Klaus Kämpf

C P M

Assembler-Listing

2.2

RÖCKRATH
MIKROCOMPUTER

Klaus Kämpf: CP/M 2.2 Assembler-Listing

ISBN 3-925074-11-2

1. Auflage, Dezember 1985

RÖCKRATH
MICROCOMPUTER
Noppiusstr. 19
5100 Aachen

Vorwort

Das Betriebsystem CP/M ist seit Jahren das Standard-Betriebsystem für 8080- und Z80-Mikrocomputer. Viele derzeit erhältliche Bücher, befassen sich ausführlich mit der Bedienung des CP/M und der Nutzung der verschiedenen Betriebssystem-Funktionen.
Das vorliegende Buch bietet nun erstmals einen Einblick in die inneren Abläufe des CP/M 2.2 und beschreibt ausführlich die Arbeitsweise dieses Betriebsystems. Neben einer Beschreibung der internen Datenformate und der Diskettenparameter, wird jede BDOS-Funktion in ihrem Ablauf dokumentiert. Daran schliessen sich die kommentierten Source-Listings der beiden CP/M-Teile CCP und BDOS an. Für diejenigen Anwender, die sich ein CP/M-BIOS selber schreiben wollen, werden noch Tips und Tricks zur Verbesserung des CP/M 2.2 gegeben.

Aachen, im November 1985

Klaus Kämpf

Inhaltsverzeichnis

Einleitung

Die Geschichte des Betriebsystems CP/M ist eng verknüpft mit der Entwicklung der Mikroprozessoren und Mikrocomputer. Die folgende Einleitung soll daher einen kurzen Rückblick auf die Entstehung und weitere Entwicklung des CP/M geben:

Im Herbst 1973 kündigte die amerikanische Elektronikfirma Intel, einen Mikroprozessor mit der Bezeichnung '8080' als Nachfolger ihres Prozessors '8008' an.
Der 8080 war damals der erste Mikroprozessor, der nicht nur diskrete Logik ersetzte, sondern auch zum Einsatz als Zentraleinheit (CPU) für Mikrocomputer geeignet war. Im April 1974 kamen dann die ersten Exemplare des 8080 zu einem Preis von etwa 360 Dollar auf den Markt.

Einer der ersten Mikrocomputer mit dem 8080 als Zentraleinheit war der 'Altair 8800'.
Der Altair 8800 wurde im Januar 1975 als Bausatz von der Zeitschrift 'Popular Electronics' vorgestellt. Inclusive Prozessor, Netzteil, Hauptplatine, binärer Ein- und Ausgabe und einem Speicher von 256 Bytes (!) kostete dieser Bausatz 395 Dollar (damals ca. 1500 DM).

Der Altair 8800 war aus mehreren Einzelplatinen aufgebaut, die alle auf einer gemeinsamen Busplatine steckten. Ein großer Vorteil dieses Aufbaus war, daß auch Zusatzplatinen anderer Firmen betrieben werden konnten.
'Digital Microsystems' war die erste Firma, die einen 'Disk-Controller' zum Anschluss von 8 Zoll Diskettenlaufwerken an den Altair 8800 anbot und auch ein Betriebssystem dazu lieferte. Unter der Bezeichnung 'Control Program for Microcomputer', kurz 'CP/M', wurde dieses Betriebssystem auch getrennt verkauft und entwickelte sich rasch zu einem Standard-Betriebssystem für 8080-Mikrocomputer.

Das erste CP/M (Versionsnummer 1.3, später 1.4) war auf folgende Grundkonfiguration abgestimmt:

- Einen Fernschreiber (Teletype, TTY) oder ein Bildschirmterminal (Cathode Ray Tube, CRT) zur Ein- und Ausgabe

- Einen Lochstreifenleser (Paper Tape Reader, RDR) und -stanzer (Paper Tape Punch, PUN) als Hintergrundspeicher

- 8 Zoll Diskettenlaufwerke im IBM 3740-Standard als Massenspeicher

Gedacht war ein solches System vor allem zur Entwicklung von Programmen für den 8080 Mikroprozessor. Noch heute zeugen die von Digital Research mitgelieferten Programme ED, ASM oder DDT davon.

Grundlagen

CCP, BDOS, BIOS

Um die Größe des Betriebssystems möglichst klein zu halten, wurde das CP/M in drei verschiedene Teile aufgespalten:

- CCP (Console Command Processor)
  'Befehls-Bearbeiter'. Zuständig für die Interpretierung der Eingabe und Ausführung der grundlegenen Befehle wie DIR, ERA etc.

- BDOS (Basic Disk Operating System)
  Betriebssystem-Kern. Zuständig für die Verwaltung der Laufwerke und der gesamten Ein- und Ausgabe.

- BIOS (Basic Input Output System)
  Systemabhängiger Teil. Zuständig für die Ansteuerung der hardwareseitigen Systemkomponenten.

Der CCP bildet die Benutzerebene des Betriebssystems, über die der Anwender Befehle eingeben und Programme starten kann. Da nach dem Start eines Programms die Benutzerebene überflüssig ist, kann der vom CCP eingenommene Speicherplatz vom jeweils laufenden Programm mitbenutzt werden.
Der CCP ist ein in sich abgeschlossenes Programm, daß, wie normale CP/M-Programme auch, Funktionen des BDOS benutzt.

Das BDOS ist der eigentliche Betriebssystem-Kern und auf allen CP/M Rechnern identisch. Dadurch ist die Kompatibilität zwischen allen Rechnern die CP/M verwenden garantiert.

Das BIOS bildet die 'Schnittstelle' zwischen dem BDOS und der Hardware des Computers.

Notwendig für Anwenderprogramme sind nur die beiden Teile 'BDOS' und 'BIOS'. Sie werden zusammen auch als 'FDOS' bezeichnet.

Speicheraufteilung

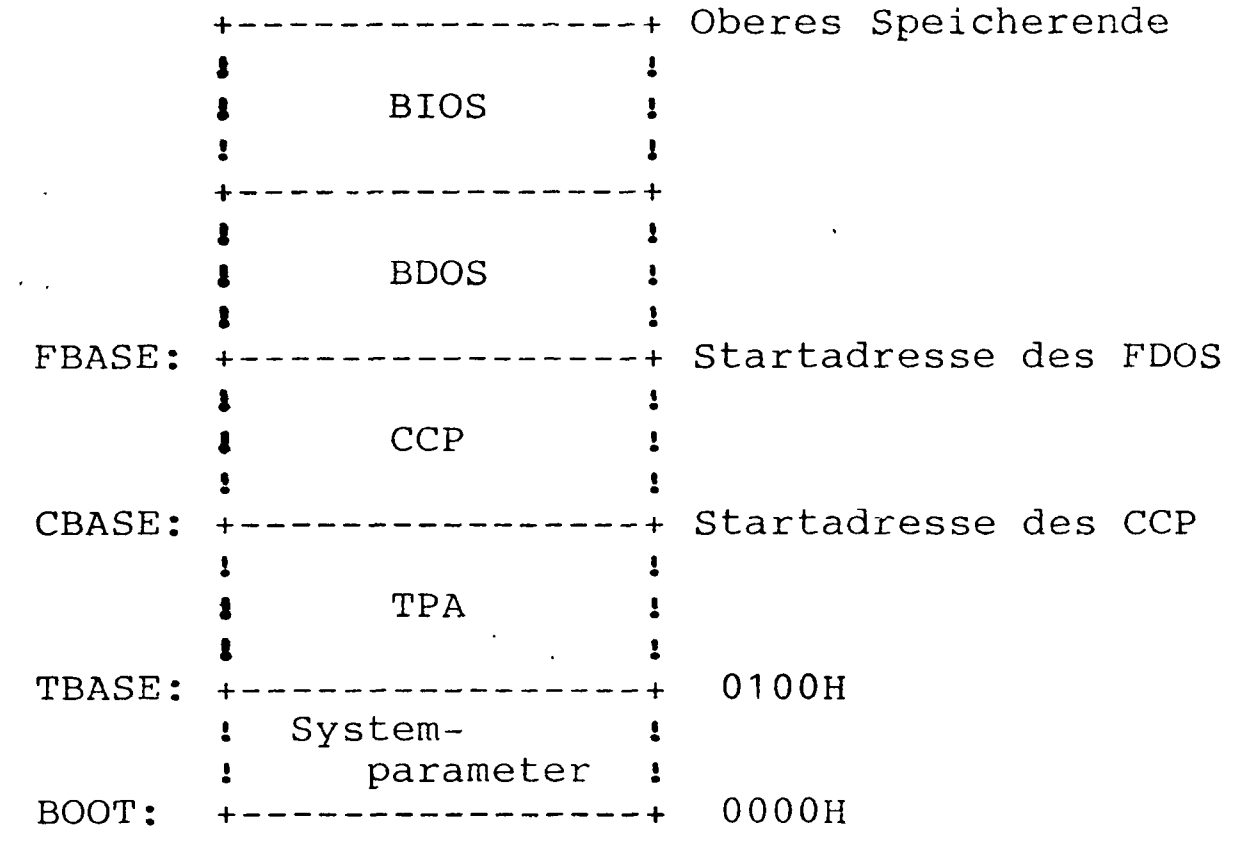
Damit das CP/M in verschiedenen Speichergrößen ablaufen kann, ist der Speicherbereich in dem das CP/M liegt nicht festgelegt. Statt dessen ist die erste Speicherseite (256 Bytes) für Systeminformationen reserviert und beinhaltet auch den 'Bezugspunkt' zum CP/M in Form eines Sprungbefehls.

Ab der zweiten Seite beginnt der Speicherbereich für Anwenderprogramme, die sogenannte 'Transient Program Area', kurz TPA. Die TPA reicht bis zur ersten Adresse des CP/M, daß immer am oberen Speicherende liegt.

Wichtig für den Betrieb von CP/M ist, daß der Speicher durchgehend ist und bei der Adresse 0000H beginnt. Da Anwender-

programme immer ab einer fester Adresse arbeiten, ist dadurch eine einheitliche Startadresse der TPA festgelegt, was erst einen Programmaustausch zwischen CP/M-Rechnern ermöglicht.

Die Speicheraufteilung im CP/M sieht im Überblick so aus:



An der Adresse BOOT befindet sich grundsätzlich ein Sprung zur Warmstartroutine des BIOS. Diese Routine lädt nach Beendigung eines Programms den CCP und das BDOS neu in den Speicher und startet danach wieder den CCP. Das Sprungziel ist immer der zweite Eintrag in der BIOS-Sprungtabelle, also BIOS + 0003H.
Daraus kann ein Programm auch die Startadresse des BIOS berechnen und, falls es das BDOS nicht braucht, auch den vom BDOS belegten Speicherbereich nutzen.

