6,- DM öS 50,- sfr 6,-

CPC Amstrad INTERNATIONAL

CPC • PCW JOYCE

12/1
Dezember 1991/
Januar 1992
6. Jahrgang

Erste Hilfe
Tips für Spielefreunde

Programmiersprachen
- Basic-Kurs: Weg mit dem GOTO
- Vier C-Compiler im Test
- Für Assemblerfreunde: RSX-Befehle »selbstgestrickt«
- Register im Griff: Inline-Hilfe für Turbo Pascal

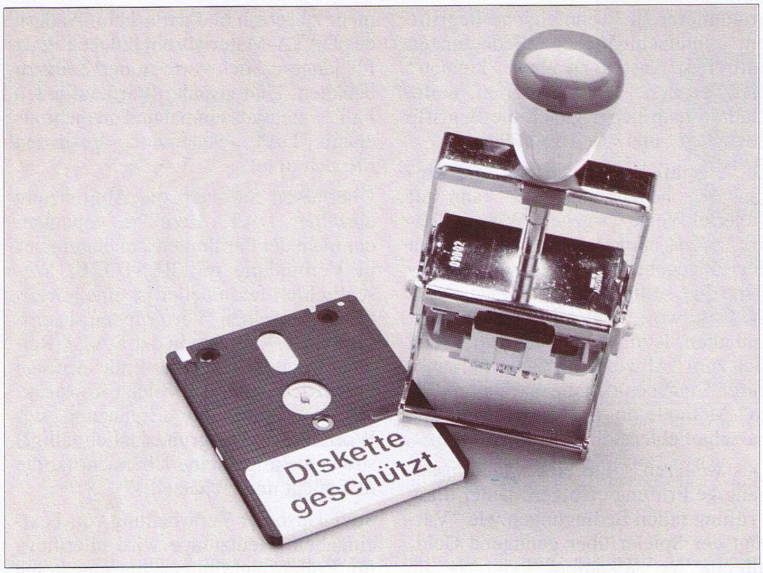
Große RSX-Bibliothek
- Alles, was Ihrem BASIC immer schon gefehlt hat
- 35 neue Befehle
- Pulldown-Menüs, RAM-Disk, Interruptverwaltung und, und, und ...

Kniffe und Kunstgriffe
- Binärdateien verschlüsseln
- Sortier-Algorithmen
- Widerstands-Kombinationen ermitteln
- Grafik-Konvertierung unter CP/M

PCW
- Grafiksystem im Eigenbau
- Neue LocoScript-Version
- Top-Software: Programmiertes Lexikon

Der Neue
PCW 512 plus

Brandaktuelle Spiele aus Frankreich



"Alkatrax"-Codierprogramm

Binärdateien leicht verschlüsselt

Das Thema Datenschutz ist schon seit den Kindertagen der Computer in aller Munde. Eine wichtige Maßnahme ist das Unlesbarmachen von Daten. Bisher konnte man nur auf dem CPC Datenfiles oder BASIC-Programme verschlüsseln. Wir stellen eine Möglichkeit vor, nun auch Maschinenprogramme vor fremdem Zugriff zu schützen.

Bisher gab es schon eine ganze Menge Codierprogramme, die jedoch alle bestimmte Nachteile hatten.

Entweder das Programm codiert mit einem Codewort, welches man leicht vergessen kann (außerdem muß man jedesmal wieder mit diesem Wort decodieren lassen), oder in einem Basic-Programm werden Zeilen durch Überlänge, falsche Zeilennummern und so weiter geschützt.

Eine weitere Möglichkeit eines Kopierschutzes ist, daß das Directory unsichtbar gemacht wird. Einfache Abhilfe: Diskmonitor, Directory Editor ... reinladen, Directory ändern, fertig. Alles wieder da. Aber warum eigene, wirkungslose und unbrauchbare Schutzarten ausdenken, wenn man nur bei den Profis schauen muß. Und genau das ist der mit Sicherheit wirkungsvollste Schutz, wie Sie gleich nach einer Erklärung erkennen können. Vorab noch eine kurze Information: Selbst der Entwickler dieses Schutzes bräuchte für ein normal geschütztes Programm vier bis zwölf Stunden (je nach Länge des Schutzes)!!! Nun folgen einige Informationen über diesen Kopierschutz und seine Arbeitsweise:

Jedem Computerbesitzer ist das Wort Kopierschutz ein bekannter Begriff (sofern er Originalsoftware benutzt und versucht hat, sie zu kopieren). Aber wie funktioniert ein Kopierschutz?

