# 22 Read Error — Theorie und Praxis

**Ein Programm läßt sich wirkungsvoll vor dem Kopieren schützen, indem man einen Sektor zerstört und in diesem Bereich wichtige Daten unterbringt.**

Bei Software, die auf Diskette gespeichert wird, dominiert eine Methode des Programmschutzes mit folgendem Prinzip:

Auf der Diskette mit diesem Programm ist ein Block mit Absicht zerstört worden. Wird jetzt dieses Programm geladen und gestartet, so wird vom Programm dieser fehlerhafte Block mit einem Direktzugriffsbefehl auf die Diskette abgefragt. Ist der Fehler vorhanden (was sich meist durch Blinken der Floppy-LED und Rattern des Schrittmotors äußert), so wird dies vom Programm erkannt. Es »merkt« dadurch sozusagen, daß »es« ein Original ist. Diese Methode funktioniert natürlich nur so lange, wie es nicht möglich ist, diesen fehlerhaften Block mitzukopieren. Bei älteren Kopierprogrammen ist dies nicht möglich. In letzter Zeit jedoch gibt es Kopierprogramme, die auf das Kopieren solcher Blöcke vorbereitet sind, indem sie die zerstörten Blöcke (in Form von Lesefehlern) erkennen und auf die Kopie »raufzaubern«. Bei dieser Prozedur wird aber in den meisten Fällen der Inhalt dieser Blöcke zerstört (auch zerstörte Blöcke können einen Inhalt haben).

Und hier zeigt sich ein Ansatzpunkt: Man müßte in diesen zerstörten Blöcken Daten unterbringen, die vom Originalprogramm gelesen werden können, von einer Kopie jedoch nicht. Das heißt, man macht sich den Effekt zunutze, daß diese zerstörten Blöcke durch das DOS nicht korrekt kopiert werden können. Wie aber kann man das DOS trotzdem dazu bringen, einen zerstörten Block kurzzeitig wieder lesbar zu machen?

Diese Frage läßt sich nur nach intensivem Studium des DOS der C 1541 beantworten. Nur wenn die Zusammenhänge und der Aufbau dieses Operations-Systems klar sind, kann auf eine solche Frage eine befriedigende Antwort gefunden werden. Der Autor fand folgende Lösung: In dem Bild ist der interne Aufbau eines Diskettenblocks dargestellt. Für uns ist die Konstante zum Beginn des Datenblocks ($07) wichtig. Das DOS braucht diese Konstante zur Erkennung des Anfangs eines Datenblocks. Sollte diese Konstante einen anderen Wert erhalten, so kann das DOS diesen Block nicht mehr lesen, das heißt er ist zerstört. Der Vergleichswert für diese Konstante liegt in der Zero-Page der Floppy (das heißt im RAM) und kann also geändert werden.