Läßt ein Programm den Speicherplatz über CBASE unberührt, so kann es auch durch ein einfaches 'RET' die Kontrolle wieder zurück an den CCP geben.

An der Adresse BOOT + 5 (normalerweise 0005H) steht ein Sprung nach FBASE, der die Verbindung zwischen Programmen und dem FDOS herstellt. Auch der CCP benutzt diesen Sprungbefehl bei der Anforderung von FDOS-Funktionen.
Dieses Sprungziel kann aber durch gewisse Programme (z.B. DDT) verändert werden.

Im normalen CP/M-Gebrauch ist die Zieladresse dieses Sprunges der niedrigste, vom FDOS verwendete Speicherplatz. Rechnet man den Speicherplatz des CCP mit zur TPA, so ist diese Adresse genau die höchste TPA-Adresse +1.

Das DDT-Programm setzt dies insofern voraus, als daß es sich selbst 'unter' das FDOS legt und den Sprung nach FBASE auf sich selbst legt. Startet man im DDT ein Programm, so ist der DDT geschützt, solange das Programm die Sprungadresse als obere Speichergrenze benutzt. Außerdem laufen im Trace-Modus des DDT alle BDOS-Aufrufe über den DDT, der damit die Kontrolle über das System behält.

Disketten

Das CP/M ist als single-user, single-tasking Betriebssystem ausgelegt. Das heißt, daß das CP/M zu einer Zeit nur einen Benutzer und ein Programm bearbeiten kann und damit auch immer nur ein Laufwerk und eine Datei (File) 'kennt'.

Jedes Laufwerk muß daher vor der Bearbeitung beim BDOS angemeldet werden, daß sich daraufhin auf dieses Laufwerk einstellt. Die zu einem Laufwerk gehörenden Daten und Tabellen sind im BIOS gespeichert, daß sie auf Anfrage dem BDOS zur Verfügung stellt.

Das Anmelden dient dem BDOS zur Einstellung seiner internen Parameter auf das neue Laufwerk bzw. auf die im Laufwerk befindliche Diskette. Über verschiedene BDOS-Funktionen kann die Diskette dann gelesen und geschrieben werden.

Zum Schreiben muß dem BDOS aber bekannt sein, welche Teile der Diskette bereits Daten enthalten und welche noch frei sind. Aus diesem Grund wird bei der Laufwerks-Anmeldung eine Belegungstabelle der Diskette erstellt, die Auskunft über freie und belegte Teile gibt.

Zur Speicherung neuer Daten kann das BDOS dann die benötigten Informationen dieser Tabelle entnehmen.
Da sich das BDOS noch zusätzlich 'merkt', ob ein Laufwerk schonmal angemeldet wurde oder nicht, braucht die Belegungstabelle nur dann erstellt zu werden, wenn das Laufwerk zum ersten Mal angewählt wird.

Das BDOS beschreibt eine Diskette immer anhand der vorliegenden Belegungstabelle, was bei einem unkontrollierten Diskettenwechsel fatale Folgen haben kann.
Zur Umgehung dieses Problems verfügt das CP/M über den sogenannten Warmstart.

Durch einen Warmstart wird das BDOS neu geladen und dadurch sein 'Gedächtnis' der bekannten Laufwerk gelöscht. Bei jeder folgenden Laufwerksanwahl wird dadurch die Belegungstabelle jeweils neu erstellt.

Zum Diskettenwechsel ohne Warmstart, bietet das BDOS auch die Möglichkeit einzelne Laufwerke ab- und neu wieder anzumelden. Der Warmstart nach einem Diskettenwechsel ist immer Aufgabe

des Benutzers; wenn er den Warmstart vergisst, tritt das Problem wieder auf.

Da Datensicherheit in einem Betriebssystem nunmal das Wichtigste ist, ist im CP/M noch ein zweiter Schutzmechanismus eingebaut, die Prüftabelle.

Bei der Anmeldung eines Laufwerks wird, neben der Belegungstabelle, auch eine Prüfsumme über das Inhaltsverzeichnis der Diskette erstellt und gespeichert.
Vor jeder Schreiboperation bildet das BDOS diese Prüfsumme neu und vergleicht sie mit der zuletzt für dieses Laufwerk errechneten Prüfsumme. verglichen.
Anhand der Differenz dieser beiden Summen, erkennt das BDOS einen Diskettenwechsel und sperrt jeden weiteren Schreibzugriff auf das Laufwerk.

Aus diesem Grund müssen im CP/M Diskettenwechsel dem System immer ausdrücklich mitgeteilt werden. Das Anmelden einer Diskette wird auch als 'einloggen' (engl. Log in) und das Abmelden als 'ausloggen' (engl. Log out) bezeichnet.

Dateien

Jede Diskette ist physikalisch in Spuren (engl. Tracks) und Sektoren (engl. Sectors) aufgeteilt.
Der Datentransfer von und zu der Diskette spielt sich immer in ganzen Sektoren ab. Aufgabe des Programms ist es, den Sektorinhalt zu interpretieren und die einzelnen Bytes auszulesen.

Zur Vereinfachung von Diskettenoperationen werden im CP/M - wie in anderen Betriebssystemen auch - zusammenhängende Informationen in Dateien (engl. Files) zusammengefasst.

Jede Datei hat einen Dateinamen (engl. Filename) und kann unter diesem Namen bearbeitet werden. Die Ansteuerung einzelner Spuren und Sektoren auf der Diskette wird vom Betriebssystem übernommen.

Dateinamen bestehen im CP/M aus bis zu 8 Zeichen, gefolgt von bis zu 3 Zeichen zur Kennzeichnung eines Dateityps (engl. Filetype). Filename und Filetyp werden meist auch zusammen als Filename bezeichnet.

Die Directory

Die Information, welche Daten bzw. welche Files auf einer Diskette aufgezeichnet sind, ist am Anfang der Diskette im Inhaltsverzeichnis (engl. Directory) enthalten.

In der Directory stehen alle wichtigen Daten die das BDOS zur Bearbeitung der Diskette benötigt. Für jedes File sind dies:

- Der Filenamen und Filetyp
- Die Länge des Files
- Die vom File belegten Bereiche auf der Diskette.

Fordert ein Programm bestimmte Daten unter Angabe eines Filenamens an, so kann das BDOS aus den Directory-Informationen den exakten Sektor auf der Diskette berechnen.

Blöcke

Das BDOS teilt jede Diskette in Blöcke (engl. Blocks) auf, um damit den Verwaltungs- und Speicheraufwand für die Belegungstabelle zu verkleinern.

Während das CP/M 1.4 noch eine feste Blocklänge von 1 Kilobyte hatte, ist die Länge eines Blocks im CP/M 2.2 variabel gehalten.

Da die Belegungstabelle in Blöcken geführt wird, kann das BDOS Diskettenplatz auch nur blockweise vergeben. Nachteil dieser Aufteilung ist, daß ein File immer ganze Blöcke belegt, auch wenn die tatsächliche Filelänge kleiner ist.

Records

Das CP/M 1.4 baut auf der physikalischen Sektorlänge des IBM 3740 Formates (128 Bytes pro Sektor) auf.
Alle Dateioperationen geschehen im CP/M 1.4 in 128 Byte 'Portionen' und auch die Länge eine Files wird in Vielfachen von 128 Bytes gemessen.

CP/M 2.2 hat diese Rechenweise für Files übernommen, unterscheidet aber zwischen Sektoren auf der Diskette und File-'Portionen'.
Ein Sektor ist im CP/M 2.2 die kleinste physikalische Aufzeichnungseinheit auf der Diskette. Eine Datei-'Portion' wird im CP/M 2.2 als Record bezeichnet. Ein Record ist der kleinste Teil einer Datei den CP/M noch addressieren bzw. unterscheiden kann.

Die interne Bearbeitung von Disketten geschieht im CP/M 2.2 immer in einzelnen Records. Der Name 'Record' wird daher im weiteren nicht nur in Verbindung mit Files, sondern auch allgemein für Diskettendaten benutzt.

Im Zusammenhang mit Disketten ist auch die Bezeichnung 'logischer Sektor' für einen Record üblich. Damit wird vor allem der Unterschied zwischen einem Sektor auf der Diskette ('physikalischer Sektor') und einem 'Sektor', wie ihn das BDOS

verarbeitet, deutlich gemacht.

Obwohl das BIOS eigentlich den physikalischen Teil des CP/M bildet, geschieht Datentransfer zwischen BDOS und BIOS in logischen Sektoren.
Zum einen wird, z.B. beim Laden einer Datei in den Speicher, am BDOS 'vorbei' geladen. Das heißt, daß das BDOS dem BIOS nur mitteilt wohin der nächste logische Sektor geladen werden soll und den Rest dem BIOS überlässt.
Der zweite Grund ist, daß das neue CP/M 2.2 nach der Einführung möglichst schnell Verbreitung finden sollte. Dazu musste der Aufwand zur Umstellung eines CP/M 1.4 BIOS auf die CP/M 2.2 Version klein gehalten werden. Da das CP/M 1.4 nur mit dem IBM 3740 Diskettenformat arbeitet, können die Diskettenein- und ausgaberoutinen vom CP/M 1.4 BIOS zum CP/M 2.2 BIOS übernommen werden.

Im CP/M 2.2 hat das BIOS die Aufgabe zwischen physikalischen und logischen Sektoren zu trennen.
Der bei einer physikalischen Sektorlänge von mehr als 128 Bytes notwendige Aufwand ist nicht ganz unerheblich.
Das BIOS muß beim Lesen jeden physikalischen Sektor in logische Sektoren aufspalten und beim Schreiben logische Sektoren zu einem physikalischen Sekto zusammenfassen.

CP/M 2.2 unterstützt dieses Zusammenfassen und Aufspalten (engl. Blocking and Deblocking) durch einen weiteren Parameter, der den Lese/Schreibroutinen im BIOS übergeben wird.

Directory-Einträge

Die Directory enthält alle Informationen über die auf der Diskette gespeicherten Files. Dazu gehören neben dem Filenamen und Filetyp, die Länge des Files und die von ihm belegten Blöcke.