Die Grundlage ist eine mit einem Spezialformat formatierte Diskette. Dieses Format ist meist nur schwer zu kopieren. Das gesamte Programm ist dabei nicht als File abgespeichert, sondern direkt auf Diskette geschrieben. Um es einzulesen, wird eine Trackladeroutine aufgerufen, die das Programm von den Tracks in den Speicher liest. Wenn man diese Laderoutine, die oft sehr kompliziert ist, ungeschützt vorliegen hat, ist es meistens ein Leichtes (für Assembler-Kundige!), das Programm einlesen zu lassen und als File normal abzuspeichern. Auf diesem Prinzip beruht übrigens das Cracken von Originalsoftware:

Man versucht also an den Lader heranzukommen. Doch dies ist wirklich sehr schwer. Da die Programmierer natürlich nicht wollen, daß man leicht an diese Laderoutine kommt, verschlüsseln sie diese mit zirka 60 bis 300 Codierschleifen. Jetzt werden Sie fragen, was sind Codierschleifen und wie funktionieren sie ? So arbeiten diese Codierschleifen in Assembler :

Codierschleifen

Ein Register, meist HL, zeigt auf das erste Byte hinter dieser gerade "aktiven" Schleife. In BC oder DE steht die Anzahl der Bytes, die decodiert oder codiert werden sollen. Dann wird das Byte in HL eingelesen, verknüpft und wieder zurückgeschrieben. HL wird erhöht, DE beziehungsweise BC erniedrigt. Diese Prozedur wird so lange wiederholt, bis der Zähler (DE oder BC) gleich 0 ist.

Dann ist die nächste Schleife decodiert, und die darauffolgenden sind "etwas" decodiert. Bei dieser und den nächsten Schleifen geht's genauso weiter, wie eben gezeigt. Als Beispiel dient hier die Beispielroutine.

Nachdem die erste Schleife n mal (n = laenge) durchlaufen ist, steht die darauffolgende decodiert, also im Format wie die erste im Speicher und wird automatisch ausgeführt, weil ja nach dem letzten Befehl der ersten Schleife die zweite gleich beginnt. Und so geht das dann immer weiter, bis alle Schleifen beendet sind und das Programm richtig im Speicher steht.

Damit aber Programmcode decodiert werden kann, muß er vorher codiert werden. Man läßt den Vorgang also rückwärts ablaufen:

Combat School
Gryzor
Trantor - The Last Stormtrooper
Human Killing Machine
Thunder Blade
Sonic Boom
Die Arche des Captain Blood
Street Fighter
Last Ninja 2
Summer Games 1+2
Crackdown
Heath Wave (Compilation)
Arcade Muscles (Compilation)
10 MEGA Games (Compilation)

Diese Programme benutzen zum Beispiel den vorgestellten Kopierschutz.

```
begin   LD HL,start     ;Beginn der De - Codierung    BASIC : HL = start
        LD BC,laenge    ;laenge = ende - start + 1    BASIC : BC = laenge
loop    LD A,(HL)       ;Wert aus HL holen            BASIC : A = PEEK (HL)
        XOR wert        ;verknüpfen (wert:0-255)
        LD (HL),A       ;verknüpften Wert in HL schr. BASIC : POKE HL, A
        INC HL          ;HL um eins erhöhen           BASIC : HL = HL + 1
        DEC BC          ;BC um eins erniedrigen       BASIC : BC = BC - 1
        LD A,B          ;Test, ob BC = 0              BASIC : A = B
        OR C            ;Dazu wird B mit C geORt      BASIC : A = A OR C
        JR NZ,loop      ;Nicht 0 ? Dann nochmal       BASIC : IF A<>0 THEN loop
```

Beispiel für die (De)Codierung eines Binärfiles. Anschließend folgt das Programm.

Zuerst beginnt man mit der letzten Schleife bei der Codierung, dann bei der vorletzten und so weiter bis hin zur ersten. Nun steht alles geXORt im Speicher. Das Schwierige am Cracken, dem Hacken der Schleifen, ist aber, daß es so unwahrscheinlich viele Sorten davon gibt. Es werden insgesamt zirka fünfzig Typen verwendet. Aber wir wollen ja nicht cracken, weil wir keine Raubkopierer sind. Wozu aber dann die ganzen Erklärungen ?

- weil man so die Funktionsweise eines Kopierschutzes durchschauen kann,
- weil man so einsehen kann, daß dies der sicherste Kopierschutz überhaupt sein muß und
- weil dadurch die Funktionsweise meines Codierprogrammes anschaulich wird!

