Auf das »!« kommt es an

Das folgende Programm wurde aus der Not geboren. Es erleichtert das Laden von Diskette und macht das umständliche Laden und Listen des Directory überflüssig. Nebenher lernen Sie eine Reihe nützlicher Maschinenroutinen kennen.

Geht es Ihnen auch so: Ich weiß nie genau, ob das Programm, das ich laden willl, nun »Disk Copy V 1.0« oder »Disk Copy V1.0« heißt. Versuche ich es mit »Disk*«, lade ich mit Sicherheit »Disk Monitor«.

Es hilft also nichts: Ich lade das Directory, liste es und — ärgere mich, weil das Programm, das ich laden wollte, ganz oben steht und beim Scrollen verschwindet. Also nochmal »LIST«, dann mit dem Cursor in die richtige Reihe fahren, »LOAD« eingeben, Cursor hinter den Programmnamen, »,8« eintippen. Der Computer antwortet mit einem verächtlichen »SYNTAX ERROR«, weil ich den Doppelpunkt mal wieder vergessen habe...

Vielleicht stelle ich mich besonders dumm an, aber es sollte auch anders gehen. Für Tätigkeiten, die meine geistigen Fähigkeiten übersteigen, gibt es doch Computer! Aus diesem Gedanken entstand das Programm.

Es wird mit »LOAD"!",8,1« geladen, startet automatisch, liest das Directory ein und schreibt hinter jedes Programm eine Nummer. Wenn der Bildschirm voll ist, wartet es auf meine Eingabe. Tippe ich »+«, bekomme ich die nächsten Programme angezeigt. Tippe ich aber die Programmnummer und dahinter ein »L«, wird das entsprechende Programm automatisch geladen. Handelt es sich um ein Maschinenprogramm, das absolut geladen werden muß (mit »,8,1«), tippe ich hinter die Nummer ein »A«.

So funktioniert es

Da ich nicht auf einen Autostart verzichten wollte, mußte das Programm vollständig in Maschinensprache geschrieben werden. Außerdem mußte es sehr kurz sein, denn ich hatte nur den Bereich von Speicherstelle 256 bis 600 ($0100 — $0258) zur Verfügung. Zu allem Unglück benötigen die Betriebssystemroutinen, die das Programm verwendet, auch noch Platz in diesem Bereich (Prozessorstack). Deshalb mußte ich auf alles verzichten, was nicht unbedingt nötig war. Es gibt also kein Menü für den Benutzer, ebenso keine Möglichkeit, das Programm zu unterbrechen, wenn man nichts laden will. Man kann nur »0L« eingeben, dann springt es mit einer Fehlermeldung ins Basic zurück. Trotzdem ist der verfügbare Speicher bis auf das letzte Byte belegt.

Die folgende Programmbeschreibung ist zwangsläufig sehr theoretisch. Das Programm läuft ja aber auch, wenn man sie nicht versteht. Machen Sie sich aber ruhig einmal die Mühe. Vielleicht können Sie doch den einen oder anderen nützlichen Tip bekommen. Wem es zu kompliziert wird, der kann bei den »Hinweisen zum Abtippen« weiterlesen.

Nehmen Sie also das Assembler-Listing zur Hand und folgen Sie mir in die Tiefen der Maschinensprache... Zur besseren Übersicht habe ich die einzelnen Teile beschrieben und immer die entsprechenden Zeilennummern dahintergesetzt.