Jedes File benutzt mindestens einen Eintrag (engl. Entry) innerhalb der Directory. Die Länge eines Eintrags ist im CP/M auf 32 Bytes festgelegt. Damit passen in jeden Directory-Record genau 4 Einträge.
Directory-Funktionen im BDOS geben daher meist einen Directory-Code zurück, der die relative Nummer eines Eintrages im Directory-Record angibt. Über diese Nummer können Programme einen Eintrag in der Directory 'finden' und selbst bearbeiten.

Als Directory-Bereich auf der Diskette, ist im CP/M 1.4 der erste Block reserviert. Damit beträgt die Höchstzahl an Einträgen auf einer CP/M 1.4 Diskette 64. Im CP/M 2.2 ist die Größe der Directory in gewissen Grenzen frei wählbar.
Es können bis zu 16 Blöcke als Directory reserviert werden und auch die Anzahl der Einträge ist ein getrennter Parameter. Bei der maximalen Blockgröße von 16 kbyte, sind im

Grundlagen

CP/M 2.2 maximal 16384 Directory-Einträge möglich

Userbereiche

Zur besseren Handhabung größerer Directories gibt es im CP/M 2.2 sogenannte User-Bereiche (engl. User Areas).

Jeder Directory-Eintrag enthält eine zusätzliche Kennung, die Usernummer (engl. User Number).

Das BDOS arbeitet immer nur innerhalb eines Userbereichs und erkennt nur die Directory-Einträge, die innerhalb des aktuellen Userbereichs liegen, d.h. die entsprechende Usernummer haben.
Eine Usernummer kann daher nicht einzelnen Files zugewiesen werden, sondern immer nur dem BDOS als ganzem.

User-Bereiche können vom CCP aus mit dem 'USER'-Befehl angewählt werden. Das BDOS unterscheidet zwar bis zu 32 User-Bereiche, der CCP erlaubt jedoch nur die Anwahl der ersten 16.

Öffnen und Schliessen von Files

Zum Arbeiten mit einem File benötigt das BDOS alle Informationen des zum File gehörenden Directory-Eintrags.
Zu diesem Zweck muß ein Programm, will es mit einem File arbeiten, dieses File erst eröffnen (engl. open).

Dazu stellt das Programm dem BDOS einen 32 Byte großen Speicherbereich - den Datei Kontroll Block (engl. File Control Block, FCB) - zur Verfügung.

Durch diese Speicherung der Directory-Informationen in einem getrennten Bereich werden zwei Ziele erreicht:
Erstens wird die Anzahl der Directory-Zugriffe erheblich reduziert, da das BDOS die Daten des Directory-Eintrags immer dem FCB entnimmt.
Zweitens können dadurch, im Widerspruch zur 'Ein-Benutzer-Struktur' des BDOS, mehrere Files quasi-gleichzeitg bearbeitet werden.
Bei jeder Fileoperation wird dem BDOS die Adresse des jeweiligen FCB übergeben. Das BDOS entnimmt daraufhin die wichtigsten Daten direkt dem FCB und speichert sie intern. Dadurch wird - aus der Sicht einer BDOS-Funktion - immer nur ein File zu einer Zeit bearbeitet.
Nach Abschluß der Fileoperation schreibt das BDOS die Daten zurück in den FCB, der damit immer die aktuellen Daten enthält.
Ist die Bearbeitung eines Files beendet, so wird es wieder geschlossen (engl. close).

Beim Schliessen eines Files schreibt das BDOS die Informationen des FCB wieder zurück in den Eintrag.
Dieses Zurückschreiben ist aber nur dann wichtig, wenn die Informationen des Eintrags (bzw. FCB) wirklich geändert wurden. Im CP/M kann nur die Schreibfunktion die Größe eines Files und damit die Daten des FCB beeinflussen.

BDOS-intern erhält daher jeder FCB eine getrennte Kennung, die Auskunft darüber gibt, ob eine Schreiboperation auf diesem FCB ausgeführt wurde oder nicht.
Nach dem Öffnen wird diese Kennung auf 'nicht geschrieben' bzw. 'nicht geändert' (engl. not altered) gesetzt und bei einer Schreiboperation gelöscht.
Die BDOS-Funktion 'File schließen' aktualisiert einen Eintrag nur, wenn diese Kennung gelöscht ist.

Eintragsnummern

Im CP/M 1.4 kann eine Diskette bis zu 256 Blöcken, bei 1k Blockgröße entsprechend 256 kbytes, umfassen. Blocknummern werden im CP/M 1.4 immer als 8-Bit Werte verwaltet.

Von den 32 Bytes eines Eintrags sind 16 Bytes für die Speicherung der vom File belegten Blocknummern vorgesehen. Pro Eintrag können damit im CP/M 1.4 maximal 16 Blöcke adressiert werden, was einer Filegröße von 16 kbytes pro Eintrag entspricht. Für jeweils 16k Daten ist daher im CP/M 1.4 ein eigener Directory- Eintrag nötig.

In Records ausgedrückt entspricht die maximale Filegröße pro Eintrag bzw. FCB, 128 Records (128 * 128 Bytes = 16k Bytes).
Da Recordnummern im CP/M immer relativ gezählt werden, ist der Bereich der Recordnummern - bezogen auf den Eintrag - 0 bis 127. Bei Erreichen der Recordnummer 128 wird im CP/M 1.4 automatisch der nächste Eintrag aus der Directory gelesen und die Nummerierung der Records wieder bei 0 angefangen.
Jeder zu einem File gehörende Eintrag wird daher noch mit einer Eintragsnummer versehen.

Zur Anwahl eines bestimmten Records innerhalb eines Files, ist also sowohl die Record- als auch die Eintragsnummer anzugeben.
Die maximale Anzahl von Einträgen pro File ist im CP/M 1.4 auf 16 begrenzt, sodaß ein File höchstens eine Größe von 256 kbytes erreichen kann.

Extends

Diese Zählweise der Eintrags- und Recordnummern wurde beim Übergang von CP/M 1.4 zu CP/M 2.2 aus Kompatibilitätsgründen beibehalten.

Da das CP/M 2.2 aber mit verschiedenen Blockgrößen arbeiten

kann, trifft die Bezeichnung 'Eintragsnummer' nicht mehr zu. Das CP/M 2.2 trennt aus diesem Grund zwischen den Begriffen Extend und Eintrag.
Ein Extend im CP/M 2.2 entspricht dem Adressierungsbereich eines Eintrags im CP/M 1.4 - 16 kbyte.
Ein Eintrag im CP/M 2.2 dagegen kann, bei der maximalen Blockgröße von 16 kbytes, bis zu 16 Extends enthalten.

Intern werden Files im CP/M 2.2 grundsätzlich in Extends verwaltet, um so die Zählweise der Recordnummern von 0 bis 127 zu erhalten.

Extendgruppen

Das CP/M 2.2 fasst noch zusätzlich jeweils 32 Extends zu einer Extendgruppe (engl. Extend Group) zusammen.
Ein File kann aus bis zu 16 Extendgruppen aufgebaut sein, was die maximale Länge eines Files auf 65536 Records (8 Megabyte) begrenzt.

Die laufende Recordnummer eines Files wird BDOS-intern immer in den drei Teilen Extendgruppe, Extend und Record verwaltet.

Aufbau eines Directory-Eintrags

Ein Eintrag umfasst 32 Bytes und enthält alle Informationen, die das BDOS zur Lokalisierung der Daten auf der Diskette benötigt. Das folgende Schema soll den prinzipiellen Aufbau eines Eintrages im CP/M 2.2 verdeutlichen:

```
     +-----------------------------------------------+
     ! UN N1 N2 N3 N4 N5 N6 N7 N8 T1 T2 T3 EX S1 EG RC !
     +-----------------------------------------------+
Pos.   0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15

     +-----------------------------------------------+
     ! B0 . . . . . . . . . . . . . . . . . . . . Bn !
     +-----------------------------------------------+
Pos.  16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
```

mit

UN	Usernummer des Files (0..31)
N1...N8	Filename (8 Zeichen)
T1...T3	Filetyp (3 Zeichen) und Fileattribut
EX	Höchste Extendnummer des Eintrags (0..31)
S1	unbenutzt, immer 0
EG	Höchste Extendgruppe des Eintrags (0..15) (auch als S2 bezeichnet)
RC	Anzahl der Records im Extend EX (0..127)
B1...Bn	Nummern der belegten Blöcke (Blocktabelle)

Die erste Position des Eintrags enthält die Nummer des Userbereiches zu dem das File gehört. Der Wert E5H an dieser Stelle weist den Eintrag als unbelegt aus.

Die Positionen N1 bis N8 bzw. T1 bis T3 sind Filename und Filetyp des zum Eintrag gehörenden Files in Großschrift.

Umfasst ein Filenamen weniger als 8 Zeichen, so werden die nicht benutzten Stellen mit Leerzeichen aufgefüllt. Gleiches gilt auch für den Filetyp.

Da Filenamen und Filetyp ASCII-Wert sind, werden immer nur die unteren 7 der 8 Bits an den Positionen N1 bis N8 und T1 bis T3 ausgenutzt. Die höchsten Bits an den Positionen T1 bis T3 werden zur Kennzeichnung eines Fileattributes verwendet. CP/M 2.2 nutzt von den drei möglichen Attribut-Bits aber nur T1 und T2 für die Attribute 'Schreibgeschützt' und 'System-

file' aus. Das dritte Bit kann von Anwenderprogrammen für eigene Zwecke genutzt werden.

Bit 7 der Position T1 weist ein File als 'schreibgeschützt' aus. Vor jedem Schreibzugriff wird dieses Bit getestet und, falls es auf '1' steht, der Zugriff mit der Fehlermeldung 'FILE R/O' abgebrochen.
Bit 7 der Position T2 zeigt das Attribut 'Systemfile' an. Dieses Attribut wird jedoch nicht vom BDOS, sondern nur vom 'DIR'-Befehl im CCP beachtet.

EX enthält die Extendnummer, EG die Extendgruppe des letzten durch den Eintrag abgedeckten Record. EX und EG werden, bei mehreren Einträgen zu einem File, immer relativ zum File-anfang gerechnet

RC ist die Anzahl von Records im höchsten Extend (EX) des Eintrages. RC liegt normalerweise im Bereich zwischen 0 und 127.
Ein Wert von 128 in RC zeigt an, daß auch der letzte Extend mit 128 Records gefüllt ist. In diesem Fall folgt noch ein weiterer Eintrag (Folgeeintrag).

In der Blocktabelle stehen die vom File belegten Block-nummern.
Die Blocktabelle kann bei 8-Bit Blocknummern bis zu 16 Blöcke und bei 16-Bit Blocknummern bis zu 8 Blöcke umfassen.

