RTS). Nachteil: Für FLAG muß eine eigene Speicherzelle, in diesem Fall Speicherzelle 2, bereitgestellt werden.

Daher verwenden wir beim zweiten Versuch ein spezielles Unterprogramm. Seine Besonderheit: Der erste Befehl besteht nur aus einem einzigen Byte, beispielsweise ein NOP-Befehl.

```
UP      NOP     ; Einbyter
        ...     ; weiter im Unterprogramm
```

Die Ausführung dieses Unterprogramms, das in jedem Fall einfach mit JSR UP aufgerufen werden kann, gestatten folgende Befehle:

```
LDA #$EA        ; dezimal 234, Code für NOP
STA UP  ; als erstes Byte ins Unterprogramm
```

Wird das UP jetzt aufgerufen, findet es dort als erstes ein NOP vor, das einfach überlesen wird. Soll die Ausführung gesperrt werden, sorgen wir einfach dafür, daß der erste Befehl in der Subroutine den augenblicklichen Rücksprung ins Hauptprogramm auslöst: Ein RTS-Befehl mit dem Code $60.

```
LDA #$60        ; dezimal 96, Code für RTS
STA UP
```

Der Befehl JSR UP hat von jetzt an keine Wirkung mehr – bis das RTS wieder gegen ein NOP ausgetauscht wird. Eine Tabelle dieser "Opcodes" ($60 für RTS, $EA für NOP und so weiter) findet man in einschlägiger Literatur, etwa im Programmierhandbuch zum Commodore 64. Immerhin gibt es 256 verschiedene Codes.

Wer noch ein Byte für das NOP einsparen möchte, den NOP-Befehl im UP entfallen lassen. Dann muß aber auch der Opcode $EA beim Erlauben des Unterprogramms gegen den Wert des ersten Bytes des "freigegebenen" Unterprogramms ausgetauscht werden. Weil das unter Umständen sehr mühsam und fehlerträchtig ist, ist die Lösung mit Verwendung eines NOPs eindeutig vorzuziehen.

Jetzt folgt der dritte Streich: Wir ändern nichts am Unterprogramm, sondern schalten per Selbstmodifizierung einfach den Aufruf an und aus.

```
AUFRUF  JSR UP  ; Unterprogramm aufrufen
Aufruf erlauben:
LDA #$20        ; dezimal 32, Code für JSR
STA AUFRUF      ; -> JSR UP
Aufruf verbieten:
LDA #$2C        ; Code für BIT
STA AUFRUF      ; -> BIT UP
```

Wird der Aufruf verboten, wandelt der Computer den JSR-Befehl in einen BIT-Befehl um. Gelangt er nun an das Label AUFRUF, findet er dort den Befehl BIT UP vor. Er bewirkt alles, nur nicht den Aufruf des Unterprogramms. Anstelle des BIT könnte man zum Verbieten auch den Code $0c einsetzen. Dabei handelt es sich um einen "illegalen" Opcode für ein Dreibyte-NOP, das auf allen uns bekannten Versionen des C 64 arbeitet. Ob das auch beim C 128 funktioniert, konnte ich noch nicht überprüfen. Allerdings werden einige Monitore und Deassembler damit Probleme haben.

Die Stärken der relativen Adressierung

Oft muß in einem Programm eine bestimmte Stelle in jedem Fall angesprungen werden, also ohne daß eine Bedingung geprüft wird. Klar, das erledigt der JMP-Befehl für uns:

```
BEQ NULL
JMP STELLE
```

Falls das Zero-Flag gesetzt wird, soll es bei NULL weitergehen, andernfalls bei STELLE. Nicht selten liegt die STELLE nur wenige Bytes von dem aufrufenden JMP entfernt, und wir könnten die relative Adressierung verwenden:

```
BEQ NULL
BNE STELLE
```

hat die gleiche Wirkung, kostet aber nur vier statt fünf Bytes Speicherplatz. Der Grund: bei dem BNE STELLE ist das Zero-Flag in jedem Fall gelöscht – dafür hat ja der Abzweig-Befehl BEQ schon gesorgt. In jedem Fall wird nach STELLE verzweigt.

Man könnte den BEQ-Befehl in dieser Anwendung als "Pseudo-Verzweigungsbefehl" bezeichnen, da die Bedingung gar nicht überprüft werden müßte (weil sie sowieso immer erfüllt ist).

Der Branchbefehl übertrifft ein JMP deutlich an Effektivität, da ein Byte weniger verbraucht wird.