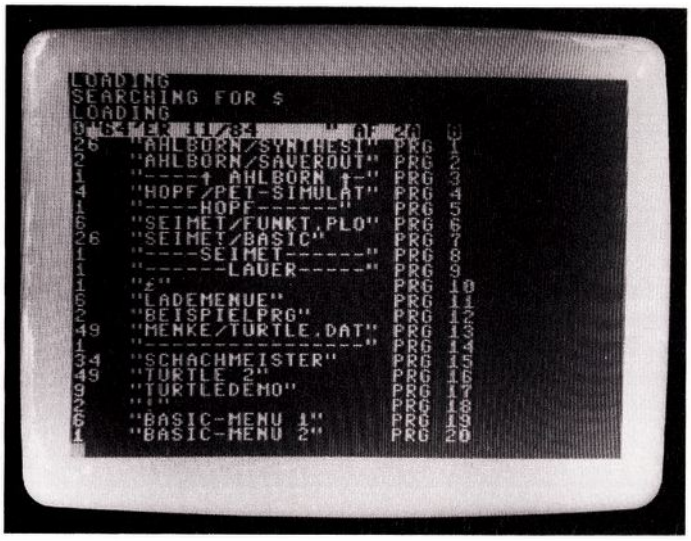
Im Programmkopf bis Zeile 320 werden die Betriebssystemroutinen und die Variablen definiert. Sie sind nochmals im Kasten erläutert. Als Startadresse wird $010A festgelegt, der Maschinencode kann aber nicht direkt dorthin assembliert

```
10 FORI=49162TO49495:READA:POKEI,A:S=S+A
20 NEXT:IFS<>35775THENPRINT"FEHLER IN DA
TAS":STOP
30 PRINT"■BITTE LEGEN SIE EINE NEUE DISK
ETTE EIN. ■DISKETTE WIRD FORMATIERT !"
40 GETA$:IFA$=""THEN40
50 OPEN1,8,15,"N0:MENU,MN":CLOSE1
60 PRINT"■■■■POKE43,0:POKE44,192:POKE45,
90:POKE46,193"
70 PRINT"■■SAVE"CHR$(34)"!"CHR$(34)",8"
80 PRINT"■";
90 FORI=631TO641:POKEI,13:NEXT:POKE198,1
0
100 DATA165,43,133,57,165,44,133,58,169,
13,32,210,255,160,0,177,57,133,59,200
110 DATA177,57,133,60,200,177,57,170,200
,177,57,32,205,189,160,3,200,177,57
120 DATA240,5,32,210,255,208,246,165,59,
133,57,165,60,133,58,160,1,177,57,240
130 DATA23,169,27,133,211,169,0,166,251,
32,205,189,230,251,230,253,165,253
140 DATA201,21,144,183,169,128,133,254,1
69,13,32,210,255,162,0,32,207,255,157
150 DATA64,3,232,201,13,208,245,202,202,
189,64,3,201,43,208,12,169,0,133,253
160 DATA36,254,48,144,133,251,16,132,201
,76,208,6,169,0,133,252,240,8,201,65
170 DATA208,208,169,128,133,252,202,189,
64,3,41,15,133,251,202,48,16,189,64
180 DATA3,41,15,170,24,165,251,105,10,13
3,251,202,208,247,166,251,24,165,43
190 DATA105,35,133,57,165,44,133,58,202,
240,111,24,165,57,105,32,133,57,144
200 DATA244,230,58,176,240,36,2,2,2,2,2,
2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
210 DATA2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,
2,2,2,2,2,2,2,2,2,2,2,169,1,162,208
220 DATA160,1,32,189,255,162,8,160,0,32,
186,255,152,133,253,133,251,166,43
230 DATA164,44,32,213,255,32,51,165,76,1
0,1,160,0,177,57,201,34,240,3,200,208
240 DATA247,152,96,32,37,2,56,101,57,133
,57,32,37,2,166,57,164,58,32,189,255
250 DATA169,1,162,8,160,0,132,10,36,252,
16,2,160,1,32,186,255,76,111,225
READY.
```

Listing »Lader«

werden, sonst würde ja der Autostart anlaufen. Deshalb wird er zunächst in den Bereich $C00A — $C158 geschrieben.

Das Programm selbst beginnt mit Zeile 1510. Die Routine »DIRIN« lädt das Directory in den Speicher. Dazu benutzen wir die Betriebssystemroutine »LOAD«. Diese braucht zur Vorbereitung die Routinen SETLFS und SETNAM. SETLFS bestimmt die logische Filenmmer (01), die Geräteadresse (08) und die Sekundäradresse (0 für Lesen). SETNAM übergibt den Filenamen, in unserem Falle »$«. LOAD muß schließlich noch wissen, wohin das Directory geladen werden soll. Diese Adresse (Anfang des Basic-Speichers, steht in ($2B/$2C) wird im X- und Y-Register übergeben. Um die richtigen Koppeladressen zu erhalten, muß anschließend noch die Basic-Routine aufgerufen werden, die das für uns erledigt ($A533). Jetzt steht unser Directory genauso im Speicher wie sonst und wartet darauf, auf dem Bildschirm ausgegeben zu werden.



