des Programmlaufs korrekt ausgeführt wird, verschluckt sie der Drucker im Listing vollständig. Das 4. Beispiel zeigt, daß von {P}10 zwischen »TEST« und »LAUF« nicht mehr zu bemerken ist.

```
TEST      LAUF


10 REM DRUCKSTARTPOSITION
15 REM IST IM LISTING
20 REM NICHT SICHTBAR
30 OPEN4,4
40 PRINT#4,"TESTLAUF"
50 PRINT#4:CLOSE4

READY.
```

Beispiel 4

Damit verbietet sich die Anwendung des synthetischen {P} bei Programmen, die zum Beispiel zur Veröffentlichung vom Drucker dokumentiert werden sollen.

Für die Zeichen {Z}Zeichenwiederholung und }[] (Punktadresse für Druckerstart) haben sich bislang noch keine sinnvollen Einsatzmöglichkeiten ergeben. Sie sollen daher jetzt nicht weiter diskutiert werden. Auch auf die Beschreibung von {Q}, {R}, {SHIFT Q} und {SHIFT R} möchte ich verzichten, da sie keine synthetischen Zeichen sind und sich damit «ganz normal« verhalten.

Ich persönlich benutze die synthetischen Steuerzeichen gern zur Gestaltung von Listings. Wie so etwas aussehen kann, habe ich Ihnen im abschließenden Beispiel 5 zusammengestellt.

Vielleicht versuchen Sie einmal, die versteckten und unsichtbaren Steuerzeichen herauszufinden.

(Jürgen Wagner)

```
100 REM DEMO ZUR OPTISCHEN STRUKTURIERUNG
110 REM      VON DRUCKER-LISTINGS MIT
120 REM"     SYNTHETISCHEN
130 REM"     STEUERZEICHEN


200 REM"*  HAUPTPROGRAMM  *

210 FORI=1TO64
220 PRINT"64'ER-MAGAZIN
230 GOSUB300:NEXT:"
240 END


300 REM"*  UNTERPROGRAMM  *

310 FORJ=1TO50:NEXT
320 RETURN:"


400 REM"ORIGINAL-LISTING!"

READY.
```

Beispiel 5

Autostart in Theorie und Praxis

Welcher Anwender hat sich nicht schon immer gewünscht, seine Programme »so einfach wie möglich« in den Computer zu bringen. Geräte der oberen Preis-/Leistungsklasse sind zu diesem Zweck mit einem Autostart-Mechanismus ausgerüstet. Dabei wird das Programm ohne weiteres Zutun nach dem Anschalten des Gerätes vom Massenspeicher (Floppy) in den Computer geladen. Hier soll jetzt die Realisierung eines solchen Autostarts auf einem C 64, beginnend mit der dazu nötigen Theorie, beschrieben werden.

Es gibt beim C 64 mehrere Möglichkeiten, einen Autostart herbeizuführen. Ich will Ihnen hier eine weniger bekannte Methode vorstellen:

Die Stack-Manipulation.

Der Stack belegt beim C 64 den Bereich $0100 bis $01ff (256 bis 511). Er wird unter anderem zum Ablegen von Rücksprungadressen benutzt. Nach Beendigung einer Routine sucht sich der Prozessor vom Stack zwei Bytes (nämlich die Rücksprungadresse in das Hauptprogramm) und springt die Adresse, die sich aus diesen beiden Bytes ergibt, an. Eigentlich ist daraus schon zu ersehen, was zu tun ist: Man müßte diese Rücksprungadresse so ändern, daß das Programm nicht an die eigentliche Rücksprungstelle springt, sondern auf eine eigene Routine. Jetzt jedoch tauchen schon die ersten Probleme auf: Wie soll der Stack geändert werden, ohne daß man dafür ein Extra-Programm braucht? Wie soll sich das Programm nach dem Ladevorgang automatisch starten?