Im übrigen ist z.B. auch bei

```
BVS GESETZT
CLV
```

das CLV überflüssig, solange davor der BVS-Befehl abgearbeitet wird.

Daß solche Kapriolen unter Umständen sehr fehlerträchtig sind, hat Commodore bestens im Betriebssystem des C 64 gezeigt. Sicherlich kennen Sie schon die Eigenart, daß eine Zeile wie

```
20 REM {SHIFT L}
```

beim LISTen einen ?SYNTAX ERROR auslöst. Der Grund dafür liegt in der LIST-Routine, im Betriebssystem ab $A73D:

```
a73d 20 47 ab   jsr $ab47       ; ein Zeichen ausgeben
a740 d0 f5      bne $a737       ; weiter listen
************************        ; FOR-Befehl
a742 a9 80      lda #$80        ; Integer sperren
```

und so weiter. Der BNE-Befehl in a740 sollte eigentlich immer ausgeführt werden: Das Zero-Flag ist an dieser Stelle immer gelöscht, weil in einem Basicprogramm keine Nullbytes vorkommen (der Befehl jsr ab47 davor gibt das aktuelle Zeichen des Programms aus). Durch Zusammentreffen unglücklicher Umstände ergibt aber eine interne Konvertierung ausgerechnet bei dem Symbol {SHIFT L} ein Nullbyte beim Listen, welches in a73d ausgegeben wird. Da das Zero-Flag jetzt gesetzt ist, wird ausnahmsweise der BNE-Befehl bei a740 nicht ausgeführt, und die LIST-Routine rutscht versehentlich in die Routine, die den FOR-Befehl bearbeitet, und die zufällig direkt hinter dem LIST-Befehl im Speicher beginnt. Diese erwartet jetzt die Parameter eines FOR-Befehls, die wir ihr natürlich nicht bieten können. Dadurch kommt der ?SYNTAX ERROR zustande.

Hätte Commodore hier nicht am falschen Ende gespart und statt

```
a740 bne a737
```

ein Byte mehr spendiert und

```
a740 jmp a737
```

geschrieben, hätte der C 64 einen Systemfehler weniger.

Sie merken anhand dieses Beispiels also, wie wichtig eine sorgfältige Planung bei dieser Technik ist.

In der nächsten Assemblercorner erfahren Sie weitere Einzelheiten über geschicktes und zeitsparendes Programmieren in Assembler. (pk)

64'er-Kurz

Stundenlanges Blättern muß nicht sein: Mit den Kurzreferenzen bieten wir komprimiertes Wissen

The Final Cartridge III Basic & Monitor

von Torsten Hahn

Basicerweiterung starten

- C= Taste beim Einschalten des Rechners drücken
- im Desktop System/Basic anwählen
- im Freezer Reset/CBM 64 anwählen

Zusätzliche Basic-Befehle

aP	APPEND "name"	lädt Basicprg. name von Kassette und hängt es an im Speicher befindliches an (keine Durchnummerierung!)
aR	ARRAY	Ausgabe aller Felder des Basicprgs.
aU	AUTO x, y	nummeriert zu editierende Prg.Zeilen x - 1. Zeilennummer y - Inkrement
bA	BAR (OFF)	schaltet Pull-Down-Menü ein (aus)
dA	DAPPEND "name"	wie APPEND, jedoch mit Floppy
dE	DEL start-ende	löschen von Zeilen start bis ende
deS	DESKTOP	Wechsel zu Desktop (Datenverlust!)
dL	DLOAD "name"	lädt File name von Disk
dO	DOS "befehl"	sendet befehl an Floppy
dO	DOS "$	Directory anzeigen
dS	DSAVE "name"	speichert Prg. unter name auf Disk
dU	DUMP	gibt alle Variablen (außer Felder) aus
dV	DVERIFY "name"	vergleicht Bas.RAM mit name auf Disk
fI	FIND "text"	sucht text und gibt entsprechende Zeilen aus
hE	HELP	gibt fehlerhafte Basiczeile aus
kI	KILL	schaltet FC III ab
mE	MEM	Anzeige der Basic-Speicherverteilung
mO	MON	Wechsel zu Monitor
mR	MREAD adresse	kopiert 192 Byte ab adresse in Kassettenpuffer
mW	MWRITE adresse	kopiert Kassettenpuffer nach adresse
oL	OLD	restauriert Basicprg. nach Reset
oR	ORDER	ordnet Zeilennrn. nach (D)APPEND
pA	PACK	packt (komprimiert) Prg. im Speicher
pD	PDIR	druckt Directory aus
pL	PLIST	gibt Listing auf Drucker aus
rE	RENUM x, y	nummeriert Prg. neu durch x - 1. Zeilennummer y - Inkrement
reP	REPLACE"s","t"	ersetzt String s durch t
tR	TRACE (OFF)	(de)aktiviert Trace-Mode; nach RUN wird akt. Basic-Zeile gezeigt
uN	UNPACK	entpackt mit PACK gepacktes Prg.