Funktionsweise des Programmes:

Nach dem Start werden Sie nach dem Namen des zu codierenden Programmes (nur Binärprogramme) und dessen Startadresse gefragt. Die Originalstartadresse muß hierbei unbedingt beibehalten werden! Dann kommt eine Abfrage nach der Anzahl der Schleifen. Geben Sie hier einen Wert zwischen 1 und X ein. X sollte nicht zu groß sein, da sonst der Speicherplatz nicht reichen würde. Aber vierzig Schleifen wären sicher schon genug.

Und dann geht's auch schon los! Das Programm wird codiert. Je nach Länge und Anzahl der Schleifen dauert dies einige Zeit; bei einer Länge von 4096 Bytes und einer Schleifenanzahl von vierzig zum Beispiel 45 Sekunden. Danach werden Sie aufgefordert, eine Taste zu drücken. Das fertige Produkt wird als CODE.BIN abgespeichert. Falls keine Abspeicherung gewünscht ist, drücken Sie einfach zweimal die Escape-Taste. Das folgende kurze Programm veranschaulicht eine Codierung:

```
FOR X=&4000 TO &5000:POKE X,&C9:NEXT
```

Danach das Codierprogramm starten, bei NAME bitte RETURN drücken, damit kein Programm geladen wird. Als Startadresse &4000, als Endadresse &5000. Danach merken Sie sich bitte die neue, vom Programm angegebene Startadresse. Danach sehen Sie sich den Speicher von &4000 bis &5000 wie folgt an:

```
FOR X=&4000 TO &5000:? HEX$(PEEK(X))"
";:NEXT
```

Jetzt müßten lauter verschiedene Werte ausgegeben werden. Das sind die codierten Bytes. Dann rufen Sie die neue Startadresse auf, die Sie sich gemerkt haben, mit: `CALL neue Startadresse` . Jetzt schauen wir nochmal den Speicher an:

```
FOR X=&4000 TO &5000:? HEX$(PEEK(X))"
";:NEXT
```

Der alte Wert &C9 steht jetzt wieder da. Wenn Sie ein Programm schützen, dann bitte nur ein Programm, welches selbständig lauffähig ist, also mit *CALL adresse* aufgerufen werden kann, und keine Grafik oder etwas in dieser Richtung. Viel Spaß beim Verschlüsseln.

Anmerkung: Die Erklärungen verlangen eine gewisse Vorkenntnis in der Programmiersprache Assembler. Aber auch Assembler-Unkundige können das Prinzip verstehen, nicht aber die Erklärung der Codierschleifen, was für die Benutzung des Programmes nicht erforderlich ist!