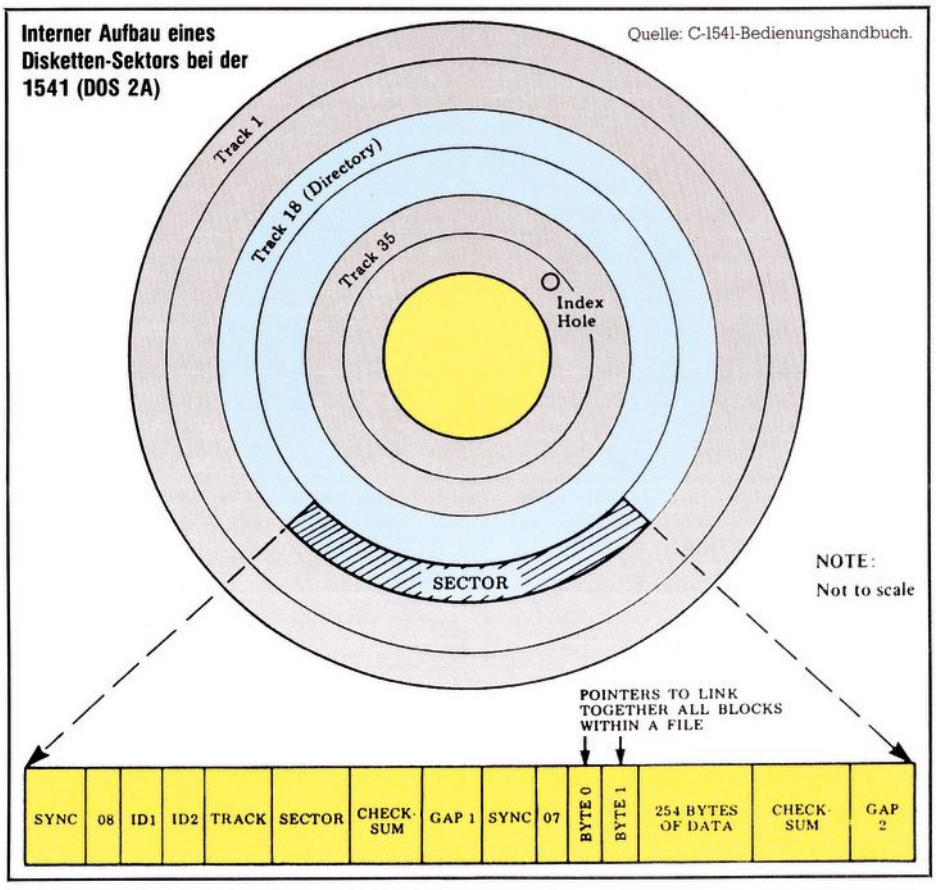
Man hat also durch Manipulation dieses Vergleichswertes die Möglichkeit, einen Block zu zerstören oder zu reparieren. Stellen Sie sich einmal folgendes vor:

1. Sie lesen einen Diskettenblock ein mit dem »U1: ...«-Befehl der Floppy.
2. Sie ändern den Vergleichswert für die Konstante $07 in der Zero-Page in irgend eine Zahl zwischen 0 und 255 außer 7.
3. Sie schreiben den Block auf die Diskette zurück, mit dem USR-Befehl »U2: ...« und
4. Sie setzen den Vergleichswert in der Zero-Page auf 7 zurück.

Sie haben jetzt folgendes gemacht: In dem Diskettenblock, den Sie eingelesen haben, stand bisher die Konstante $07. Dann haben Sie den Vergleichswert für die Konstante geändert, den Block wieder auf die Diskette geschrieben (wobei die geänderte Konstante auf die Diskette geschrieben wurde) und den Vergleichswert wieder auf den Normalwert gesetzt. Im Endeffekt haben Sie also auf der Diskette eine andere Konstante stehen, als in der Vergleichs-Speicherzelle der Floppy-Zero-Page. Versuchen Sie jetzt, diesen Block zu lesen, so werden Sie merken, daß das Laufwerk nur Spot- und Blinkeffekte hervorbringt. Mit anderen Worten: Sie haben diesen Block zerstört, und zwar mit dem Fehler Nummer 22, der das Fehlen eines Datenblockheaders anzeigt.

Dieser Diskettenfehler könnte jetzt wie oben beschrieben von dem zu schützenden Programm abgefragt werden.

Einigen Lesern wird wahrscheinlich jetzt schon klar sein, was passiert, wenn wir nun den Vergleichswert in der Floppy-Zero-Page auf



den Wert setzen, den wir vorhin auf die Diskette geschrieben haben. Wir gehen also jetzt wie folgt vor:
1. Vergleichswert in der Floppy-Zero-Page auf den Wert setzen, mit dem der Block vorhin auf die Diskette zurückgeschrieben wurde.
2. Den Block mit dem »U1: ...«-Befehl einlesen und eventuell auch mittels »GET#« in den Computer holen.
3. Den Vergleichswert wieder auf den Standard-Wert $07 setzen.