Tastaturbelegung

	F1	LIST
	F2	Monitor aktivieren
	F3	RUN
	F4	OLD
	F5	DLOAD
	F6	DSAVE
	F7	DOS"$
	F8	DOS"
CTRL	RETURN	druckt den aktuellen Bildschirm aus
	CLR/HOME	Cursor in letzte Bildschirmzeile setzen

Pull-Down-Menü

Sofern das Pull-Down-Menü über den Befehl BAR eingeschaltet wurde, bietet das Basic des FC III diese Technik an. Um einen Befehl auszuwählen, drücken Sie die Feuertaste und halten sie nieder, bis Sie den entsprechenden Befehl mit Joystick oder Maus selektiert haben. Um Speicher zu sparen deaktivieren Sie das Menü mit BAR OFF.

Monitor starten

- in Basic F2 drücken oder MON eingeben
- System/Monitor im Basic Pull-Down-Menü wählen
- in Freezer Exit/Monitor anwählen

Monitorbefehlsübersicht

A xxxx mnemo (op)	startet den Assembler ab xxxx mit dem 1. Befehl mnemo und Operanden op
C xxxx yyyy zzzz	vergleicht Bereich von xxxx-yyyy mit dem ab zzzz; Gleichheit, keine Meldung
D xxxx yyyy	disassembliert von xxxx bis yyyy
EC xxxx yyyy	Bereich als Zeichen ausgeben; mit * (.) wird Bit gesetzt (gelöscht)
ES xxxx yyyy	Bereich als Sprite ausgeben; mit * (.) kann Bit gesetzt (gelöscht) werden
F xxxx yyyy zz	füllt Bereich xxxx-yyyy mit Byte zz
G xxxx	startet Maschinenprg. ab xxxx
H xxxx yyyy zz	sucht Bereich nach max. 8 Byte langer Folge (zz) ab; auch mit Strings ("string") möglich
I xxxx yyyy	stellt Bereich als editierbaren Text dar
L "name",xx,yyyy	lädt File von Gerät xx nach Ziel (nur Disk) yyyy
M xxxx yyyy	Bereich als Hex-Dump ausgeben (Hex-Zahlen editierbar)
O x	Speicherbankumschaltung; x (0-7) wird in Adresse $0001 Bits 0-2 geschrieben
OD	Wechsel zu Floppy-Mon; zurück mit O
P	Umleitung Bildschirmausgabe auf Drucker ein(aus)schalten
R	Registerinhalte ausgeben
S "name",xx, yyyy,zzzz	File name von yyyy-zzzz auf Gerät xx speichern
T xxxx yyyy zzzz	kopiert Bereich xxxx-yyyy nach zzzz
X	Rückkehr zu Basic
# dezimalzahl	wandelt Dez.- in Hexzahl um
$ hexzahl	wandelt Hex- in Dezimalzahl um
@ befehl	sendet befehl an Floppy
*R xx yy zz	Diskettensektor von Spur xx, Sektor yy lesen und in Page zz ablegen (Zahlen in Hex)
*W xx yy zz	schreibt Page zz in Spur xx, Sektor yy auf Disk (Zahlen in Hex)

Registerdarstellung

```
PC  IRQ  BK AC XR YR SP NV#BDIZC
AB25 EA31 07 8D FF 1C F9 *.**...*
```

PC Program Counter
IRQ Interrupt-Vektor
BK aktive Speicherbank
AC Accumulator
XR X-Register
YR Y-Register
SP Stack Pointer

N Negative Flag
V Overflow Flag
nicht benutzt
B Break Flag
D Dezimal Flag
I Interrupt Flag
Z Zerro Flag
C Carry Flag

Funktionstasten

F3	Crsr in erste Zeile
F5	Crsr in letzte Zeile
F7	Directory anzeigen

Tabelle für O x Befehl

x	aktive Speicherbereiche		
0	RAM	RAM	RAM
1	RAM	Z-Gen	RAM
2	RAM	Z-Gen	Kernel
3	Basic	Z-Gen	Kernel
4	RAM	RAM	RAM
5	RAM	I/O	RAM
6	RAM	I/O	Kernel
7	Basic	I/O	Kernel

referenz

auf kleinstem Raum. Damit lassen sich Fragen schneller beantworten als mit dem Handbuch.