Die Theorie

Auf beide Fragen gibt es eine Antwort: Da sich der Stack beim Ladevorgang ändern soll, ist es das einfachste, den (manipulierten) Stack einfach mit abzuspeichern! Damit hätten wir dann auch den von uns gewünschten Inhalt des Stacks geladen. Dann aber tauchen schon die nächsten Fragen auf: Woher wissen wir, aus welchem Stack-Bereich der Prozessor seine Rücksprungadresse holt? Die Antwort ist: Wir wissen es nicht! Unsere einzige Möglichkeit ist, den ganzen Stack mit der von uns gewünschten Rücksprungadresse zu belegen. Das Programm, auf das unsere Adresse zeigt (ein Maschinenprogramm), hängen wir direkt an den Stack an. Von dieser Startroutine wird nun das eigentliche Hauptprogramm, das wiederum hinter der Routine liegt, angesprungen. Die Reihenfolge der Programmteile und des benötigten Speichers ist in Bild 1 zusammengefaßt:

Startadresse des Programms auf Disk/Kas.: $0100	
$0100 — $01ff:	Stack, Lowbyte-1 und Highbyte der Startadresse unseres Startprogramms ($02)
$0200 — $0202:	Normaler Inhalt (unverändert)
$0203 — $0276:	Bereich für unser Startprogramm (technisch bedingt, immer gleich)
$0277 — $03ff:	Normaler Inhalt (unverändert)
$0400 — $07ff:	Bildschirmbereich, sollte Leerzeichen ($20) enthalten.
$0800:	Muß ein Null-Byte ($00) enthalten, damit der RUN-Befehl arbeiten kann.
$0801 — $xxxx:	Eigentliches Hauptprogramm, das vom Startprogramm angesprungen wird.

Bild 1. Programmadressen

Der Ladevorgang unseres Autostart-Programms muß mit »LOAD "xxx",8,1« erfolgen, damit das Programm nicht ab der Adresse $0801 (normaler Basic-Speicher), sondern ab $0100 geladen wird. Schauen wir uns nun noch einmal an, was im einzelnen beim Ladevorgang passiert:

1. Laden des Programms von Diskette oder Kassette mit »LOAD "xxx",8,1«.
2. Nach Beendigung des Ladevorgangs will sich der Prozessor vom Stack die Rücksprungadresse ins Basic holen, findet aber die (soeben geladene) Adresse auf unser eigenes Startprogramm und springt dies an.
3. Unser Startprogramm springt jetzt das eigentliche Hauptprogramm an, das irgendwo ab $0801 stehen sollte.

Nachdem also das Programm mit »LOAD "xxx",8,1« geladen wurde, startet es sich selbst sofort.

Zum Abschluß sei noch gesagt, daß einige Vorgänge zur besseren Verständlichkeit vereinfacht werden mußten. Bei eigenen Experimenten sei geraten, das abgedruckte Demo-Programm zu modifizieren. Außer einigen saftigen Abstürzen kann eigentlich nichts passieren.

Die Praxis

Das abgedruckte Demo-Programm läuft ohne Änderungen auf einem C 64 mit einer 1541-Floppy (Gerätenummer 8). Die Anpassung an andere Gerätenummern dürfte keine Schwierigkeit darstellen. Lediglich in den Zeilen 110, 112, 120 und 180 ist die 8 durch eine 9 zu ersetzen. Die Beschreibung der einzelnen Programmteile ist in Bild 2 zusammengefaßt.

Nun zum Programm selbst: Das Programm Autostart-Maker gibt dem Benutzer die Möglichkeit, ein beliebiges Programm mit einem Autostart zu versehen. Dabei muß folgende Bedingung erfüllt werden: Das Programm muß mindestens eine Basic-Zeile enthalten. Dies ist notwendig, da der vom Autostart-Maker generierte Autostart das Hauptprogramm mit dem RUN-Befehl startet. Zu diesem Zweck prüft der Autostart-Maker die Startadresse des gewünschten Programms und wirft eine Fehlermeldung aus, falls diese ungleich $0801 (Basic-Start) ist.

In den Zeilen 30000- stehen die DATAs für das Startprogramm, das das eigentliche Hauptprogramm anspringen soll. Die Bedeutung einzelner Zeilen läßt sich auch den REM-Statements des Programms entnehmen.