Wir haben den zerstörten Block kurzfristig lesbar gemacht, indem wir der Floppy einen anderen Vergleichswert untergejubelt haben, als der Standard-Wert. Somit »denkt« die Floppy beim Einlesen des Blocks, es sei alles in Ordnung. Nachdem der Block dann weiterverarbeitet wurde, haben wir den Vergleichswert für die Konstante wieder auf den normalen Wert gesetzt. Der fragliche Block gilt jetzt also wieder als zerstört.

Wichtig dabei ist, daß nach wie vor die 256 Bytes Inhalt des Blocks unverändert vorliegen, das heißt durch die »Zerstörung« des Blocks wurde an seinem Inhalt nichts geändert.

Das ist der Grundgedanke dieser Erweiterung bekannter Methoden zum Schützen von Software. Dieses Prinzip läßt sich jetzt vielfach variieren:

— Das zu schützende Programm könnte beispielsweise (das heißt ohne Verkettungsadressen auf den nächsten Block) abgespeichert und die davon belegten Blöcke zerstört werden. Von einem Vorprogramm aus werden dann die Blöcke einzeln restauriert, eingelesen und wieder zerstört (aus Sicherheitsgründen).

— Man könnte für jeden Block verschiedene Konstanten benutzen, die sich nach einem Algorithmus errechnen lassen (zum Beispiel Spur, EXOR, Sektor).

Eines muß hier ganz deutlich gesagt werden: Es hängt nur von der Programmierfähigkeit des »Schützers« ab, wie wirksam der Schutz ist. Die hier vorgestellte Methode liefert eben doch nur das Prinzip.

Noch ein Hinweis: Ein Programmschutz kann nur dann wirksam sein, wenn es keine Möglichkeit gibt, das Programm nach dem Ladevorgang einfach abzubrechen und abzuspeichern. Versuchen Sie daher, Ihre Lade- beziehungsweise Vorprogramme so zu schützen, daß sie möglichst keine Art der Einsicht erlauben. Bewährt hat sich in diesem Zusammenhang compiliertes Basic. Schreiben Sie also Ihre Ladeprogramme ruhig in Basic und compilieren Sie sie dann oder lassen sie compilieren.

Das Programm »Son of Destroyer« soll den Einstieg in diese Technik des Programmschutzes erleichtern und dessen Arbeitsweise verdeutlichen. Es ist gewissermaßen ein Programm zum Experimentieren und Sammeln von Erfahrungen. Es bietet folgendes: Man kann eine Diskette zerstören und wieder restaurieren, wobei die Bereiche der Diskette, die behandelt werden sollen, grafisch auf dem Bildschirm dargestellt werden können. Dazu bietet »Son of Destroyer« folgende Kommandos:
F1: Block als belegt kennzeichnen
F3: Block als frei kennzeichnen
F5: Spur als belegt kennzeichnen
F7: Spur als frei kennzeichnen