Aufbau eines File Control Blocks

Der Aufbau eines File Control Blocks entspricht dem eines Directory-Eintrags.
Vor dem Öffnen eines Files enthält der FCB nur das Laufwerk, den Filename und den Filetyp des gewünschten Files. Zusätzlich kann noch eine Extendnummer angegeben werden, sodaß ein bestimmter Extend schon beim Öffnen erreicht wird.

Die Anwahl einer Extendgruppen ist nicht direkt möglich; geöffnet wird immer die erste Extendgruppe.

Zum Öffnen muß der FCB folgende Daten enthalten:

```
      +-----------------------------------------   ------+
      ! LW N1 N2 N3 N4 N5 N6 N7 N8 T1 T2 T3 EX  0 ... 0  0 !
      +-----------------------------------------   ------+
Pos.    0  1  2  3  4  5  6  7  8  9 10 11 12 13    31 32
```

mit

LW	Laufwerkscode
	0 - Aktuelles Laufwerk
	1 - Laufwerk A
	2 - Laufwerk B
	...
	16 - Laufwerk P
N1...N8	Filename (8 Zeichen)
T1...T3	Filetyp (3 Zeichen)
EX	Extendnummer

Die 'File öffnen'-Funktion im BDOS durchsucht die Directory des gewünschten Laufwerks nach einem passenden Eintrag und kopiert diesen in den FCB. Alle weiteren Filefunktionen des BDOS stützen sich nur noch auf die im FCB enthaltenen Informationen.

Ein geöffneter FCB bezieht sich immer auf einen bestimmten Extend des Files. Die genaue Recordnummer innerhalb dieses Extends wird in einem weiteren Byte an der Position 32 gespeichert. Durch Vergleich dieser Recordnummer mit der höchsten Recordnummer des Extends kann das BDOS entscheiden, wann das Ende eines Extends erreicht ist. Die genaue Belegung der 33 Bytes eines geöffneten FCB gibt das folgende Schema:

FCB nach dem Öffnen:

```
      +-----------------------------------------------+
      ! LW N1 N2 N3 N4 N5 N6 N7 N8 T1 T2 T3 EX S1 EG RC !
      +-----------------------------------------------+
Pos.    0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15

      +-----------------------------------------------+
      ! B0  . . . . . . . . . . . . . . . . Bn CR !
      +-----------------------------------------------+
Pos.  16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
```

mit

LW	Nummer des Laufwerks 0 = Aktuelles Laufwerk 1 = Laufwerk A 2 = Laufwerk B ... 16 = Laufwerk P
N1...N8	Filename (8 Zeichen)
T1...T3	Filetyp (3 Zeichen)
EX	Aktuelle Extendnummer (0..31)
S1	unbenutzt, immer 0
EG	Aktuelle Extendgruppe (0..15)
RC	Anzahl der Records im Extend EX (0..127)
B1...Bn	Nummern der belegten Blöcke (Blocktabelle)
CR	Aktuelle Recordnummer im Extend EX (0..127)

Über die Position LW können vom BDOS auch nach dem Öffnen noch Files von verschiedenen Laufwerken unterschieden werden.

Ist LW nicht 0, so wird vor jeder Fileoperation das gewünschte Laufwerk angewählt und nach Abschluss der Operation wieder auf das alte Laufwerk zurückgeschaltet.
Nach der Anwahl des Laufwerks speichert das BDOS den Wert der LW-Position und setzt dort die aktuelle Usernummer ein. Dadurch entsprechend sich - BDOS-intern - FCB und Eintrag in den ersten 12 Positionen.

Die nächsten vier Bytes sind zwischen FCB und Eintrag unterschiedlich: Da bei Fileoperationen immer mit einzelnen Records gearbeitet wird, beschreibt ein FCB - anders als ein Eintrag - keine Filegröße, sondern einen bestimmten Record.

EG und EX bilden im FCB die Nummer des Extends, in dem der zu bearbeitende Record liegt. RC ist Anzahl von Records in diesem Extend.
Die genaue Recordnummer innerhalb des Extends ist in der zusätzlichen Position CR enthalten.

B1 bis Bn entsprechend genau der Blocktabelle des Eintrags. Aus dieser Blocktabelle entnimmt das BDOS während der Bearbeitung die Blocknummer für den Diskettenzugriff.

Sequentieller Zugriff <-> Direkter Zugriff

Im CP/M 1.4 ist es Aufgabe des Programms die Eintrags- und Recordnummer im FCB zu setzen. Soll ein bestimmter Record innerhalb des Files erreicht werden, so muß das Programm die Extend- und Recordnummer selbst berechnen und in den FCB einsetzen.
Aus der absoluten Recordnummer ergibt sich die Extendnummer aus der Division durch 128. Der Divisionrest ist die Recordnummer innerhalb des Extends. Diese Werte müssen aber immer vor dem Öffnen gesetzt sein, damit das BDOS die richtige Blocktabelle in den FCB überträgt.

Alle höheren Recordnummern können im CP/M 1.4 nur sequentiell (nacheinander) erreicht werden.
Die im CP/M 1.4 benutzte Zugriffsart auf ein File, trägt daher die Bezeichnung sequentieller Zugriff (engl. Sequential Access). Die direkte Anwahl eines beliebigen Records ist im sequentiellen Zugriff nicht möglich.

CP/M 2.2 bietet daher mit dem direkten Zugriff (engl. Random Access) eine weitere Zugriffsart, mit der jeder Record eines Files direkt erreicht werden kann.

Beim Random Access wird über zusätzliche Stellen im FCB eine absolute Recordnummer zwischen 0 und 65535 dem BDOS übergeben.

Zur Aufnahme der Recordnummer beim Random Access, wird der FCB ab der Position 33 um die 3 Bytes R0, R1 und R2 erweitert.
R0 und R1 enthalten die 16-Bit Recordnummer mit dem niederwertigen Byte in R0. R2 dient bei der Umrechnung zwischen SRN und RRN der Kennung eines Überlaufs und sollte immer auf Null gesetzt werden.

Das BDOS berechnet aus dieser 'Random Record Number' (RRN) die 'Sequential Record Number' (SRN), bestehend aus Extendgruppe, Extend und Recordnummer.

Datenformate

Zwischen der Random Record Nummer und der Sequential Record Nummer besteht folgender Zusammenhang:

RRN -> SRN:

RRN = EG * 32 * 128 + EX * 128 + CR

SRN -> RRN:

EG = RRN / 4096 = RRN / (32 * 128)

EX = (RRN / 128) modulo 32

CR = RRN modulo 128

Der Disk Parameter Header

Das CP/M 2.2 erlaubt den Anschluß von bis zu 16 Laufwerken unterschiedlicher Kapazität. Die zu jedem Laufwerk gehörenden Parameter werden vom BIOS verwaltet und dem BDOS zur Verfügung gestellt.

Bei der Anwahl eines Laufwerks über das BIOS, erhält das BDOS die Adresse eines Datenbereiches mit der Bezeichnung DPH. DPH ist die Abkürzung für Disk Parameter Header, was soviel wie 'Disketten-Parameter-Anfang' bedeutet.

Der DPH ist der 'Anfang' der Diskettenparameter und enthält, neben Daten, noch Verweise zu weiteren Tabellen.

Ein DPH umfasst 8 Einträge zu je 16 Bit und hat folgende Struktur:

	XLT	NHDE	CLTK	FSCT	DIRBUF	DPB	CSV	ALV
Byte	0/1	2/3	4/5	6/7	8/9	A/B	C/D	E/F

mit

XLT — Adresse der Sektor-Verschränkungs-Tabelle
(engl. X-lation Table)

NHDE — Nummer des letzten belegten Eintrags in der Directory der Diskette +1
(engl. Number of Highest Directory Entry)

CLTK — Aktuelle (zuletzt angewählte) logische Spurnummer des Laufwerks
(engl. Current Logical TracK)

FSCT — Absolute (logische) Recordnummer des ersten Records der aktuellen Spur
(engl. First Sector of Current Track)

DIRBUF — Adresse eines 128-Byte Buffers für Directory-Operationen
(engl. DIRectory BUFfer)

DPB — Adresse des Disk Parameter Blocks
(engl. Disk Parameter Block)

CSV — Adresse des Prüfsummenvektors
(engl. Check Sum Vector)

ALV — Adresse des Belegungsvektors
(engl. ALlocation Vector)

Nur NHDE, CLTK und FSCT sind wirkliche Daten, alle anderen Werte sind Verweise auf weitere Tabellen.

Die Sektor-Verschränkungs-Tabelle (XLT) dient zur Umrechnung von logischen zu physikalischen Sektornummern einer Spur. Die Länge dieser Tabelle entspricht der Anzahl logischer Sektoren die im DPB (s.u.) definiert sind.

NHDE ist die Nummer des höchsten belegten Directory-Eintrags+1 und dient der Geschwindigkeitserhöhung bei Directory-Zugriffen.
NHDE wird jeweils beim Einloggen eines Laufwerks und beim Löschen oder Neuanlegen eines Eintrages neu berechnet. Die Berechnung der Prüfsumme und das Suchen eines Eintrages, bleibt im BDOS auf den so abgegrenzten Directory-Bereich beschränkt.

CLTK und FSCT dienen dem BDOS der schnelleren Berechnung der anzuwählenden Spur- und Sektornummer bei einem Diskettenzugriff.
Das BDOS berechnet aus Block- und Recordnummer des FCB eine absolute logische Sektornummer der Diskette. Mit der Anzahl von logischen Sektoren pro Spur (siehe DPB) kann daraus die Spur- und Sektornummer durch Division bestimmt werden. Da eine Division jedoch sehr zeitaufwendig ist, wird diese Rechnung im BDOS auf eine wiederholte Subtraktion zurückgeführt.
Von der absoluten Sektornummer wird solange die Anzahl der Sektoren pro Spur abgezogen, bis das Ergebnis negativ wird.
Die Anzahl der durchgeführten Subtraktionen ist dann genau die (logische) Spurnummer.
Durch Mitführen der zuletzt angewählten Spurnummer (CLTK) und der absoluten Recordnummer des ersten Records dieser Spur (FSCT), wird diese Rechnung im BDOS noch beschleunigt.

Das BDOS vergleicht die neue anzuwählende absolute Sektornummer immer mit FSCT. Ist die neue Sektornummer kleiner, so wird solange die Spurnummer CLTK dekrementiert und von FSCT die Anzahl von Sektoren pro Spur abgezogen, bis die Sektornummer wieder größer als FSCT ist.
In einer zweiten Schleife wird danach das Gleiche mit umgekehrten Vorzeichen wiederholt: Es wird CLTK erhöht und zu FSCT die Anzahl der Sektoren pro Spur addiert, bis die anzuwählende absolute Sektornummer wieder kleiner als FSCT ist.