Alle Programme sind durchnumeriert

```
PRUEFSUMMENLISTE
BLOCKGROESSE 20

ZEILE   ANZAHL    SUMME     KEIN POKE?
-------------------------------------
  100      20      2263
  120      40      4991
  130      60      7435
  140      80     10735
  150     100     13594
  160     120     16221
  170     140     19025
  180     160     21080
  190     180     23840
  200     200     26287
  200     220     26327
  210     240     26367
  220     260     27732
  230     280     30452
  240     300     32441
  250     320     34281
-------------------------------------
GESAMT    334     35775
```

Prüfsummen »Lader«

Die Ausgabe kann aber nicht wie beim normalen Listen erfolgen, denn wir wollen ja hinter jedem Namen eine Nummer ausgeben. Also müssen wir das selbst in die Hand nehmen. Mit einem kühnen Sprung geht es deshalb jetzt nach DIROUT. Sehen wir uns nun den Speicherausdruck eines Directory etwas genauer an:

Es geht los bei $0801. Dort steht die Adresse des nächsten Eintrages $081F. Wie immer steht im Speicher zuerst das Low-Byte und dahinter das High-Byte. Es folgt der Diskettenname, der in diesem Falle mit dem des Verfassers auffallend übereinstimmt. Der erste Eintrag beginnt also mit Adresse $081F. Dort finden wir wieder die Adresse des nächsten Eintrages ($083F). Jeder Eintrag hat übrigens die gleiche Länge, nämlich 32 Byte. Wir werden davon später noch Gebrauch machen.

Die 2 Byte hinter der Koppeladresse enthalten die Länge des Programms in Blöcken, in unserem Falle also $0002. Nun kommen ein paar Leerzeichen ($20). Wieviele, hängt davon ab, ob die Blocklänge ein, zwei oder drei Ziffern benötigt. Die Namen sollen ja beim Listen schließlich ordentlich untereinander stehen. Jetzt folgt der Filename, von Anführungszeichen eingeschlossen. Sie haben es sicher schon bemerkt: Das Directory dieser Diskette hat als erstes Programm das »!« wie sich das für anständige Disketten gehört!

Jetzt wieder einige Leerzeichen und schon ist der nächste Fileeintrag erreicht. Das geht so weiter bis eine Koppeladresse erreicht wird ($089D in Adresse $087F), die aus dem Rahmen fällt. In Adresse $089D stehen nämlich zwei jämmerliche Nullen. So, sagt sich das Betriebssystem, jetzt ist aber Schluß. Und es hat wie so oft recht...

Zurück zu unserem Programm. Es soll ja jetzt ein Directory auf den Bildschirm ausgeben, und zwar schöner, als es das Betriebssystem jemals könnte. Wir initialisieren also unsere Variable USE mit dem Basic-Anfang ($0801). In Zeile 390 beginnt eine Schleife, die immer eine Zeile ausgibt. Zuerst holen wir uns die Koppeladresse und speichern sie in NEXT (Zeile 410 — 460). Als nächstes müssen wir die 2 Byte mit der Blocklänge in eine ordentliche Dezimalzahl umwandeln. Das erledigt für uns die Basic-Routine »NUMOUT« (470 — 520). Jetzt brauchen wir nur noch den Namen mit allen Leerzeichen davor und dahinter, auszugeben (530 — 580).

Eigentlich müßte nun die Programmnummer ausgegeben werden, aber damit warten wir noch. Aus folgendem Grund: Wir wollen diese Schleife ja für jede Zeile benützen, auch für die letzte. In der steht aber die »BLOCKS FREE«-Meldung und dahinter darf ja keine Nummer mehr erscheinen. Wir müssen also vorher prüfen, ob wir schon die letzte Zeile erreicht haben.

Dazu übertragen wir die Adresse des nächsten Files, die wir in NEXT aufbewahrt haben, nach USE. Zeigt USE jetzt auf eine Null, sind wir fertig und springen ans Ende unserer Routine, nach DIR4 (590 — 650). Ist das aber noch nicht der Fall, geben wir unsere Programmnummer aus. Den Cursor stellen wir dazu auf Spalte 27 ($1B) und benutzen wieder NUMOUT, um FILENR als Dezimalzahl auszugeben (660 — 700). Eine Zeile wäre geschafft! Im folgenden bereiten wir uns auf die nächste Filenummer vor und springen wieder nach oben zu DIR1 zurück. Sollten aber schon 20 Zeilen auf dem Bildschirm stehen, merken wir uns im ENDFLG, daß wir noch nicht fertig sind und verlassen die Ausgaberoutine vorerst. Erinnern Sie sich, wir wollten vermeiden, daß der Bildschirm ohne unseren ausdrücklichen Befehl wegscrollt (710 — 800).