Klaus Meffert/rs

```
10 'CODIER.LDR                                  [777]
20 'Generiert PROGRAMM.BIN fuer das             [2821]
30 'Alkatraz Protection System                  [2837]
40 '(C) 1991 Klaus Meffert                      [435]
50 '    & CPC Internationl                      [1846]
60 '                                            [117]
70 FOR i=&AF00 TO &AFBF                         [1282]
80 READ a$:w=VAL("&H"+a$)                       [880]
90 s=s+w:POKE i,w:NEXT                          [1039]
100 IF s<> 21056 THEN PRINT"Fehler":END         [2578]
110 SAVE"codier.bin",b,&AF00,&C0:END            [2169]
120 DATA 3E,08,32,2C,AF,21,F0,3F                [903]
130 DATA 01,10,02,C5,7E,57,3A,2C                [856]
140 DATA AF,5F,06,09,3C,FE,80,C2                [1399]
150 DATA 1C,AF,D6,80,10,F6,32,2C                [1607]
160 DATA AF,7B,AA,77,23,C1,0B,78                [961]
170 DATA B1,20,E0,C9,18,01,00,00                [975]
180 DATA 16,00,21,00,00,01,00,00                [1422]
190 DATA 5E,7B,AA,77,53,0B,23,78                [1593]
200 DATA B1,C2,38,AF,C9,3E,09,32                [1319]
210 DATA F0,AF,21,30,40,01,10,00                [1439]
220 DATA 3A,F0,AF,C5,57,06,0D,3C                [1530]
230 DATA FE,80,C2,5F,AF,D6,80,10                [1722]
240 DATA F6,32,F0,AF,7E,AA,57,3A                [1499]
250 DATA F1,AF,AA,57,3A,F2,AF,AA                [1124]
260 DATA 77,00,00,23,C1,0B,78,B1                [1965]
270 DATA C2,50,AF,C9,3E,0A,32,2D                [350]
280 DATA AF,21,45,87,01,02,02,FD                [1586]
290 DATA 21,FF,3F,C5,3A,2D,AF,FD                [1724]
300 DATA AE,00,AC,AD,FD,77,00,0B                [1150]
310 DATA 2B,FD,23,3A,2D,AF,06,0F                [1815]
320 DATA 3C,FE,80,C2,A8,AF,D6,80                [1378]
330 DATA 10,F6,32,2D,AF,C1,0B,78                [751]
340 DATA B1,C2,8B,AF,C9,FF,FF,FF                [1569]
350 DATA 00,00,00,00,FF,FF,FF,FF                [1667]
```

```
10 '*********************************          [1383]
20 '*Special Alkatraz Protection System*        [2073]
30 '*Version 2.0 (c) 1991 Klaus Meffert*        [1926]
40 '*           & CPC International *           [966]
50 '*********************************          [1383]
60 'Dieses Programm erstellt einen Kopier-      [3011]
70 'schutz, wie er bei komerzieller Soft-       [3307]
80 'ware verwendet wird.                        [1653]
90 '                                            [117]
100 'Vorbereitung                               [643]
110 LOAD"codier.bin",&AF00                      [1692]
120 zyk=6:var=0:stufe=0                         [1965]
130 'Kopfzeile ausgeben                         [1933]
140 MODE 2                                      [513]
150 PRINT STRING$(80,"*");                      [1398]
160 PRINT"    ALKATRAZ PROTECTION SYSTEM        [5873]
(c) 1991 Klaus Meffert & CPC International
"
170 PRINT STRING$(80,"*")                       [1446]
180 'Eingaben                                   [1013]
190 INPUT"Programmname : ",name$                [2404]
200 IF name$="" THEN 230                        [393]
210 INPUT"Startadresse : ",adr                  [2868]
220 MEMORY adr-1:LOAD ""+name$,adr              [349]
230 INPUT"Wieviele Schleifen : ",loops          [3421]
240 'Variablen entsprechend den Eingaben d      [3262]
imensionieren
250 DIM a$(40),art(loops),lo(loops-1),hi(l      [4750]
oops-1),lo2(loops-1),hi2(loops-1)
260 DIM zy(loops-1),wert(loops-1),typ(loop      [5846]
s-1),lo3(loops-1),hi3(loops-1)
270 PRINT"Welcher Bereich soll codiert wer      [2969]
den ?
280 INPUT"Von : ",anfang                        [2117]
290 INPUT"Bis : ",ende                          [768]
```

```
300 FOR x=1 TO loops                                [1525]
310  art(x)=INT(RND*5)+1                            [1243]
320 NEXT                                            [350]
330 'Benoetigten Speicher berechnen                 [3036]
340 aut=1:space=anfang-1                            [2353]
350 laenge=ende-space+1                             [1235]
360 IF laenge<0 THEN laenge=laenge+65536            [2585]