Ein mit diesem Autostart versehenes Programm ist auch in gewissem Sinne geschützt. Es läßt sich weder mit RUN/STOP noch mit RESTORE abbrechen. Wird extern ein RESET-Impuls erzeugt, so wird der ganze Speicher gelöscht, bevor in die normale RESET-Routine verzweigt wird. Diese Eigenschaften gehen auf den Aufbau unseres Startprogramms zurück.

Zeilennr.	Funktion
0 — 50	Copyright & Ausgabe der Kopfzeile
60 — 61	Eingabe des Programmnamens und Kürzen auf 16 Zeichen
70 — 95	Warteschleife auf einen Tastendruck
100	Zusammensetzen des neuen Namens
110 — 165	Generieren des Programmteils, der den Autostart enthält, auf Diskette ($0100-$0800)
170 — 300	Verbinden (Linken) des Autostart-Zusatzes und des eigentlichen Hauptprogramms
310	Ladeinstruktionen für das neu generierte, mit Autostart versehene Programm

Bild 2. Programmbeschreibung

Wenn aber alles in Ordnung ist, so arbeitet das Programm eine Weile mit der Diskette, bis es eine Ende-Meldung ausgibt. Sollte ein Diskettenfehler auftreten, so macht sich das Programm optisch und akustisch bemerkbar. Danach hat der Benutzer die Wahl: das Programm zu beenden oder einen neuen Start zu versuchen.

Im Programm sind folgende Unterroutinen enthalten:
10000 Fehlerkanal lesen und auswerten
20000 Gong ausgeben (wird von Fehler-Routine aufgerufen)

Geladen wird das neue Programm mit »LOAD "name",8,1«. Der Name entspricht dem des Ursprungprogramms mit dem Zusatz »/a«. Noch ein Tip: Wenn Sie im Besitz eines Basic-Compilers sind, so sollten Sie den Autostart-Maker compilieren. Und nun viel Spaß mit dem Autostart-Maker. Wenden Sie ihn doch am besten gleich mal bei Ihrem Programm an.

(Andreas Wurf)

Liste der verwendeten Variablen	
na$:	Name des zu modifizierenden Programms
t$, a$ — d$:	Variablen für diverse Zwecke
nw$:	Name des fertigen Autostart-Programms
a :	Variable für DATA-Elemente
si :	Startadresse des SID-Chips
i :	Schleifenvariable für diverse Zwecke