Dieses 'Herantasten' an die Spurnummer beschleunigt die Berechnung der Spurnummer ganz erheblich, da auf eine Division verzichtet werden kann.
Aus der Differenz der absoluten Sektornummer und des neuen FSCT-Wertes berechnet das BDOS dann noch die logische Sektornummer innerhalb der Spur.

DIRBUF ist die Adresse eines 128-Byte Puffers für Directory-Operationen. Alle Laufwerke können den selben DIRBUF benutzen, da das BDOS den Inhalt des DIRBUF bei einem Lauf-

werkswechsel verwirft.

Der Disk Parameter Block (DPB) ist ziemlich komplex und wird im nächsten Abschnitt getrennt besprochen. Mehrere DPH können auf ein und denselben DPB verweisen.

Im CSV sind die Prüfsummen (engl. Checksum) der einzelnen Directory-Records gespeichert.
Pro Directory-Record ist im CSV ein Byte vorhanden, daß die Quersumme aller 128 Bytes des entsprechenden Records enthält. Vor der Änderung eines Directory-Records wird seine Quersumme bestimmt und mit der im CSV gespeicherten Quersumme verglichen. Bei einer Abweichung sperrt das BDOS alle weiteren Schreibzugriffe auf dieses Laufwerk. Näheres dazu wurde bereits im Zusammenhang mit dem Warmstart beschrieben.

Der Allocation Vektor (ALV) bildet die Belegungstabelle (besser: Belegungsvektor) der Diskette.
Aus dem Belegungsvektor kann das BDOS ersehen, welche Blöcke auf der Diskette noch frei und welche belegt sind. Aus diesem Vektor holt auch das STAT-Programm seine Information über den noch verfügbaren Platz auf der Diskette.
Die Bezeichnung 'Vektor' deutet beim ALV auf die bitweise Speicherung hin. Für jeden Block der Diskette ist im ALV ein Bit vorhanden, daß entsprechend auf 0 (Block frei) oder 1 (Block belegt) gesetzt wird.
Die Zuordnung der Blöcke zu den Bits geschieht in absteigender Bitnummernfolge (höchstes Bit eines Bytes zuerst) und aufsteigender Bytefolge (erstes Byte des ALV zuerst).

Die Position eines zu einem Block gehörenden Bits, läßt sich sehr einfach berechnen:
Die Blocknummer geteilt durch 8 ergibt die Position des Bytes innerhalb des ALV; der Divisionsrest die Nummer des Bits innerhalb des ALV, wobei 0 das höchst- und 7 das niederwertigste Bit bezeichnet.

Der Disk Parameter Block

Der Disk Parameter Block (DPB) beinhaltet alle Parameter, die Größe und Aufteilung der Diskette beschreiben, insbesondere die Anzahl der logischen Sektoren pro Spur, die Blockgröße, die Anzahl der Blocks auf der Diskette und die Größe der Directory.

Ein DPB umfasst 15 Bytes in folgender Aufteilung:

SPT	BSH	BLM	EXM	DSM	DRM	AL0	AL1	CKS	OFF
0/1	2	3	4	5/6	7/8	9	A	B/C	D/E
16	8	**8**	**8**	16	16	**8**	8	16	16

(Die erste Zahlenreihe gibt die Position des Parameters innerhalb des DPB an und die zweite, ob es sich um einen 8- oder 16-Bit Wert handelt.)

mit

SPT — Anzahl der logischen Sektoren pro Spur
(engl. Sectors Per Track)

BSH — 2-er Exponent der Blockgröße
(engl. Block SHift factor)

BLM — Anzahl von Records pro Block -1
(engl. Block Length Mask)

EXM — Anzahl der Extends pro Eintrag -1
(engl. Extend Mask)

DSM — Höchste Blocknummer der Diskette
(engl. Data Storage Maximum)

DRM — Höchste Eintragnummer in der Directory
(engl. DiRectory Maximum)

AL0 — Erstes Byte des ALV
(engl. Allocation Vector byte 0)

AL1 — Zweites Byte des ALV
(engl. Allocation Vector byte 1)

CKS — Anzahl der zu prüfenden Directory-Records
(engl. CheckK vector Size)

OFF — Anzahl der reservierten Spuren am Anfang der Diskette
(engl. track OFFset)

Die Verwendung des SPT-Wertes wurde bereits im letzten Abschnitt beschrieben. SPT entspricht auch der Länge der Sektor-Verschränkungs-Tabelle (XLT).

BSH und BLM beinhalten beide die Blocklänge BLS (engl. BLock Size). Durch diese 'doppelte' Angabe der Blocklänge werden bestimmte BDOS-interne Rechenoperationen vereinfacht.

Zwischen der Blocklänge in Bytes und den beiden Werten BSH und BLM gilt folgender Zusammenhang:

BLS	BSH	BLM
1024	**3**	**7**
2048	**4**	15
4096	**5**	31
8192	**6**	63
16384	**7**	127

kurz

BLS = 128 * 2^BSH = 128 * (BLM+1)

In EXM wird die Anzahl von Extends pro Directory-Eintrag definiert.
EXM ist abhängig von der Blockgröße und der Anzahl der Blöcke pro Diskette. Je nachdem, ob weniger als 256 Blöcke (8-Bit Blocknummern) oder mehr als 255 Blöcke (16-Bit Blocknummern) definiert sind, gibt es für EXM jeweils zwei Möglichkeiten:
(Die erste EXM-Spalte gibt die Werte für 8-Bit Blocknummern, die zweite für 16-Bit Blocknummern an.)

BLS	EXM	
1024	0	**-**
2048	**1**	**0**
4096	3	**1**
8192	**7**	**3**
16384	15	**7**

DSM enthält die Anzahl von Blöcken pro Diskette -1 bzw. die höchste Blocknummer der Diskette.
Die Gesamtkapazität der Diskette in Bytes ergibt sich aus dem Produkt von BLS und (DSM+1); für den Allocation Vector müssen (DSM / 8) + 1 Bytes reserviert werden.

Rein theoretisch ist zwar eine Maximalkapazität von einem Gigabyte (65536 Blöcke a 16 kbyte) denkbar, diese ist aber durch die Verwaltung der Recordnummern im 16-Bit Format auf 65536 Records, also 8 Megabyte (65536 * 128 Bytes) beschränkt.

Diskettenparameter

DRM+1 ist die Anzahl der Directory-Einträge der Diskette. Da die Eintragsnummern - genauso wie die Blocknummern - von Null an gezählt werden, enthält DRM den um eins verminderten Wert.

AL0 und AL1 bilden die ersten beiden Bytes des Allocation Vectors und müssen daher, wie der ALV, als Bitvektor gesehen werden.
Für jeden als Directory-Bereich reservierten Block der Diskette, muß das entsprechende Bit in diesem Vektor gesetzt werden. Das höchste Bit von AL0 entspricht dem ersten Block der Diskette (Block Nummer 0) und das letzte Bit in AL1 dem sechzehnten Block (Block Nummer 15). Da AL0 und AL1 nur einen 16-Bit Vektor bilden, ist die Anzahl der maximal möglichen Directory-Blöcke auf 16 beschränkt.

CKS kennzeichnet die Länge des Prüfsummen-Vektors (CSV) und damit die Anzahl der zu prüfenden Directory-Records die sich aus der Beziehung CKS = (DRM + 1) / 4 ergibt.
Falls eine Prüfung der Directory nicht erwünscht oder, wie z.B. bei Festplattenlaufwerken, nicht notwendig ist, kann CKS auch Null sein.

OFF ist die Anzahl der reservierten Spuren auf der Diskette. Dieser Wert ist für das BDOS unerheblich, wird aber vor dem Setzen der physikalischen Spurnummer zur errechneten logischen Spurnummer (CLTK) addiert.

Aufruf von BDOS-Funktionen

Alle BDOS-Funktionen werden über den Sprungbefehl zum BDOS an der Adresse BOOT+0005H erreicht. Dadurch ist, unabhängig von der tatsächlichen Lage des BDOS im Speicher, die Betriebssystem-Schnittstelle für alle CP/M-Rechner identisch.

Ein Programm lädt die gewünschte Funktionsnummer in das C-Register des Prozessors und führt danach ein 'CALL' nach BOOT+0005H aus. 8-Bit Funktionsparameter werden im E-Register, 16-Bit Parameter im Registerpaar DE übergeben. Das BDOS führt daraufhin die gewünschte Funktion aus und gibt einen 8- (im Register A) oder 16-Bit Funktionswert (im Registerpaar HL) zurück.

Das BDOS im CP/M 2.2 bietet insgesamt 39 verschiedene Funktionen zur Zeichen- und Dateibearbeitung, die im folgenden beschrieben werden. Neben der Funktionsnummer und der englischen und deutschen Funktionsbezeichnung, ist auch die Startadresse der jeweiligen Funktion im BDOS-Listing angegeben.

'I:' bezeichnet die notwendige Registerbelegung beim Aufruf des BDOS, 'O:' die vom BDOS zurückgelieferten Registerwerte.

Nummer: **0**

Bezeichnung: System Reset
Warmstart

Adresse: BIOS+0003H

I: C = 00H

O: -

Ausgeführte Funktion:

Direkter Sprung zur Warmstartroutine im BIOS. Der Warmstart lädt den CCP und das BDOS neu in den Speicher und übergibt die Kontrolle an den CCP. Die Funktion 0 entspricht einem Sprung zur Adresse BOOT.

Nummer: **1**

Bezeichnung: Console Input
Consoleneingabe

Adresse: 02C8H

I: C - 01H

O: A - Zeichen von der Console

Ausgeführte Funktion:

Console Input holt das nächste Zeichen von der Console und sendet darstellbare Zeichen über die Consolenausgabe als Echo zurück.
Die Steuerzeichen 'Carriage Return' (ASCII 0DH), 'Line Feed' (ASCII 0AH) und 'Backspace' (ASCII 08H) werden ebenfalls ausgegeben. Das 'Tab'-Zeichen (ASCII 09H) erzeugt auf der Ausgabe soviele Leerzeichen wie zur Erreichung der nächsten, durch 8 teilbaren Spaltenposition (Tabulator-Position) notwendig sind.
Bei Erkennung von CTRL-S (ASCII 13H) wartet Console Input auf ein weiteres Zeichen und hält dadurch die Verarbeitung an.
CTRL-C (ASCII 03H) führt zu einem direkten Sprung nach BOOT, was im allgemeinen einen Warmstart auslöst.
Die Console Input Funktion ist erst beendet, wenn tatsächlich ein Zeichen von der Console erkannt wurde.