**Listing »Son of Destroyer«**

```
0 REM" ****************************
                <157>
1 REM" **{26SPACE}**                <090>
2 REM" *{6SPACE}SON OF DESTROYER
  {6SPACE}*                         <077>
3 REM" *{6SPACE}----------------
  {6SPACE}*                         <216>
4 REM" *{5SPACE}C64 - VERSION 1.19
  {5SPACE}*                         <210>
5 REM" *{2SPACE}ENTWURF UND PROGRAMM
   VON{2SPACE}*                     <116>
6 REM" *{8SPACE}ANDREAS WURF{8SPACE}*
                <077>
7 REM" *{28SPACE}*                  <012>
8 REM" **{26SPACE}**                <097>
9 REM" ****************************
                <166>
10 :                                <068>
11 :                                <069>
15 GOSUB 10000:REM            TITLE
                <216>
20 GOSUB 20000:REM            HELPPA
  GE                                <162>
30 GOSUB 30000:REM            SET VA
  RS                                <143>
35 GOSUB 40000:REM            PARAMS
                <049>
40 GOSUB 50000:REM            WORKFI
  ELD                               <026>
45 AW$="SLH{F1,F3,F5,F7,F2,F4,F8,UP,
  DOWN,LEFT,RIGHT}":DIM FU$(LEN(AW$))
                <123>
47 DW$="{HOME,24DOWN}"
  :FOR I=1 TO LEN(AW$):READ FU$(I)
  :NEXT                             <192>
50 POKE VI,PEEK(VI) OR 128          <145>
60 POKE 198,0:WAIT 198,1:GET T$     <239>
70 POKE VI,PEEK(VI)AND NOT 128      <076>
80 FOR I=1 TO LEN(AW$)
  :IF T$=MID$(AW$,I,1)THEN 87       <031>
85 NEXT:GOTO 50                     <255>
87 PRINT DW$"{WHITE}FUNCTION
  :{SPACE,RVSON}"FU$(I)"{RVOFF,
  12SPACE,LIG.BLUE,HOME}"           <006>
90 ON I GOSUB 4000,5000,200,500,800,
  1000,1300,1600,1900,2500,3000,3100,
  3200,3300                         <052>
95 GOTO 50                          <077>
200 GOSUB 20000:GOSUB 50000:RETURN
                <203>
500 POKE VI,42:RETURN               <133>
800 POKE VI,46:RETURN               <151>
1000 FOR I=0 TO 20:IF PEEK(VI+I*40)<>
  46 AND PEEK(VI+I*40)<>42 THEN 1010
                <124>
1005 POKE VI+I*40,42:NEXT           <119>
1010 FOR I=1 TO 20:IF PEEK(VI-I*40)<>
  46 AND PEEK(VI-I*40)<>42 THEN 1020
                <138>
1015 POKE VI-I*40,42:NEXT           <130>
1020 RETURN                         <141>
1300 FOR I=0 TO 20:IF PEEK(VI+I*40)<>
  46 AND PEEK(VI+I*40)<>42 THEN 1310
                <173>
1305 POKE VI+I*40,46:NEXT           <169>
1310 FOR I=1 TO 20:IF PEEK(VI-I*40)<>
  46 AND PEEK(VI-I*40)<>42 THEN 1320
                <187>
1315 POKE VI-I*40,46:NEXT           <180>
1320 RETURN                         <187>
1600 OPEN 15,GN,15:OPEN 2,GN,2,"#"
  :V=1868                           <116>
1605 FOR T=0 TO 34:FOR S=0 TO 20    <003>
1610 P=PEEK(V-(S*40)+T)
  :POKE V-(S*40)+T,PEEK(V-(S*40)+T)
  OR 128                            <217>
1620 IF P<>42 THEN 1665             <120>
1630 PRINT#15,"U1:"2;0;T+1;S        <039>
1640 PRINT#15,"M-W";CHR$(ME);CHR$(0);
  CHR$(1);CHR$(KO)                  <134>
1650 PRINT#15,"U2:"2;0;T+1;S        <060>