The Final Cartridge III — Desktop

von Torsten Hahn

Desktop starten

- C64 mit FC III einschalten
- System/Desktop im Basic Pull-Down-Menü wählen
- in Basic Befehl Desktop eingeben
- in Freezer Exit/Desktop anwählen

Die Desktop-Menüleiste

INFO	SYSTEM	PROJECT	UTILITIES	CLOCK
DESKTOP	BASIC	NOTEPAD	PREFERENCES	TIME
VERSION	FINAL KILL	DLINK	BASIC PREFS	ALARM
	FREEZER	TLINK	CALCULATOR	SETTINGS
	REDRAW		DISK	
			TAPE	

- Bewegen Sie den Mauszeiger auf die Menüleiste; Drücken Sie Feuer
- Selektieren Sie den Befehl und lassen Sie die Feuertaste los

DESKTOP	- Desktopinfo	PREFERENCES	- Deskeinstellungen
VERSION	- Versionsinfo	BASIC PREFS	- Basiceinstellungen
BASIC	- Wechsel zu Basic	CALCULATOR	- Taschenrechner
FINAL KILL	- FC III abschalten	DISK	- Diskutilities
FREEZER	- Wechsel zu Free.	TAPE	- Kassettenutility
REDRAW	- Screen neuzeich.	TIME	- Zeitanzeige ein/aus
NOTEPAD	- Textverarbeitung	ALARM	- Alarm ein/aus
D/TLINK	- nicht aktiv	SETTINGS	- Zeiteinstellungen

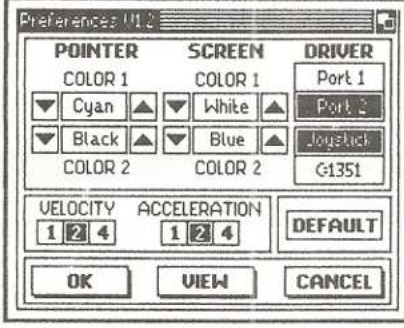
Windowbedienung

- Klicken Sie auf den Schließknopf um das Fenster zu schließen
- der Hintergrundknopf bringt verdeckte Fenster nach vorn
- bewegen Sie das Fenster durch Klicken auf den Balken

Joysticksimulation

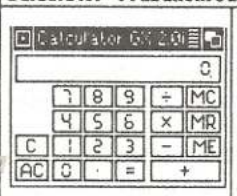
Oben F1 Links F5
Unten F3 Rechts F7
Feuer C=

Preferences (Einstellungen)



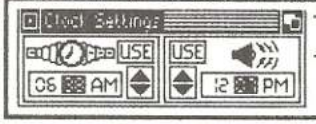
Pointer Pfeilfarben innen (Col1) und außen (Col2)
Screen Screenvorder- (Col1) und Hintergrund (Col2)
Driver Port und Eingabegerät
Velocity Pfeilgeschwindigkeit
Acceleration Pfeilbeschleunigung
Default Voreinst. wiederholen
View Einstellung ansehen

Calculator (Taschenrechner)



Icon	Funktion	Taste
+	Addition	+
-	Subtraktion	-
x	Multiplikation	*
:	Division	/
=	Ergebnis	=
ME	Speicher-Eingabe	E
MC	Speicher Löschen	C
MR	Speich.-Ausgabe	R
C	Eingabe Löschen	Clr
AC	Alles Löschen	Inst
.	Dezimalpunkt	.