370 LOCATE 1,6:FOR x=1 TO 6:PRINT SPACE$(8          [2640]
0);:NEXT
380 PRINT:FOR x=1 TO loops                          [2121]
390  IF art(x)=1 THEN space=space-18                [2735]
400  IF art(x)=2 THEN space=space-16                [1335]
410  IF art(x)=3 THEN space=space-25                [1905]
420  IF art(x)=4 THEN space=space-28                [1208]
430 NEXT                                            [350]
440 space=space-6:begin=space                       [1160]
450 IF space<&1C50+loops*50 THEN LOCATE 1,          [9519]
25:PRINT"          Achtung - Zu wenig Spe
icher - Bitte eine Taste druecken ! ":CALL
 &BB06:RUN 120
460 laenge=ende-space+1                             [1235]
470 IF laenge<0 THEN laenge=laenge+65536            [2585]
480 'Dateiinformationen ausgeben                    [1309]
490 LOCATE 55,5:PRINT"Start  :"HEX$(space,          [8818]
4):LOCATE 55,6:PRINT"Laenge :"HEX$(laenge+
57,4):LOCATE 55,7:PRINT"Ende   :"HEX$(ende
+57,4)
500 gesamt=laenge+space                             [1472]
510 GOSUB 1350:RESTORE 1450:FOR x=1 TO 7:R          [5241]
EAD a$:POKE space,VAL("&"+a$)
520 space=space+1:NEXT                              [872]
530 GOSUB 1350:LOCATE 1,12:PRINT"Loop Nr.:          [5682]
":FOR zaw=1 TO loops:LOCATE 10,12:PRINT za
w
540 ON art(zaw) GOSUB 760,890,1020,1170             [1566]
550 NEXT                                            [350]
560 GOSUB 1350:RESTORE 1470:spa=ende+1              [2885]
570 READ a$:FOR x=1 TO LEN(a$)                      [717]
580 POKE spa,ASC(MID$(a$,x,1)):spa=spa+1:N          [2091]
EXT
590 'Programm codieren                              [1190]
600 GOSUB 1350:PRINT:PRINT:PRINT"Codiere            [5872]
..... Bitte warten !!!!"
610 FOR x=var-1 TO 0 STEP -1:ON typ(x) GOS          [3063]
UB 680,700,720,740
620 NEXT                                            [350]
630 LOCATE 1,24:PRINT"Neue Startadresse :           [4251]
"HEX$(begin)
640 PRINT"          Bitte die Sicherungsdis         [7265]
kette einlegen und eine Taste druecken":CA
LL &BB06
650 SAVE"code",b,begin,laenge+57,start              [2678]
660 END                                             [110]
670 'Codieren mit Typ 1                             [657]
680 POKE &AF33,lo(x):POKE &AF34,hi(x):POKE          [5878]
 &AF36,lo2(x):POKE &AF37,hi2(x):POKE &AF31
,wert(x):CALL &AF30:RETURN
690 'Codieren mit Typ 2                             [664]
700 POKE &AF06,lo(x):POKE &AF07,hi(x):POKE          [8181]
 &AF09,lo2(x):POKE &AF0A,hi2(x):POKE &AF01
,zy(x):CALL &AF00:RETURN
710 'Codieren mit Typ 3                             [663]
720 POKE &AF4B,lo(x):POKE &AF4C,hi(x):POKE          [9090]
 &AF4E,lo2(x):POKE &AF4F,hi2(x):POKE &AF46
,zy(x):POKE &AFF1,lo3(x):POKE &AFF2,hi3(x)
:CALL &AF45:RETURN
730 'Codieren mit Typ 4                             [670]
740 POKE &AF7D,zy(x):POKE &AF89,lo(x):POKE          [8276]
 &AF8A,hi(x):POKE &AF85,lo2(x):POKE &AF86,
hi2(x):POKE &AF82,lo3(x):POKE &AF83,hi3(x)
:CALL &AF7C:RETURN
750 END                                             [110]
760 'Typ 1                                          [82]
770 lo=0:hi=0:wert=INT(RND*65)+1:wert(var)          [4791]
=wert:merk=space:RESTORE 1370
780 FOR x=1 TO 18:READ a$(x):POKE space,VA          [4857]
L("&"+a$(x)):space=space+1:NEXT
790   POKE merk+7,wert                              [972]
800 space=space-1:hi=INT(space/256):lo=spa          [4530]
ce-hi*256:IF hi<0 THEN hi=256+hi
810   POKE merk+1,lo:POKE merk+2,hi                 [1448]
820 lo(var)=lo:hi(var)=hi                           [1808]
830 zahl=ende-space+1:hi=INT(zahl/256):lo=          [5869]
zahl-hi*256:IF hi<0 THEN hi=256+hi
840   POKE merk+4,lo:POKE merk+5,hi                 [933]
850 lo2(var)=lo:hi2(var)=hi:zyk=zyk+(ende-          [6189]
space+1)*9:zyk=zyk+3:space=space+1
860   GOSUB 1340                                    [974]
870 typ(var)=1:var=var+1                            [491]
880 RETURN                                          [555]
890 'Typ 2                                          [89]
900 zyk=zyk+2:GOSUB 1340:zy(var)=zyk                [1896]