1660 PRINT#15,"M-W";CHR$(ME);CHR$(0);
  CHR$(1);CHR$(KN)                  <153>
1665 POKE V-(S*40)+T,PEEK(V-(S*40)+T)
   AND 127                          <012>
1666 GET T$:IF T$<>"*" THEN 1670    <037>
1668 PRINT DW$;"{WHITE,RVSON,SPACE}
  FUNCTION ABORTED{SPACE,RVOFF,
  LIG.BLUE,10SPACE,HOME}":CLOSE 2
  :CLOSE 15:RETURN                  <178>
1670 NEXT S,T:INPUT#15,F1$,F2$,F3$,F4$
  :PRINT"{HOME,6RIGHT,RVSON}"F1$"
   "F2$" "F3$" "F4$                 <079>
1680 CLOSE 2:CLOSE 15:RETURN        <112>
1900 OPEN 15,GN,15:OPEN 2,GN,2,"#"
  :V=1868                           <161>
1905 FOR T=0 TO 34:FOR S=0 TO 20    <048>
1910 P=PEEK(V-(S*40)+T)
  :POKE V-(S*40)+T,PEEK(V-(S*40)+T)
  OR 128                            <006>
1920 IF P<>42 THEN 1945             <166>
1925 PRINT#15,"M-W";CHR$(ME);CHR$(0);
  CHR$(1);CHR$(KO)                  <164>
1930 PRINT#15,"U1:"2;0;T+1;S        <084>
1935 PRINT#15,"M-W";CHR$(ME);CHR$(0);
  CHR$(1);CHR$(KN)                  <173>
1940 PRINT#15,"U2:"2;0;T+1;S        <095>
1945 POKE V-(S*40)+T,PEEK(V-(S*40)+T)
   AND 127                          <037>
1950 GET T$:IF T$<>"*" THEN 1970    <069>
1960 PRINT DW$;"{WHITE,RVSON,SPACE}
  FUNCTION ABORTED{SPACE,RVOFF,
  LIG.BLUE,10SPACE,HOME}":CLOSE 2
  :CLOSE 15:RETURN                  <215>
1970 NEXT S,T:INPUT#15,F1$,F2$,F3$,F4$
  :PRINT"{HOME,6RIGHT,RVSON}"F1$"
   "F2$" "F3$" "F4$                 <124>
1980 CLOSE 2:CLOSE 15:RETURN        <157>
2100 GOSUB 40000:GOTO 50            <229>
2500 PRINT DW$"{10SPACE,RVSON,WHITE,
  7SPACE}GOODBYE{7SPACE,HOME}":END
                <214>
3000 IF PEEK(VI-40)=42 OR PEEK(VI-40)
  =46 THEN VI=VI-40:RETURN          <034>
3005 RETURN                         <086>
3100 IF PEEK(VI+40)=42 OR PEEK(VI+40)
  =46 THEN VI=VI+40:RETURN          <132>
3105 RETURN                         <187>
3200 IF PEEK(VI-1)=42 OR PEEK(VI-1)=4
  6 THEN VI=VI-1:RETURN             <082>
3205 RETURN                         <031>
3300 IF PEEK(VI+1)=42 OR PEEK(VI+1)=4
  6 THEN VI=VI+1:RETURN             <179>
3305 RETURN                         <131>
4000 REM ** SAVE WORKPAGE **        <117>
4010 :                              <243>
4020 OPEN 15,GN,15:V=1868           <052>
4030 PRINT#15,"S:SOD.TEMP"
  :OPEN 2,8,2,"SOD.TEMP,U,W"
  :PRINT#2,GN:PRINT#2,KO            <044>
4040 FOR T=0 TO 34:FOR S=0 TO 20    <143>
4050 PRINT#2,CHR$(PEEK(V-(S*40)+T));
                <240>
4060 NEXT S,T:CLOSE 2               <076>
4070 INPUT#15,F1$,F2$,F3$,F4$:CLOSE 15
                <065>
4080 PRINT"{HOME,6RIGHT,RVSON}"F1$"
   "F2$" "F3$" "F4$                 <237>
4090 RETURN                         <151>
5000 REM ** LOAD WORKPAGE **        <082>
5010 :                              <223>
5020 OPEN 15,GN,15:V=1868
  :OPEN 2,GN,2,"SOD.TEMP,U,R"
  :INPUT#2,GN:INPUT#2,KO            <106>
5030 FOR T=0 TO 34:FOR S=0 TO 20    <113>
5040 GET#2,A$:A$=A$+CHR$(0)         <242>
5050 POKE V-(S*40)+T,ASC(A$)        <191>
5060 NEXT S,T:CLOSE 2               <056>
```