Der Bildschirm hat sich inzwischen gefüllt, unsere Nummern stehen fein säuberlich hinter den Programmen, nun sollte uns ein freundlich blinkender Cursor dazu animieren, dem Computer mitzuteilen, wie es weitergehen soll. Also hinein in die Routine WAHL.

Das Wichtigste erledigt hier die Betriebssystemroutine CHRIN. Sie läßt den Cursor blinken, bis »RETURN« gedrückt wird, schreibt die eingegebenen Zeichen auf den Bildschirm und liest sie anschließend vom Bildschirm wieder ab, damit wir sie schön der Reihe nach bearbeiten können. Wir speichern alle Zeichen zunächst im BUFFER und sehen uns dann das letzte Zeichen genauer an (820 — 900). Als erstes prüfen wir, ob ein »+« eingegeben wurde. Wenn ja, setzen wir FILEANZ wieder auf Null, und, je nachdem ob das ENDFLG Ende signalisiert oder nicht, auch die FILENR. Anschließend erfolgt der Rücksprung in die Directory-Ausgabe (910 — 980).

```
STARTADRESSE
10 OPEN1,8,15
20 OPEN2,8,2,"#"
30 PRINT#1,"U1 2 0 17 0"
40 PRINT#1,"B-P 2 3"
50 PRINT#2,CHR$(1)
60 PRINT#1,"U2 2 0 17 0"
70 CLOSE2:CLOSE1
READY.
```

Ändern der Startadresse. Lesen Sie hierzu die »Hinweise zum Abtippen«.

Wurde kein »+« eingegeben, prüfen wir weiter auf »L« beziehungsweise »A«. In Abhängigkeit davon setzen wir das ABSFLG (990 — 1070). Jetzt müssen wir herausfinden, welche Programmnummer geladen werden soll. Dazu müssen wir die ein oder zwei Dezimalzahlen in ein Hexbyte umwandeln, weil unser Computer nun mal nichts anderes versteht. Andererseits ist es ja nicht einzusehen, daß wir uns auf sein mathematisches Niveau herabbegeben und unsere Zahlen demnächst als Hex- oder noch schlimmer als Binärzahlen eingeben. Das Umwandeln ist ja gar nicht so schwierig. Wir holen uns die Einerziffer und zählen dann so oft 10 ($0A) dazu, wie die Zehnerziffer angibt. Da der Computer sich unsere Ziffern aber nicht als 0,1...9 merkt, sondern als $30, $31... $3A, müssen wir jeweils die oberen 4 Bits ausmaskieren. Das war's denn auch schon. In FILENR steht zur Belohnung tatsächlich die gewünschte Filenummer mundgerecht für unseren Computer (1080 — 1200).

Jetzt haben wir das Schlimmste — fast — hinter uns. Wir brauchen nur noch den passenden Filenamen zu suchen. Den holen wir uns aus dem Directory. In bewährter Weise benutzen wir unsere Variable USE. Um den Diskettennamen zu überlesen, zählen wir zum Basic-Anfang in $2B/$2C 35 ($23) dazu. Damit liegen wir so ungefähr richtig, auf jeden Fall vor dem ersten Namen. Erinnern sie sich, jeder Eintrag im Directory belegt genau 32 Byte. Allerdings beginnen die Filenamen nicht immer so schön regelmäßig wie in unserem Beispiel, immer an der selben Stelle. Wir müssen den Anfang des Namens also noch genau suchen. Vorher brauchen wir aber erst einmal die richtige Stelle im Directory. Wir addieren zu USE jetzt daher so oft 32 ($20), wie in FILENR angegeben (1250 — 1400).

Jetzt müssen wir in unserem Programm ein Stück übersprungen, weil die folgenden Bytes wegen des Autostarts auf 2 gesetzt werden müssen. Weiter geht es mit LAD2 in Zeile 1780. Vorher steht allerdings im Programm noch die Subroutine FINDA.E, was so viel wie »Finde den Anfang beziehungsweise das Ende des Namens« bedeutet. Sie durchsucht den Text, auf den USE zeigt, nach einem Anführungszeichen und übergibt im Akku, wieviel Zeichen es bis dahin sind (1680 — 1750).