Variablenliste »Autostart«

```
0 REM" ******************************
1 REM" **    AUTOSTART - MAKER      **
2 REM" *      COPYRIGHT (C) BY       *
3 REM" *       ANDREAS DURF          *
4 REM" *      2000  HAMBURG 73       *
5 REM" *      HABENSTIEG  10 B       *
6 REM" *    TEL.: (040) 647 40 28    *
7 REM" **      VERSION 64            **
8 REM" ******************************
9 :
10 PRINT CHR$(9)+CHR$(14)+CHR$(8) "
UTOSTART - MAKER
20 PRINT"
    COPYRIGHT (C) 1983 ANDREAS DURF
"
30 PRINT"
          COMMODORE 64 - VERSION
"
40 PRINT"(/OTE:  DUR PROGRAM MUST HAVE A  
-"
50 PRINT" TART, SUCH AS '10  (XXXX)' !!)"
60 PRINT" NTER /AME OF PROG.: ";:OPEN 1,0:INP
UT#1,NA$:CLOSE 1:PRINT
61 NA$=LEFT$(NA$,16):REM "VUERZEN AUF 16 STELLE
N
70 PRINT" NTER PROG'S DISK AND PRESS A KEY !"
80 FOR I=0 TO 20:GET A$:IF A$<>"" THEN 100
85 NEXT:PRINT" NTER PROG'S DISK AND PRESS A K
EY !"
90 FOR I=0 TO 20:GET A$:IF A$<>"" THEN 100
95 NEXT:GOTO70
100 NW$=LEFT$(NA$,14)+"/A":REM" /AME DES NEUEN -
PROGRAMMS
110 OPEN 15,8,15:PRINT
112 OPEN 1,8,0,NA$:GOSUB 10000:CLOSE 1
120 OPEN 1,8,5,NW$+",P,W":PRINT" ENERATE AU
TOSTART":GOSUB 10000
130 PRINT#1,CHR$(0)+CHR$(1);:REM" PROGRAMMSTART
:= $0100
140 FOR I=256 TO 514:PRINT#1,CHR$(2);:NEXT:REM"
LOW-1 /  IGHBYTE DES STARTPGMS.
150 RESTORE:FOR I=515 TO 606:READ A:PRINT#1,CHR$
(A);:NEXT:REM" STARTPROGRAMM
160 FOR I=607 TO 1023:PRINT#1,CHR$(PEEK(I));:NEX
T:REM" /ORMALER ANHALT
164 FOR I=1024 TO 2047:PRINT#1,CHR$(32);:NEXT:RE
M" ILDSCHIRM MIT LEERZEICHEN
165 PRINT#1,CHR$(0);:GOSUB 10000:CLOSE 1
170 PRINT"- LINK TOGETHER BOTH FILES"
180 OPEN 1,8,0,NA$:GOSUB 10000:OPEN 2,8,5,NW$+",
P,A":GOSUB 10000
190 GET#1,A$:A$=A$+CHR$(0):GET#1,B$:B$=B$+CHR$(0
):REM" STARTADRESSE DES PGMS.
200 IF ASC(A$)=1 AND ASC(B$)=8 THEN 250:REM" E
ST AUF  PROGRAMM
210 PRINT"START-ADR. OF SOURCE-PGM IS NOT UNIQ
UE"
220 PRINT"TO $0801. CAN'T USE IT.":CLOSE 1:CL
OSE 2:CLOSE 15:GOSUB 20000:END
250 GET#1,A$:IF A$="" THEN A$=CHR$(0)
260 IF ST THEN PRINT#2,A$;:GOTO 300
270 PRINT#2,A$;:GOTO 250
300 GOSUB 10000:CLOSE 1:CLOSE 2:CLOSE 15
310 PRINT"OK. TYPE 'RUN"CHR$(34)+NW$+CHR$(3
4)",8,1' TO LOAD.":END
400 :
410 REM"  ROUTINEN UND -ABF ANGS
420 :
10000 INPUT#15,A$,B$,C$,D$:IF VAL(A$)=0 THEN RET
URN
10010 PRINT"ERROR# "A$": "B$" ON "C$",
"D$:GOSUB 20000
10020 PRINT"  QUIT OR RESTART ?"
10030 GET T$:IF T$<>"Q" AND T$<>"R" THEN 10030
10040 CLOSE 1:CLOSE 2:CLOSE 15:IF T$="R" THEN RU
N
10050 END
20000 SI=54272:POKE SI+5,9:POKE SI+6,0:POKE SI+2
4,15
20010 POKE SI,30:POKE SI+1,30:POKE SI+4,17:FOR I
=0 TO 300:NEXT:POKE SI+4,0
20020 POKE SI,20:POKE SI+1,20:POKE SI+4,17:FOR I
=0 TO 600:NEXT:POKE SI+4,0
20030 POKE SI+24,0:RETURN
30000 DATA 169,52,162,193,141,20,3,142,24,3,162,
4,189,16,253,157,4,128,202
30010 DATA 16,247,169,57,162,2,141,0,128,141,2,1
28,142,1,128,142,3,128
30020 DATA 165,174,166,175,133,45,134,46,32,99,1
66,32,142,166,76,174,167
30030 DATA 169,0,162,8,133,158,134,159,160,0,169
,0,145,158,200,208,251,230
30040 DATA 159,165,159,201,208,208,239,169,0,162
,9,157,0,128,202,16,250
30050 DATA 76,226,252
READY.
```

Mit diesem Listing können Sie jedes Programm, das mindestens eine Basic-Zeile enthält, mit einem Autostart versehen, direkt nach dem Laden ausgeführt wird.