F2: Die als belegt gekennzeichneten Blöcke einer Diskette zerstören
F4: Die als belegt gekennzeichneten Blöcke einer Diskette restaurieren
»*«: Bricht die Funktionen F2 und F4 vorzeitig ab.
»h«: Zeigt die Befehlsliste an. Die Belegung einzelner Blöcke auf dem Bildschirm geht dabei verloren!
»s«: Speichert das Arbeitsfeld auf die Diskette, die sich momentan im aktuellen Laufwerk befindet.
»l«: Lädt das Arbeitsfeld.
F8: Beendet das Programm.

Das Programm fragt Sie nach der Gerätenummer der aktuellen Floppy und nach der Konstante, die Sie als neuen Wert auf dem Header der Blöcke stehen haben wollen, die Sie zerstören. Danach sehen Sie das Arbeitsfeld vor sich und links unten blinkt ein Cursor. Diesen Cursor können Sie jetzt wie gewohnt mit den Cursor-Steuertasten bewegen. Wenn Sie auf dem Block oder der Spur angelangt sind, die zerstört (oder restauriert) werden sollen, dann drücken Sie F1 zum Belegen dieses Blockes oder F5 zum Belegen der ganzen Spur. Das Rücksetzen geschieht analog dazu mit F3 beziehungsweise F7.

Danach können Sie mit F2 beziehungsweise F4 die als belegt gekennzeichneten Blöcke der im Laufwerk befindlichen Diskette zerstören beziehungsweise restaurieren.

Die beiden letztgenannten Funktionen können durch Druck auf die »*«-Taste vorzeitig beendet werden.

Mit »s« oder »l« kann das momentane Arbeitsfeld auf Diskette gespeichert beziehungsweise von Diskette geladen werden.

Mit »h« ist es Ihnen jederzeit möglich, sich eine Kommandoübersicht zu verschaffen, wobei allerdings das Arbeitsfeld gelöscht wird. Mit F8 schließlich beenden Sie das Programm.

Sie müssen beim Zerstören beziehungsweise Restaurieren der Diskette nur auf zwei Dinge achten:
1. Die Konstante, die Sie eingeben, muß beim Zerstören dieselbe sein wie beim Restaurieren und
2. Sie müssen beim Zerstören und Restaurieren immer dieselben Blöcke als belegt kennzeichnen.

Bleibt noch, Ihnen beim Erproben dieser Methode viel Spaß und Erfolg zu wünschen. Auch wenn's beim ersten Mal nicht klappt: Bleiben Sie am Ball. Es lohnt sich!