910 rreg=1:lo=0:hi=0:merk=space:RESTORE 13          [4338]
90:FOR x=1 TO 16:READ a$(x)
920   POKE space,VAL("&"+a$(x)):space=spac          [3056]
e+1:NEXT
930 space=space-1:hi=INT(space/256):lo=spa          [4530]
ce-hi*256:IF hi<0 THEN hi=256+hi
940   POKE merk+1,lo:POKE merk+2,hi                 [1448]
950 lo(var)=lo:hi(var)=hi                           [1808]
960 zahl=ende-space+1:hi=INT(zahl/256):lo=          [5869]
zahl-hi*256:IF hi<0 THEN hi=256+hi
970   POKE merk+4,lo:POKE merk+5,hi                 [933]
980 lo2(var)=lo:hi2(var)=hi:zyk=zyk+(ende-          [4565]
space+1)*9:space=space+1
990   GOSUB 1340                                    [974]
1000 typ(var)=2:var=var+1                           [1314]
1010 RETURN                                         [555]
1020 'Typ 3                                         [88]
1030 zyk=zyk+5:GOSUB 1340:zy(var)=zyk:merk          [4195]
=space:RESTORE 1410:FOR x=1 TO 25
1040 READ a$(x):POKE space,VAL("&"+a$(x)):          [2852]
space=space+1:NEXT
1050 hi=INT((space-16)/256):lo=(space-16)-          [2681]
hi*256:IF hi<0 THEN hi=256+hi
1060   POKE merk+7,lo:POKE merk+8,hi                [1176]
1070 lo3(var)=lo:hi3(var)=hi                        [1276]
1080 space=space-1:hi=INT(space/256):lo=sp          [4530]
ace-hi*256:IF hi<0 THEN hi=256+hi
1090   POKE merk+1,lo:POKE merk+2,hi                [1448]
1100 lo(var)=lo:hi(var)=hi                          [1808]
1110 zahl=ende-space+1:hi=INT(zahl/256):lo          [5869]
=zahl-hi*256:IF hi<0 THEN hi=256+hi
1120   POKE merk+4,lo:POKE merk+5,hi                [933]
1130 lo2(var)=lo:hi2(var)=hi:zyk=zyk+(ende          [4306]
-space+1)*13:space=space+1
1140   GOSUB 1340                                   [974]
1150 typ(var)=3:var=var+1                           [556]
1160 RETURN                                         [555]
1170 'Typ 4                                         [95]
1180 zyk=zyk+4:GOSUB 1340:zy(var)=zyk               [2208]
1190 lo=0:hi=0:merk=space:RESTORE 1440:FOR          [2183]
 x=1 TO 28:READ a$
1200   POKE space,VAL("&"+a$):space=space+          [1634]
1:NEXT
1210 space=space-1:hi=INT(space/256):lo=sp          [4530]
ace-hi*256:IF hi<0 THEN hi=256+hi
1220   POKE merk+5,lo:POKE merk+6,hi                [2271]
1230 lo(var)=lo:hi(var)=hi                          [1808]
1240 zahl=ende-space+1:hi=INT(zahl/256):lo          [5869]
=zahl-hi*256:IF hi<0 THEN hi=256+hi
1250   POKE merk+1,lo:POKE merk+2,hi                [1448]
1260 lo2(var)=lo:hi2(var)=hi                        [1812]
1270 RANDOMIZE timer:lo3(var)=INT(RND*256)          [4785]
:POKE merk+8,lo3(var)
1280 hi3(var)=INT(RND*256):POKE merk+9,hi3          [4182]
(var)
1290 zyk=zyk+(ende-space+1)*15:space=space          [2230]
+1
1300   GOSUB 1340                                   [974]
1310 typ(var)=4:var=var+1                           [1161]
1320 RETURN                                         [555]
1330 '                                              [117]
1340 za=zyk:za=INT(za/128):zyk=zyk-za*128:          [3626]
RETURN
1350 stufe=stufe+1:LOCATE 55,9:PRINT"Stufe          [3679]
  : "stufe:RETURN
1360 'DATA-FIELD                                    [411]
1370 '1.Typ                                         [95]
1380 DATA 21,00,00,01,00,00,16,00,7e,aa,57          [3233]
,77,23,0b,78,b1,20,f6
1390 '2.Typ                                         [719]
1400 DATA 21,00,00,11,00,00,ed,5f,ae,77,1b          [1685]
,7a,b3,23,20,f6
1410 '3.Typ                                         [190]
1420 DATA 21,00,00,01,00,00,cd,00,00,dd,e1          [4573]
,ed,5f,dd,ac,ae,dd,ad,77,0b,79,23,b0,20,f2
1430 '4.Typ                                         [303]
1440 DATA 01,00,00,fd,21,00,00,21,00,00,ed          [4758]
,5f,fd,ae,00,ac,ad,fd,77,00,0b,2b,fd,23,78
,b1,20,ee
1450 'INTRODUCTION                                  [1108]
1460 DATA f3,ed,56,3e,04,ed,4f                      [1371]
1470 DATA "Special Alkatraz Protection Sys          [8821]
tem (c)1991 Klaus Meffert & CPC Internatio
nal"
1480 DATA @                                         [215]
```