Diese Zahl addieren wir zu USE — eins mehr, denn das Anführungszeichen selbst gehört ja nicht mit zum Namen. Mit FINDA.E erhalten wir im Akku die Länge des Namens, so wie es die Routine SETNAM, die wir oben schon benutzt haben, verlangt. Noch ein kurzer Sprung nach SETLFS, wo wir in Abhängigkeit vom ABSFLG als Sekundäradresse Null oder Eins übergeben. Den Rest erledigt die Basic-Routine BASICLOAD (1780 — 1940). Geschafft, die schönste Zeile im Assemblerlisting ist erreicht: .EN heißt ENDE.

Sicher, es war nicht ganz einfach, aber wenn Sie mir bis hierhin gefolgt sind, haben Sie eine ganze Reihe kleiner Routinen gelernt, wie sie in jedem Maschinenprogramm gebraucht werden.

NUMOUT ($BDCD) — wandelt zwei Hexbytes in eine Dezimalzahl um und gibt sie aus. (HByte im Akku, LByte im X-Reg)
SETLFS ($FFBA) — übergibt logische Filenummer (Akku), Geräteadresse (X-Reg) und Sekundäradresse (Y-Reg).
SETNAM ($FFBD) — übergibt den Filenamen. Akku enthält die Länge des Namens, X-Reg das LByte, Y-Reg das HByte der Adresse, an der der Name beginnt.
LOAD ($FFD5) — lädt das durch SETLFS und SETNAM bezeichnete File an die Adresse, die im X-Reg (LByte) und im Y-Reg (HByte) angegeben ist. Der Akku enthält 0 für LOAD und 1 für VERIFY.
CHRIN ($FFCF) — holt Zeichen von dem in $99 festgelegten Eingabegerät. Der Cursor blinkt, bis RETURN eingegeben wird. Die Zeichen werden dann vom Bildschirm gelesen und im Akku übergeben.
CHROUT ($FFD2) — gibt das ASCII-Zeichen im Akku aus.
Variable
BUFFER — ist ein Zwischenspeicher.
USE und NEXT — enthalten die Adresse des gegenwärtigen beziehungsweise nächsten Directory-Eintrages.
FILENR — ist die aktuelle Filenummer im Directory.
ABSFLG — ist ein Flag, das normales oder absolutes Laden signalisiert.
FILEANZ — enthält die Anzahl der auf dem Bildschirm ausgedruckten Files.
ENDFLG — ist ein Flag, das signalisiert, ob das Ende des Directory bereits erreicht ist.

Betriebssystem-Routinen

Hinweise zum Abtippen

Das Programm kann nicht an die Stelle geschrieben werden, an der es endgültig stehen soll, weil sonst ja der Autostart anlaufen würde. Wir schreiben es daher zuerst nach 49152 ($C000). Tippen Sie zuerst den Basic-Lader ein. Nach RUN muß der Text aus Zeile 30 erscheinen, sonst haben Sie einen Fehler gemacht. Legen Sie nun eine neue Diskette ein, und drücken Sie eine Taste. Das Programm wird nun auf die Diskette geschrieben, allerdings mit der falschen Startadresse. Schalten Sie danach bitte den Computer aus und wieder ein. Um die falsche Startadresse zu ändern, geben Sie das Programm »Startadresse« ein und starten Sie es mit RUN. Jetzt wird die Adresse geändert, und Sie haben das »!« lauffähig auf Ihrer Diskette. Um es auf andere Disketten zu übertragen, verwenden Sie ein Kopierprogramm, zum Beispiel »Super Copy«.

Und dann lehnen Sie sich zurück, geben LOAD »!«,8,1 ein und genießen die Früchte Ihrer Arbeit.