Nummer: **2**

Bezeichnung: Console Output
Consolenausgabe

Adresse: 0190H

I: C = 02H
E = ASCII-Wert

O: -

Ausgeführte Funktion:

Der ASCII-Wert vom Register E wird zur Console ausgegeben und die Consoleneingabe überprüft. Vor der eigentlichen Zeichenausgabe wird der Consolenstatus über die BDOS-Funktion 11 überprüft. Alle dort beschriebenen Verhaltensweisen gelten daher auch für die Consolenausgabe. Die Behandlung des 'Tab'-Zeichens ist wie in Funktion 1.

Nummer: 3

Bezeichnung: Reader Input
'Reader'-Eingabe

Adresse: 02CEH

I: C - 03H

O: A - Zeichen vom 'Reader'-Kanal

Ausgeführte Funktion:

Reader Input holt das nächste Zeichen vom 'Reader'-Kanal, stellt es aber nicht auf der Console dar.
Das BDOS führt keine weitere Zeichenbehandlung durch, sondern benutzt direkt die entsprechende BIOS-Funktion.

Nummer: 4

Bezeichnung: Punch Output
'Punch'-Ausgabe

Adresse: BIOS + 0012H

I: C - 04H
E = ASCII-Wert

O: -

Ausgeführte Funktion:

Das BDOS verzweigt direkt zur entsprechenden BIOS-Funktion, die das Zeichen zum 'Reader'-Kanal ausgibt.

Nummer: 5

Bezeichnung: List Output
Druckerausgabe

Adresse: BIOS + 000FH

I: C = 05H
E = ASCII-Wert

O: -

Ausgeführte Funktion:

Das BDOS verzweigt direkt zur entsprechenden BIOS-Funktion, die das Zeichen zum 'List'-Kanal ausgibt. Der 'List'-Kanal ist im CP/M als Druckerausgabe definiert.

Nummer: **6**

Bezeichnung: Direct Console I/O
Direkte Consolenein/ausgabe

Adresse: 02D4H

I: C - 06H
E - 0FFH für Consoleneingabe
E = 0FEH für Consolen-Status prüfen
E - ASCII Zeichen zur Consolenausgabe sonst

O: A - Zeichen von der Console (falls E - 0FFH)
A - Consolen-Status (falls E - 0FEH)
A undefiniert sonst

Ausgeführte Funktion:

Über die direkte Consolenein/ausgabe kann die Console unter Umgehung des BDOS erreicht werden. Alle Sonderbehandlungen von Zeichen, wie sie das BDOS bei den Funktionen 1 und 2 vornimmt, werden unterdrückt. Die direkte Consolenein/ausgabe benutzt direkt die entsprechenden BIOS-Funktionen.

Nummer: **7**

Bezeichnung: Get I/O Byte
I/O Byte holen

Adresse: 02EDH

I: C = 07H

O: A = I/O-Byte von der Adresse BOOT+0004H

Ausgeführte Funktion:

Das BDOS gibt den Inhalt der Speicherstelle BOOT+0004H im A-Register zurück. Über das I/O-Byte kann im BIOS eine weitere Aufschlüsselung der Ein/Ausgabekanäle vorgenommen werden, dies hat jedoch keinen Einfluss auf das BDOS.

Nummer: **8**

Bezeichnung: Set I/O Byte
I/O Byte setzen

Adresse: 02F3H

I: C - 08H
E - I/O-Byte

O: -

Ausgeführte Funktion:

Der im E-Register übergebene Wert wird in der Speicherstelle BOOT+0004H abgelegt. Damit kann ein Programm die Zuordnung der Ein/Ausgabekanäle beeinflussen, falls das I/O-Byte vom BIOS beachtet wird.

Nummer: **9**

Bezeichnung: Print String
Zeichenkette ausgeben

Adresse: 02F8H

I: C - 09H
DE -> Zeichenkette

O: -

Ausgeführte Funktion:

Die einzelnen Zeichen der Zeichenkette werden über die BDOS-Funktion 2 zur Console ausgegeben. Die weitere Zeichenbehandlung ist daher identisch mit der Consolenausgabe. Das Zeichen '$' zeigt das Ende der Zeichenkette an und wird nicht ausgegeben.

Nummer: 10

Bezeichnung: Read Console Buffer
Zeile von der Console einlesen

Adresse: 01E1H

I: C = 0AH
DE -> Zeilenbuffer

O: Zeilenbuffer gefüllt mit den eingegebenen Zeichen

Ausgeführte Funktion:

Alle folgende Zeichen von der Consoleneingabe werden in den Zeilenpuffer übernommen und über die Consolenausgabe dargestellt.
Die Eingabe ist beendet, wenn entweder eines der Zeichen 'Carriage Return' (ASCII 0DH) oder 'Line Feed' (ASCII 0AH) erkannt wird oder das Ende des Buffers erreicht ist.
Die Bufferlänge erhält das BDOS im ersten Byte des Zeilenbuffers mitgeteilt. Nach Beendigung der Funktion steht im zweiten Byte die Anzahl der tatsächlich eingegebenen Zeichen gefolgt von den eigentlichen Zeichen. Die Bufferlänge ist damit immer um zwei Bytes größer als die maximale Zeilenlänge.
Während der Eingabe werden vom BDOS verschiedene Steuerzeichen zur Editierung der Zeile erkannt:

CTRL-C (ASCII 03H)	führt zu einem Warmstart, falls es als erstes Zeichen eingegeben wird.
CTRL-E (ASCII 05H)	beginnt eine neue Zeile auf der Consolenausgabe, ohne jedoch den Zeilenbuffer zu ändern.
CTRL-H (ASCII 08H)	löscht das jeweils letzte Zeichen sowohl im Buffer als auch auf der Consolenausgabe.
CTRL-J (ASCII 0AH)	beendet die Zeileneingabe und sendet ein 'Carriage Return' (ASCII 0DH) zur Consolenausgabe.
CTRL-M (ASCII 0DH)	hat die selbe Wirkung wie CTRL-J
CTRL-P (ASCII 10H)	schaltet die parallele Druckerausgabe zur Consolenausgabe (Protokollfunktion) ein oder aus.
CTRL-R (ASCII 12H)	beginnt eine neue Ausgabezeile und zeigt die bisher eingegebene Bufferzeile nochmals an.
CTRL-U (ASCII 15H)	beginnt eine neue Ausgabezeile und startet die gesamte Zeileneingabe neu. Die bisher ausgegebenen Zeichen werden nicht gelöscht.
CTRL-X (ASCII 18H)	löscht die bisher eingegebenen Zeichen sowohl im Buffer als auch auf der Consolenausgabe. Die Zeileneingabe wird daraufhin neu begonnen.
DEL (ASCII 7FH)	löscht das jeweils letzte Zeichen des Buffers und gibt es nochmals auf der Console aus.

Nummer: 11

Bezeichnung: Get Console Status
Consolenstatus testen

Adresse: 02FEH

I: C = 0BH

O: A = FFH, falls ein Zeichen von der Console ansteht
= 00H sonst

Ausgeführte Funktion:

Es wird geprüft, ob ein Zeichen von der Console zur Eingabe ansteht. Die Consolenstatus-Funktion speichert ein anstehendes Zeichen zwischen und gibt es beim nächsten Consoleneingabe-Aufruf als Zeichen zurück.
Das Zeichen CTRL-S wird von dieser Funktion erkannt, die daraufhin auf ein weiteres Zeichen wartet. CTRL-C bewirkt einen direkten Sprung nach BOOT, was im allgemeinen zu einem Warmstart führt.
Falls ein anstehendes Zeichen nicht 'abgeholt' wird, gibt die Consolenstatus-Funktion immer FFH zurück, ohne den Status der Console wirklich zu testen. In diesem Fall werden die Steuerzeichen CTRL-S und CTRL-C nicht mehr erkannt !

Nummer: 12

Bezeichnung: Return Version Number
CP/M Versionsnummer zurückmelden

Adresse: 0C7EH

I: C = 0CH

O: HL = 0022H (bei CP/M 2.2)

Ausgeführte Funktion:

Über die BDOS-Funktion 12 können Programme auf eine bestimmte CP/M Version angepasst werden. Im H-Register wird für CP/M der Wert 00H, für MP/M (multi-tasking CP/M) der Wert 01H zurückgegeben. Das L-Register bezeichnet dann die genaue Versionsnummer. L = 00H bezeichnet alle Versionen vor CP/M 2.0, die Werte 20H, 21H, 22H usw. die entsprechenden CP/M Versionen 2.0, 2.1, 2.2 usw.

Nummer: 13

Bezeichnung: Reset Disk System
BDOS in den Ausgangszustand bringen

Adresse: 0C83H

I: C - 0DH

O: A - FFH, falls auf dem Laufwerk A:, Userbereich 0 ein Filenamen mit '$' beginnt.
A - 00H sonst

Ausgeführte Funktion:

Diese Funktion wird dazu benutzt, das BDOS in einen kontrollierten Ausgangszustand zu bringen. Die Schreiberlaubnis wird für alle Laufwerke freigegeben und das Laufwerk A: angewählt. Die DMA-Adresse wird auf den Standardwert BOOT+0080H gesetzt. Die Anwahl von Laufwerk A: wird im Zusammenhang mit dem zurückgegebenen Parameter vom CCP benutzt. Falls auf dem Laufwerk A:, Userbereich 0 ein Filenamen mit '$' beginnt, so versucht der CCP das Submit-File '$$$.SUB' zu öffnen und den nächsten Befehl aus diesem File zu lesen.

Nummer: 14

Bezeichnung: Select Disk
Laufwerk anwählen

Adresse: 0C45H

I: C = 0EH
E = Nummer des anzuwählenden Laufwerks
= 00H für Laufwerk A:
...
= 0FH für Laufwerk P:

O:

Ausgeführte Funktion:

Das gewählte Laufwerk wird als Bezugslaufwerk für alle weiteren Dateioperationen festgelegt. Falls das Laufwerk seit dem letzten Warmstart noch nicht angewählt war, liest das BDOS die Directory und initialisiert die Belegungs- und Prüfsummentabelle.
Das BDOS prüft vor Aufruf der BIOS-Funktion zur Laufwerksanwahl, die im Register E übergebene Laufwerksnummer nicht auf ihren Wert.
Das BIOS gibt dem BDOS die Adresse des zum Laufwerk gehörenden DPH (falls unter der Nummer ein Laufwerk existiert) oder den Wert 0000h zurück (falls kein Laufwerk existiert bzw. der

Wert > 0FH war).
Im Fehlerfall meldet das BDOS einen 'Select Err' und bricht die Funktion über einen Warmstart ab.