Clock Setting (Uhr/Alarm-Einstellung)

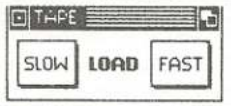


- USE übernimmt eingest. Werte
- 12/24 Stundenanzeige durch Klick auf AM/PM

Disk Operations (Diskettenoperationen)

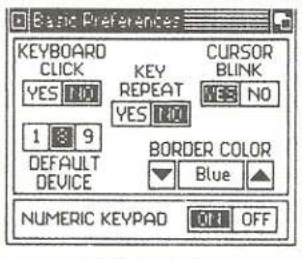
Dir Directoryfenster öffnen
Read Status Diskstatus lesen
Run + Do in Dirfenster markiertes File laden und starten
Validate + Do Diskette validieren
Scratch + Do markiertes File löschen
Initialize + Do Disk initialisieren
Fast Format + Do schnelles Formatieren
Emty + Do Diskette löschen
Rename + Do markiert. File umbenennen
Change Diskname + Do Diskettennamen ändern
Sort Umschalten zwischen Read (Dir neu lesen) und Line (Zeile in Dir einfügen)

Tape (Laden von Kassette)



- Laden mit einfacher (Slow) und 10-facher (Fast) Geschwindigkeit)
- Desktop wird verlassen (Sprung zu Basic)

Basic References (Einstellung FC III Basic)



Keybord Click Tastaturpieps
Key Repeat Tastenwiederholg.
Cursor Blink Cursorblinken
Default Device Geräteadresse
Border Color Rahmenfarbe
Num. Keypad 10-Tastatur C128 im 64-Mode

Notepad (Textverarbeitung)

Projekt		File		Line	
New	neuen Text beginnen (alten löschen)	Top of File	Cursor an Textanfang setzen	Space 0	Zeilenabstände wählen
Load	Text von Disk laden	Redraw	Bildschirm neuzeichnen	⋮	
Save	Text auf Disk speichern	Freezer	Freezer; kein Datenverlust	Space 2	
TLoad	Text von Kassette laden	Screen		Character	
TSave	Text auf Kassette speichern	Wordurap	Wortumbruch	Space 0	Zeichenabstände wählen
Print	Text drucken	Bold	Fettschrift	⋮	
Quit	Notepad verlassen	Proportional	Proprtionalschrift	Space 4	

Assembler-Bibliothek

Klein, aber fein sind unsere Assembler-Bibliothek-Routinen. Und wie immer haben wir auch heute wieder für jeden etwas: Vom Anfänger bis zum Profi gibt's etwas zu lernen oder abzutippen.

von Peter Klein

In unserer Assembler-Bibliothek geht´s diesmal um Stringhandling. "New Strout" sind zwei Unterprogramme von **Kirill Müller**, die Strings auf den Bildschirm schreiben (ähnlich der STROUT-Routine des Betriebssystems ab $AB1E). Mit der neuen Routine kann man die Zeichenketten wahlweise beliebig positionieren (NEW STROUT 1) oder zentrieren (NEW STROUT 2). Im Gegensatz zur Original-Routine nutzt die NEW STROUT-Routine mehr als 255 Zeichen. Notfalls können Sie mit einem einzigen String also den kompletten Bildschirm vollschreiben. Die Strings sollten im normalen ASCII-Format im Speicher stehen (also auch Steuerzeichen à la RVS ON usw.). Ein String wird in Teilstrings zerlegt (ein Teilstring = eine Zeile), die dann beliebig positioniert bzw. zentriert werden. Ein String für Routine 1 hätte das Format:

Zeile (0-24, 1 Byte), Spalte (0-39, 1 Byte), "Teilstring" (Textstring), $00 (1 Byte),..., "Teilstring", $00, $FF (Endbyte)

Für Routine 2:

Zeile (0-24, 1 Byte), Länge (1-40, Anzahl der Buchstaben im Teilstring ohne Steuerzeichen, 1 Byte), "Teilstring" (Textstring), $00 (1 Byte),..., "Teilstring", $00, $FF (Endbyte)

Die Routine wird per

```
LDA #< (Lowbyte Startadresse des Strings)
LDY #> (Highbyte Startadresse des Strings)
SEC/CLC (für Bildschirm löschen bzw. beibehalten)
JSR STROUT
```

gestartet und kehrt nach Ausgabe des Strings wieder mit RTS zurück.

ACHTUNG PROGRAMMIERER!

Sie haben viele kleine Unterroutinen geschrieben, die in Ihrer Schublade vergammeln? Her damit! Sie sind bares Geld wert. Egal ob es eine Routine ist, die Sprites initialisiert, den Joystick besonders effizient abfragt oder beispielsweise Werte in Basic-Variablen ablegt. Ihrer Fantasie sind keine Grenzen gesetzt. Zwei Bedingungen gibt es allerdings: Die Assemblerfiles müssen erstens in einem der folgenden Formaten vorliegen:

- HYPRA-ASS
- GIGA-ASS
- Profi-ASS
- Input-Macro-Assembler
- TURBO-ASS
- GIGA-ASS
- MagicFormel-ASS
- VIS-ASS

Außerdem müssen die Source-Codes möglichst ausführlich kommentiert sein. Eine Anleitung bzw. Programmbeschreibung auf Diskette kann auch nicht schaden. Die Routinen müssen sich per JSR aufrufen lassen und mit RTS wieder beendet werden.