(Dietrich Weineck/rg)

```
0020;**********************
0030;*                     *
0040;* LADEPROGRAMM !      *
0050;* MAI 1984            *
0060;* N.MANN & D.WEINECK  *
0070;* FLEETRADE 40        *
0080;* 2800 BREMEN         *
0090;* TEL. 0421 / 493090  *
0100;*                     *
0110;**********************
0120;
0130;
0140NUMOUT      .DE $BDCD
0150BASICLOAD   .DE $E16F
0160SETLFS      .DE $FFBA
0170SETNAM      .DE $FFBD
0180LOAD        .DE $FFD5
0190CHRIN       .DE $FFCF
0200CHROUT      .DE $FFD2
0210;
0220BUFFER      .DE $0340
0230USE         .DE $39
0240NEXT        .DE $3B
0250FILENR      .DE $FB
0260ABSFLG      .DE $FC
0270FILEANZ     .DE $FD
0280ENDFLG      .DE $FE
0290;
0300            .BA $010A
0310            .MC $C00A
0320            .OS
0330;
0340;
0350DIROUT      LDA *$2B
0360            STA *USE
0370            LDA *$2C
0380            STA *USE+1
0390DIR1        LDA #$0D
0400            JSR CHROUT
0410            LDY #$00
0420            LDA (USE),Y
0430            STA *NEXT
0440            INY
0450            LDA (USE),Y
0460            STA *NEXT+1
0470            INY
0480            LDA (USE),Y
0490            TAX
0500            INY
0510            LDA (USE),Y
0520            JSR NUMOUT
0530            LDY #$03
0540DIR2        INY
0550            LDA (USE),Y
0560            BEQ DIR3
0570            JSR CHROUT
0580            BNE DIR2
0590DIR3        LDA *NEXT
0600            STA *USE
0610            LDA *NEXT+1
0620            STA *USE+1
0630            LDY #$01
0640            LDA (USE),Y
0650            BEQ DIR4
0660            LDA #$1B
0670            STA *$D3
0680            LDA #$00
0690            LDX *FILENR
0700            JSR NUMOUT
0710            INC *FILENR
0720            INC *FILEANZ
0730            LDA *FILEANZ
0740            CMP #$15
0750            BCC DIR1
0760            LDA #$80
0770DIR4        STA *ENDFLG
0780            LDA #$0D
0790            JSR CHROUT
0800;
0810;
0820WAHL        LDX #$00
0830WAHL1       JSR CHRIN
0840            STA BUFFER,X
0850            INX
0860            CMP #$0D
0870            BNE WAHL1
0880            DEX
0890            DEX
0900            LDA BUFFER,X
0910            CMP #'+'
0920            BNE WAHL2
0930            LDA #$00
0940            STA *FILEANZ
0950            BIT *ENDFLG
0960            BMI DIR1
0970            STA *FILENR
0980            BPL DIROUT
0990WAHL2       CMP #'L'
1000            BNE WAHL3
1010            LDA #$00
1020            STA *ABSFLG
1030            BEQ WAHL4
1040WAHL3       CMP #'A'
1050            BNE WAHL
1060            LDA #$80
1070            STA *ABSFLG
1080WAHL4       DEX
1090            LDA BUFFER,X
1100            AND #$0F
1110            STA *FILENR
1120            DEX
1130            BMI LADEN
1140            LDA BUFFER,X
1150            AND #$0F
1160            TAX
1170            CLC
1180WAHL5       LDA *FILENR
1190            ADC #$0A
1200            STA *FILENR
1210            DEX
1220            BNE WAHL5
1230;
1240;
1250LADEN       LDX *FILENR
1260            CLC
1270            LDA *$2B
1280            ADC #$23
1290            STA *USE
1300            LDA *$2C
1310            STA *USE+1
1320LAD1        DEX
1330            BEQ LAD2
1340            CLC
1350            LDA *USE
1360            ADC #$20
1370            STA *USE
1380            BCC LAD1
1390            INC *USE+1
1400            BCS LAD1
1410;
1420;
1430TEXT1       .BY '*'
1440;
1450;
1460            .BY 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
1470            .BY 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
1480            .BY 2 2 2 2 2 2 2 2 2 2 2 2 2 2
1490;
1500;
1510DIRIN       LDA #$01
1520            LDX #L,TEXT1
1530            LDY #H,TEXT1
1540            JSR SETNAM
1550            LDX #$08
1560            LDY #$00
1570            JSR SETLFS
1580            TYA
1590            STA *FILEANZ
1600            STA *FILENR
1610            LDX *$2B
1620            LDY *$2C
1630            JSR LOAD
1640            JSR $A533
1650            JMP DIROUT
1660;
1670;
1680FINDA.E      LDY #$00
1690CMP1        LDA (USE),Y
1700            CMP #$22
1710            BEQ CMP2
1720            INY
1730            BNE CMP1
1740CMP2        TYA
1750            RTS
1760;
1770;
1780LAD2        JSR FINDA.E
1790            SEC
1800            ADC *USE
1810            STA *USE
1820            JSR FINDA.E
1830            LDX *USE
1840            LDY *USE+1
1850            JSR SETNAM
1860            LDA #$01
1870            LDX #$08
1880            LDY #$00
1890            STY *$0A
1900            BIT *ABSFLG
1910            BPL LAD3
1920            LDY #$01
1930LAD3        JSR SETLFS
1940            JMP BASICLOAD
1950            .EN
```

Das Assemblerlisting zum Ladeprogramm