Nummer: 15

Bezeichnung: Open File
File öffnen

Adresse: 0C9CH

I: C - 0FH
DE -> FCB

O: A - Fehlercode
- 0FFH, falls das File nicht existiert
- Directory-Code des geöffneten Eintrags sonst

Ausgeführte Funktion:

Das in der ersten FCB-Position angegebene Laufwerk (0 - aktuelles Laufwerk, 1 - Laufwerk A:,..., 16 - Laufwerk P:) wird angewählt und ein auf den FCB passender Eintrag gesucht. Die Suche nach dem passenden Eintrags und auch die Berechnung des Directory-Code geschieht über die BDOS-Funktion 17 'Search for First'.
Der gefundene Directory-Eintrag wird komplett in den FCB übertragen, der damit nach dem Öffnen alle Directory-Informationen enthält.
Durch Angabe einer Extendnummer in der entprechenden FCB-Position, kann direkt der Eintrag geöffnet werden, der den gewünschten Extend abdeckt. Die direkte Anwahl einer Extendgruppen ist nicht möglich, es wird immer ein Extend in der Extendgruppe 0 geöffnet.

Nummer: 16

Bezeichnung: Close File
File schliessen

Adresse: 0CA5H

I: **C** = 10H
DE -> FCB

O: **A** = Fehlercode
= FFH, falls das File nicht existiert oder der FCB fehlerhafte Daten enthält
= Directory-Code des geschlossenen Eintrags sonst

Ausgeführte Funktion:

Über die BDOS-Funktion 17 ('Search for First') wird ein auf den FCB passender Eintrag gesucht und die Daten des FCB in diesen Eintrag übernommen. Falls das über die FCB-Position 0 angewählte Laufwerk schreibgeschütz ist, wird die Funktion mit einem 'R/O'-Fehler abgebrochen.
Ist das File seit dem Öffnen unverändert ('NA'-Flag gesetzt), so wird die 'Close File'-Funktion sofort beendet und der Wert 00H zurükgegeben.
Vor dem Zurückschreiben des Eintrags, wird die Blocktabelle des gefundenen Eintrags mit der Blocktabelle des FCB verglichen und die Blocktabellen einander angepasst. Differenzen der Blocknummern dürfen nur bei jeweils unbelegten Tabellen-Einträgen auftreten.
Da die ersten 32 Bytes des FCB in den Eintrag kopiert werden, ist die neue Länge des Files durch die Werte EX, EG und RC im FCB gegeben. Vor dem Schliessen eines Files muß die Recordposition daher immer auf den letzten Record des Files gebracht werden.

Nummer: **17**

Bezeichnung: Search for First
Ersten passenden Eintrag suchen

Adresse: 0CABH

I: C = 11H
DE -> FCB

O: A = Fehlercode
= FFH, falls das File nicht existiert
= Directory-Code des gefundenen Eintrags
sonst

Ausgeführte Funktion:

Das in der ersten FCB-Position angegebene Laufwerk wird angewählt und die Directory nach dem ersten, auf den FCB passenden Eintrag, durchsucht.
Es werden dabei nur die Directory-Einträge beachtet, die im aktuellen Userbereich liegen und in den Positionen 1-14 mit de FCB übereinstimmen.
Der Wert 3FH ('?') in einer der FCB-Positionen 1 bis 14 wird als 'Joker' gewertet und bei der Suche nach einem passenden Eintrag nicht beachtet.
Bei einem 'Joker' in der FCB-Position 0, gibt die Funktion den ersten Directory-Eintrag, ohne Beachtung seiner Daten, zurück. Diese Funktion ist zwar im allgemeinen nicht sinnvoll, kann aber im Zusammenhang mit der Funktion 'Search for Next' (BDOS-Funktion 18) zur Erreichung aller Directory-Einträge benutzt werden.

Aus Gründen der Kompatibilität mit der alten CP/M Version 1.4, kopiert diese Funktion den Directory-Record, der den gefundenen Eintrag enthält, an die DMA-Adresse. Dadurch können aber auch Anwenderprogramme die Directory-Informationen auswerten und für eigene Zwecke nutzen.

Nummer: 18

Bezeichnung: Search for Next
Nächsten passenden Eintrag suchen

Adresse: 0CC8H

I: C - 12H

O: A - Fehlercode
- FFH, falls kein weiterer Eintrag gefunden werden konnte
- Directory-Code des nächsten gefundenen Eintrags sonst

Ausgeführte Funktion:

Ausgehend vom letzten 'Search for First', wird der nächste passende Eintrag gesucht. Die Angabe einer FCB-Adresse als Funktionsparameter ist nicht nötig, da der FCB vom letzten 'Search for First'-Aufruf benutzt wird. Der Directory-Record, der den gefundenen Eintrag enthält, wird zur aktuellen DMA-Adresse kopiert.
Die Bezeichnungen 'erster' und 'nächster' in den BDOS-Funktionen 17 und 18, beziehen sich immer auf die Nummer des Eintrages in der Directory.

Nummer: 19

Bezeichnung: Delete File
File löschen

Adresse: 0CD7H

I: C = 13H
DE -> FCB

O: A = Fehlercode
= 00H, kein Fehler
= FFH, falls kein passender Eintrag gefunden wurde

Ausgeführte Funktion:

Beim Löschen eines Files, werden alle auf den FCB passenden Einträge aus der Directory gelöscht und die in der Blockta-

belle des Eintrages verzeichneten Blöcke durch Änderung der Belegungstabelle freigegeben.
Das Löschen eines Eintrages aus der Directory, geschieht durch Ersetzen der ersten Eintragsposition (Usernummer) mit dem Wert E5H. Alle weiteren Bytes des Eintrages bleiben unverändert.
Dies hat den Vorteil, daß durch Wiedereinsetzen der Usernummer ein Eintrag wieder aktiviert werden kann. Beim Reaktivieren ist jedoch zu beachten, daß manche der vormals belegten Blöcke bereits wieder neu vergeben sein können.

Nummer: 20

Bezeichnung: Read Sequential
Sequentielles Lesen

Adresse: 0CE0H

I: **C** - 14H
DE -> FCB

O: **A** - Fehlercode
- 00H, kein Fehler
- 01H, Ende des Files erreicht
- FFH, falls zu dem File kein Directory-Eintrag existiert

Ausgeführte Funktion:

Der durch die FCB-Werte CR, EX und EG definierte Record des Files wird gelesen und die Recordposition im FCB um 1 erhöht. Beim Erreichen einer Extendgrenze wird der aktuelle Extend automatisch geschlossen und der folgende Extend eröffnet. Falls der nächste Extend nicht mehr vom FCB abgedeckt ist, öffnet das BDOS auch einen neuen Eintrag. Der Fehlercode FFH wird zurückgegeben, wenn in diesem Fall der alte Eintrag nicht mehr gefunden werden konnte.
Vor dem Lesen eines Files muß über die BDOS-Funktion 26 'Set DMA Address' ein 128-Byte Speicherbereich zur Aufnahme des Records definiert werden. Das BDOS erhöht nach einem sequentiellen Lesezugriff zwar die Recordposition im FCB, nicht aber die DMA-Adresse. Beim Lesen mehrerer Records in den Speicher ist das Anwenderprogramm für die Anpassung der DMA-Adresse zuständig.

Nummer: 21

Bezeichnung: Write Sequential
Sequentielles Schreiben

Adresse: 0CE6H

I: C - 15H
DE -> FCB

O: A - Fehlercode
- 00H, kein Fehler
- 01H, Fehler beim Diskzugriff
oder falsche Recordnummer
- 02H, Diskette voll

Ausgeführte Funktion:

Aus dem zuletzt über 'Set DMA-Address' (BDOS-Funktion 26) definierten Speicherbereich, werden 128 Bytes in das angegebene File geschrieben. Die exakte Recordposition ist durch die Werte EG, EX und CR im FCB bestimmt.
Nach Abschluss der Fuktion wird die Recordposition um eins erhöht und, bei Erreichen der Extendgrenze, ein neuer Extend oder sogar ein neuer Eintrag eröffnet.
Dieses 'Vorgreifen' auf den nächsten Record führt allerdings dazu, daß evtl. ein Block oder Eintrag zuviel belegt wird, da der neue Record erst beim folgenden Schreibzugriff mit Daten gefüllt wird.
Abhilfe bietet in diesem Fall nur der direkte Zugriff (Random Access), der diesen 'Vorgriff' unterdrückt.

Nummer: 22

Bezeichnung: Make File
Neues File erzeugen

Adresse: 0CECH

I: C = 16H
DE -> FCB

O: A = Fehlercode
= FFH, falls die Directory voll ist
= Directory-Code des neuen Eintrages sonst

Ausgeführte Funktion:

'Make File' wird dazu benutzt, ein neues File zu erzeugen und zu öffnen, zu dem noch kein Directory-Eintrag vorliegt.
Maßgebend für den neuen Eintrag sind nur die ersten 12 Bytes des FCB, da alle weiteren Bytes automatisch auf 0 initialisiert werden.

Die 'Make File'-Funktion sucht den ersten freien Eintrag in der Directory, kopiert die 32 Bytes des FCB in den Eintrag und schreibt ihn wieder zurück in die Directory.
Falls im selben Userbereich bereits ein weiterer Eintrag mit dem gleichen Filenamen und Filetyp existiert, kann das BDOS zwischen beiden Einträgen nicht unterscheiden. Vor dem Erzeugen eines neuen Files muß daher, z.B. durch vorherigen Aufruf der 'Delete File'-Funktion, sichergestellt sein, daß der Filenamen und Filetyp nur einmal existiert.

Nummer: **23**

Bezeichnung: Rename File
Filenamen ändern

Adresse: 0CF5H

I: C = 17H
DE -> FCB (Alter und neuer Filenamen)

O: A = Fehlercode
= 00H, kein Fehler
= FFH, falls kein Eintrag mit dem alten Filenamen gefunden wurde

Ausgeführte Funktion:

Die ersten 12 Bytes des über DE adressierten FCB geben den alten Filenamen und Filetyp an. Ab Position 17 muß der neue Filenamen und ab Position 25 der neue Filetyp folgen. Nicht benutzte Stellen im Filenamen oder Filetyp müssen Leerzeichen enthalten.
Das in der ersten FCB-Position angegebene Laufwerk wird selektiert und alle Einträge, die in Usernummer, Filename und Filetyp mit dem FCB übereinstimmen, mit dem neuen Filenamen zurückgeschrieben.