(Andreas Wurf/rg)

```
5070 INPUT#15,F1$,F2$,F3$,F4$:CLOSE 15
                                    <045>
5080 PRINT"{HOME,6RIGHT,RVSON}"F1$"
    "F2$" "F3$" "F4$                <217>
5090 RETURN                         <131>
9999 STOP                           <198>
10000 REM ** AUSGABE DES KOPFBLATTES
    **                              <129>
10001 REM ** UND VORBEREITEN DES
    **                              <119>
10002 REM ** DES BILDSCHIRMS
    **                              <128>
10003 :                             <116>
10010 POKE 646,PEEK(53280)
    :PRINT"{CLR}"CHR$(9);CHR$(14);
    CHR$(8);                        <050>
10020 PRINT"{RVSON,4SPACE}S O N
    {4SPACE}O F{4SPACE}D E S T R O
     Y E R{3SPACE}"                 <128>
10030 PRINT"{UP,RVSON,4SPACE}
    ================================
    {3SPACE}"                       <178>
10040 PRINT"{UP,RVSON,40SPACE}"     <223>
10050 PRINT"{UP,RVSON,4SPACE}***
    {27SPACE}***{3SPACE}"           <229>
10060 PRINT"{3DOWN,3SPACE}
    EIN PROGRAMM ZUM ZERSTOEREN UND"
                                    <056>
10070 PRINT"{3SPACE}WIEDERHERSTELLEN
     VON DISK-BLOCKS."              <199>
10075 PRINT"{DOWN,3SPACE}DER INHALT
     DIESER BLOCKS BLEIBT"          <183>
10077 PRINT"{3SPACE}VOLLSTAENDIG ERHA
    LTEN."                          <110>
10080 PRINT"{7DOWN,7SPACE}*** DRUECKE
    {SPACE,RVSON}RETURN{RVOFF,SPACE}***
                                    <188>
10090 GET T$:IF T$<>CHR$(13) THEN 100
    90                              <104>
10095 RETURN                        <036>
10096 :                             <209>
10097 :                             <210>
20000 REM ** AUSGABE DES HELPBLATTES
    **                              <145>
20003 :                             <171>
20020 PRINT"{CLR,RVSON,4SPACE}S O
    N{4SPACE}O F{4SPACE}D E S T R
    O Y E R{3SPACE}"                <074>
20030 PRINT"{UP,RVSON,4SPACE}
    ================================
    {3SPACE}"                       <233>
20040 PRINT"{UP,RVSON,14SPACE}HELP
    - PAGE{15SPACE}"                <137>
20050 PRINT"{2DOWN,2SPACE}ALLOCATE
    BLOCK{2SPACE}=>{2SPACE,RVSON,
    SPACE}F1{SPACE,RVOFF}           <067>
20060 PRINT"{2SPACE}FREE BLOCK{6SPACE}
    =>{2SPACE,RVSON,SPACE}F3{SPACE,
    RVOFF}                          <010>
20070 PRINT"{2SPACE}ALLOCATE TRACK
    {2SPACE}=>{2SPACE,RVSON,SPACE}F5
    {SPACE,RVOFF}                   <067>
20080 PRINT"{2SPACE}FREE TRACK{6SPACE}
    =>{2SPACE,RVSON,SPACE}F7{SPACE,
    RVOFF}                          <044>
20090 PRINT"{2SPACE}DESTROY DISK
    {4SPACE}=>{2SPACE,RVSON,SPACE}F2
    {SPACE,RVOFF}                   <239>
20100 PRINT"{2SPACE}REBUILD{2SPACE}
    DISK{3SPACE}=>{2SPACE,RVSON,SPACE}
    F4{SPACE,RVOFF}                 <216>
20110 PRINT"{DOWN,2SPACE}
    ABORT FUNCTION{2SPACE}=>{2SPACE,
    RVSON,SPACE}*{SPACE,RVOFF}      <207>
20115 PRINT"{2SPACE}HELP-PAGE{7SPACE}
    =>{2SPACE,RVSON,SPACE}H{SPACE,
    RVOFF}                          <118>
20117 PRINT"{2SPACE}SAVE PAGE{7SPACE}
    =>{2SPACE,RVSON,SPACE}S{SPACE,
    RVOFF}                          <092>
20118 PRINT"{2SPACE}LOAD PAGE{7SPACE}
    =>{2SPACE,RVSON,SPACE}L{SPACE,
    RVOFF}                          <071>
20120 PRINT"{2DOWN,2SPACE}
    QUIT PROGRAM{4SPACE}=>{2SPACE,
    RVSON,SPACE}F8{SPACE,RVOFF}     <059>
20140 PRINT"{2DOWN,BSPACE}***{2SPACE}
    PRESS{SPACE,RVSON}RETURN{RVOFF,
    2SPACE}***"                     <134>
20150 WAIT 198,1:GET T$
    :IF T$<>CHR$(13) THEN 20150     <164>