Schicken Sie Ihre Source-Codes an:

Markt & Technik Verlag AG
64´er-Redaktion
Stichwort: Ass-Bibliothek
Postfach 1304
85531 Haar bei München

Listing 1: NEW STROUT 1 (Turbo-Assembler)

```
$4200  ;****************************************
$422B  ;*      NEW STROUT-ROUTINE 1 WITH       *
$4256  ;*   CURSOR-POSITIONING AND OPTIONAL     *
$4281  ;*             CLEARSCREEN              *
$42AC  ;*       (W) 1993 BY THE KILLER         *
$42D7  ;*        ALIAS KIRILL MUELLER          *
$4302  ;****************************************
$432D
$4331      £LA P=$FB:   BELIEBIGE ZEROPAGE-ADRESSN
$4357
$435B      £LA CLS=$E544:  >
$4381      £LA BSOUT=$FFD2;> KERNAL-ROUTINEN
$43A7      £LA PLOT=$FFF0: >
$43CD
$43D1  STROUT:
$43DB                      STA P
$43DF                      STY P+1
$43E5
$43E9                      BCC STROUT1
$43F3                      JSR CLS
$43F9  STROUT1:
$4404                      LDY #$00
$440B                      LDA (P),Y
$4413                      TAX
$4417                      CPX #$FF
$441E                      BEQ STROUT4
$4428                      INY
$442C                      LDA (P),Y
$4434                      TAY
$4438                      CLC
$443C                      JSR PLOT
$4443                      LDY #$02
$444A  STROUT2:
$4455                      LDA (P),Y
$445D                      BEQ STROUT3
$4467                      JSR BSOUT
$446F                      INY
$4473                      JMP STROUT2
$447D  STROUT3:
$4488                      INY
$448C                      CLC
$4490                      TYA
$4494                      ADC P
$4498                      STA P
$449C                      LDA P+1
$44A2                      ADC #$00
$44A9                      STA P+1
$44AF                      JMP STROUT1
$44B9  STROUT4:
$44C4                      RTS
$44C8
$44CC
```

© 64'er

Listing 2: NEW STROUT 2 (Turbo-Assembler)

```
$4200  ;****************************************
$422B  ;*      NEW STROUT-ROUTINE 2 WITH       *
$4256  ;*   CURSOR-POSITIONING AND OPTIONAL     *
$4281  ;*             CLEARSCREEN              *
$42AC  ;*       (W) 1993 BY THE KILLER         *
$42D7  ;*        ALIAS KIRILL MUELLER          *
$4302  ;****************************************
$432D
$4331      £LA P=$FB:   BELIEBIGE ZEROPAGE-ADRESSE
$4357
$435B      £LA CLS=$E544:  >
$4381      £LA BSOUT=$FFD2;> KERNAL-ROUTINEN
$43A7      £LA PLOT=$FFF0: >
$43CD
$43D1  STROUT:
$43DB                      STA P
$43DF                      STY P+1
$43E5
$43E9                      BCC STROUT1
$43F3                      JSR CLS
$43F9  STROUT1:
$4404                      LDY #$00
$440B                      LDA (P),Y
$4413                      TAX
$4417                      CPX #$FF
$441E                      BEQ STROUT4
$4428                      INY
$442C                      SEC
$4430                      LDA #40
$4436                      SBC (P),Y
$443E                      LSR
$4442                      TAY
$4446                      CLC
$444A                      JSR PLOT
$4451                      LDY #$02
$4458  STROUT2:
$4463                      LDA (P),Y
$446B                      BEQ STROUT3
$4475                      JSR BSOUT
$447D                      INY
$4481                      JMP STROUT2
$448B  STROUT3:
$4496                      INY
$449A                      CLC
$449E                      TYA
$44A2                      ADC P
$44A6                      STA P
$44AA                      LDA P+1
$44B0                      ADC #$00
$44B7                      STA P+1
$44BD                      JMP STROUT1
$44C7  STROUT4:
$44D2                      RTS
$44D6
$44DA
$44DE
```

© 64'er

Dieses Symbol zeigt an, welche Programme auf Diskette erhältlich sind