20160 RETURN                        <156>
20161 :                             <073>
20162 :                             <074>
30000 REM ** SETZEN DER PARAMETER **
                                    <049>
30001 :                             <224>
30010 GN=8:KO=139:VI=1868:ME=71:KN=7
    :RETURN                         <029>
30011 :                             <234>
30012 :                             <235>
40000 REM ** ANPASSEN DER PARAMETER **
                                    <232>
40001 :                             <023>
40010 PRINT"{CLR,RVSON,4SPACE}S O
    N{4SPACE}O F{4SPACE}D E S T R
    O Y E R{3SPACE}"                <174>
40020 PRINT"{UP,RVSON,4SPACE}
    ================================
    {3SPACE}"                       <077>
40030 PRINT"{UP,RVSON,13SPACE}SET
    PARAMETERS{13SPACE}"            <090>
40040 OPEN 1,0:PRINT"{2DOWN}"       <105>
40050 PRINT"{2SPACE}DEVICE #{5SPACE}
    :"GN"{UP}":PRINT"{17RIGHT}";
    :INPUT#1,GN:PRINT"{DOWN}"       <249>
40070 PRINT"{2SPACE}CONSTANT{5SPACE}
    :"KO"{UP}":PRINT"{17RIGHT}";
    :INPUT#1,KO:PRINT"{DOWN}"       <174>
40080 CLOSE 1:IF GN<8 OR GN>14 THEN 4
    0010                            <057>
40100 IF KO<0 OR KO>255 THEN 40010
                                    <123>
40110 RETURN                        <216>
50000 PRINT"{CLR}";                 <190>
50005 PRINT"{RVSON,SPACE}I/O
    : 00 OK 00 00{23SPACE,UP}"
    :PRINT"{UP}";                   <054>
50010 TR$="{4SPACE}.................
    ................."              <039>
50020 FOR I=0 TO 20:PRINT TR$:NEXT
                                    <050>
50030 PRINT"{HOME,DOWN,21RIGHT}
    XXXXXXXXXXXXXXXXX               <161>
50035 PRINT"{21RIGHT}XXXXXXXXXXXXXXX
    XX                              <130>
50040 PRINT"{28RIGHT}XXXXXXXXXXX    <106>
50050 PRINT"{34RIGHT}XXXXX          <018>
50060 PRINT"{HOME,DOWN}";
    :FOR I=20 TO 0 STEP-1:PRINT" "I
    :NEXT                           <024>
50070 PRINT"{4RIGHT}123456789012345678
    90123456789012345"              <208>
50080 PRINT"{4SPACE,RVSON}TRACKS
    {RVOFF,3SPACE}111111111122222222
    333333"                         <060>
50090 PRINT"{HOME,6DOWN}"
    :TR$="SECTORS"                  <096>
50095 FOR I=1 TO LEN(TR$)
    :PRINT"{RVSON}"MID$(TR$,I,1):NEXT
    :RETURN                         <005>
63040 :                             <112>
63050 REM ** DATA'S FUER FUNKTIONEN **
                                    <062>
63060 :                             <132>
63065 DATA "SAVE PAGE","LOAD PAGE"
                                    <015>
63070 DATA "HELP-PAGE",
    "ALLOCATE BLOCK","FREE BLOCK",
    "ALLOCATE TRACK"                <213>
63080 DATA "FREE TRACK",
    "DESTROY DISK","REBUILD DISK"   <035>
63090 DATA "QUIT PROGRAM","UP","DOWN",
    "LEFT","RIGHT"                  <208>
```

**Variablen bei »Son of Destroyer«:**

| | |
|---|---|
| aw$ | : Auswahl-Möglichkeiten am Zentralverteiler |
| fu$ | : Array mit Funktionsnamen |
| dw$ | : Hilfsstring zur Textausgabe |
| i | : Variable für Schleifen |
| vi | : Basispunkt des Cursors |
| t$ | : Variable für diverse Zwecke |
| gn | : Aktuelle Gerätenummer des benutzten Laufwerks |
| v | : Hilfsvariable für Bildschirmaufbau |
| s, t | : Schleifenvariable für Spur, Sektor |
| p | : Hilfsvariable |
| me | : Speicherstelle des Vergleichswertes in der Floppy-Zero-Page (71) |
| kn | : =7, Originalkonstante |
| ko | : vom Benutzer eingegebene Konstante |
| f1$—f4$ | : Variablen für Floppy-Fehlermeldungen |