Nummer: **24**

Bezeichnung: Return Login Vector
Login-Vektor zurückgeben

Adresse: 0CFEH

I: C = 18H

O: HL = Login-Vektor

Ausgeführte Funktion:

Der Login-Vektor ist ein 16-Bit Wert, dessen niederwertigstes Bit dem Laufwerk A: und höchstwertigstes Bit dem Laufwerk P:

entspricht. Eine '1' an einer der Bitpositionen zeigt an, daß das entsprechenden Laufwerk seit dem letzten Warmstart schonmal angewählt wurde. Aufgrund der Information des Login-Vektors, entscheidet das BDOS bei einer Laufwerksanwahl, ob die Belegungs- und Prüfsummentabelle neu erstellt werden muß.

Nummer: **25**

Bezeichnung: Return Current Disk
Aktuelle Laufwerksnummer zurückgeben

Adresse: 0D04H

I: C - 19H

O: A - Laufwerksnummer
- 0 für Laufwerk A:
...
- 15 für Laufwerk P:

Ausgeführte Funktion:

Die Nummer des zuletzt über die BDOS-Funktion 14 'Select Disk' angewählten Laufwerks, wird im A-Register zurückgegeben.

Nummer: 26

Bezeichnung: Set DMA Address
DMA-Adresse festlegen

Adresse: 0D0AH

I: C = 1AH
DE = DMA-Adresse

O: -

Ausgeführte Funktion:

Die im Registerpaar DE angegebene Adresse wird für alle weiteren Dateioperation als Anfangsadresse eines 128-Byte Buffers benutzt. Nach einem Warmstart oder der BDOS-Funktion 13 'Reset Disk System' ist die DMA-Adresse auf BOOT+0080H festgelegt.

Nummer: 27

Bezeichnung: Get Allocation Address
Adresse des Belegungsvektors (ALV) holen

Adresse: 0D11H

I: C - 1BH

O: HL -> ALV des aktuellen Laufwerks

Ausgeführte Funktion:

Die bei der Laufwerksanwahl im DPH angegebene Adresse des Belegungsvektors (ALV), wird dem aufrufenden Programm mitgeteilt. Aus dem ALV berechnet z.B. das STAT-Programm den verbleibenden Speicherplatz auf einer Diskette.

Nummer: 28

Bezeichnung: Write Protect Disk
Laufwerk schreibschützen

Adresse: 052CH

I: C - 1CH

O: -

Ausgeführte Funktion:

Alle weiteren Schreibzugriff auf das aktuelle Laufwerk werden unterbunden und führen zu der Meldung 'BDOS ERR on d: R/O', wobei d: das Laufwerk bezeichnet. Diese Funktion wird auch BDOS-intern nach dem Erkennen einer Prüfsummendifferenz benutzt.

Nummer: 29

Bezeichnung: Get Read/Only Vector
R/O-Vektor holen

Adresse: 0D17H

I: **C** = 1DH

O: HL = R/O-Vektor

Ausgeführte Funktion:

Der R/O-Vektor ist ein 16-Bit Wert und hat den selben Aufbau wie der Login-Vektor (siehe BDOS-Funktion 24, 'Return Login

Vetor'). Ein gesetztes Bit in diesem Vektor, weist das entsprechende Laufwerk als schreibgeschützt aus.

Nummer: **30**

Bezeichnung: Set File Attributes
File Attribute setzen

Adresse: 0D1DH

I: C = 1EH
DE -> FCB

O: A = Fehlercode
= 00H, kein Fehler
= FFH, falls das File nicht existiert

Ausgeführte Funktion:

Die ersten 12 Bytes des FCB werden in alle passenden Directory-Einträge kopiert und die Einträge wieder zurückgeschrieben. Damit können die Fileattribute (höchste Bits im Filetyp) eines Files geändert werden.

Nummer: **31**

Bezeichnung: Get Disk Parameter Address
Adresse des DPB holen

Adresse: 0D26H

I: C = 1FH

O: **HL** -> DPB des aktuellen Laufwerks

Ausgeführte Funktion:

Die bei der Laufwerksanwahl im DPH angegebene Adresse des Disk Parameter Blocks (DPB) wird dem aufrufenden Programm mitgeteilt.
Aus dem DPB können alle laufwerksspezifischen Parameter entnommen werden. Aus der dort angegebenen Blockgröße und der Anzahl der Blöcke berechnet z.b. das STAT-Programm die Diskettenkapazität in Bytes. Zusammen mit dem Belegungsvektor (ALV) kann so auch der verbleibende Platz in Bytes bestimmt werden.
Spezielle Programme können den DPB auch ändern und so eine dynamische Anpassung an verschiedene Diskettenformate erreichen.

Nummer: 32

Bezeichnung: Set/Get User Code
Usernummer holen/setzen

Adresse: 0D2DH

I: C - 20H
E - FFH (Holen der Usernummer)
E - Usernummer (Setzen der Usernummer)

O: A - aktuelle Usernummer (falls E - FFH)
= 00H sonst

Ausgeführte Funktion:

Abhängig vom im Register E übergebenen Wert, wird entweder ein neuer Userbereich gewählt oder die Nummer des aktuellen Bereichs zurückgegeben.
Beim Setzen der Usernummer, wird der Wert im E-Register modulo 32 genommen und somit der Wertebereich auf 0 bis 31 begrenzt.

Nummer: **33**

Bezeichnung: Read Random
Direkter Lesezugriff

Adresse: 0D41H

I: C = 21H
DE -> FCB mit Random Record Nummer (RRN)

O: A = Fehlercode
= 0, kein Fehler
= 1, der Record enthält noch keine Daten
= 3, der zuletzt angewählte Extend kann nicht geschlossen werden
= 4, der Record liegt 'hinter' dem Fileende
= 6, die gewählte Recordnummer ist größer als 65535

Ausgeführte Funktion:

Durch den direkten Lesezugriff kann ein beliebiger Record eines Files direkt gelesen werden.
Die gewünschte Recordnummer wird als 16-Bit Wert in zusätzlichen FCB-Positionen R0 (Position 33, niederwertiges Byte) und R1 (Positionen 34, höherwertiges Byte) angegeben.
Ein drittes Byte an der Position 35 (R2) ist zwar zur Erweiterung der Recordnummer auf 24-Bit definiert, im CP/M 2.2 jedoch nicht benutzt. Enthält dieses Byte jedoch einen von Null verschiedenen Wert, so wird der Fehlercode 6 'Seek past

physical end of disk' zurückgegeben.
Die angegebene Recordnummer wird in die entsprechenden Werte Extendgruppe (EG), Extend (EX) und Recordnummer im Extend (CR) umgerechnet und ein sequentieller Lesezugriff ausgeführt. Nach Beendigung der 'Read Random'-Funktion sind die FCB-Positionen EG, EX und CR daher immer auf den folgenden Record aktualisiert.
Die angegebene Random Record Nummer wird vom BDOS nicht beeinflusst, wodurch ein folgender Zugriff wieder den selben Record behandelt.
Vor dem Lesezugriff wird ein evtl. notwendiger neuer Extend automatisch geöffnet und der alte Extend geschlossen. Der Fehlercode 1 tritt dann auf, wenn der neue Extend zwar existiert, die neue Recordposition aber über dem höchsten Record dieses Extends liegt.
Die Fehlercode 2 und 5 werden beim direkten Lesezugriff nicht zurückgegeben.

Nummer: 34

Bezeichnung: Write Random
Direkter Schreibzugriff

Adresse: 0D47H

I: C = 22H
DE -> FCB mit Random Record Nummer (RRN)

O: A = Fehlercode
= 0, kein Fehler
= 2, Diskette voll
= 3, der zuletzt angewählte Extend kann nicht geschlossen werden
= 5, ein neuer Eintrag konnte nicht geöffnet werden
= 6, die gewählte Recordnummer ist größer als 65535

Ausgeführte Funktion:

Für die 'Write Random'-Funktion gilt sinngemäß das selbe wie für die 'Read Random'-Funktion. Mit dem direkten Schreibzugriff kann ein File jedoch auch vergrößert werden, weshalb die Fehlercodes 1 und 4 nicht zurückgegeben werden.
Falls zur Erweiterung des Files ein neuer Eintrag erforderlich und die Directory voll ist, wird der Fehlercode 5 zurückgegeben.

Nummer: **35**

Bezeichnung: Compute File Size
Filelänge bestimmen

Adresse: 0D4DH

I: C = 23H
DE -> FCB

O: R0, R1 und R2 im FCB gesetzt

Ausgeführte Funktion:

Diese Funktion durchsucht alle passenden Einträge und berechnet die Recordnummer des letzten Records im jeweiligen Eintrag. Die höchste, so gefundene Recordnummer bestimmt die Filelänge und wird als Random Record Nummer in die Positionen R0, R1 und R2 des FCB eingetragen.
Durch einen nachfolgenden direkten Lesezugriff können Extendgruppe, Extend und Recornummer im FCB gesetzt und, über weitere sequentielle Schreibzugriffe, Records an das File angehängt werden.

Nummer: 36

Bezeichnung: Set Random Record
Random Record Nummer bestimmen

Adresse: 0C0EH

I: C = 24H
DE -> FCB

O: R0, R1 und R2 im FCB gesetzt

Ausgeführte Funktion:

Aus der sequentiellen Recordposition im FCB (Extendgruppe, Extend und Record) wird die Random Record Nummer berechnet und im FCB eingesetzt. Dadurch kann ein 'Umschalten' vom sequentiellen in den direkten Zugriff erreicht werden.
Das 'Zurückschalten' erfolgt bei jedem weiteren direkten Zugriff, der aus der Random Record Nummer wieder die entsprechende Extendgruppennummer, Extendnummer und Recordnummer berechnet und in den FCB einsetzt.

Nummer: 37

Bezeichnung: Reset Drive
Laufwerk zurücksetzen

Adresse: 0D53H

I: C - 25H
DE - Disk Vektor

O: A - 00H

Ausgeführte Funktion:

Durch das Zurücksetzen eines Laufwerkes, wird das entsprechende Bit aus dem Login- und R/O-Vektor gelöscht und beim nächsten Zugriff auf dieses Laufwerk, der Belegungsvektor und die Prüfsummentabelle neu errechnet.
Der im Registerpaar DE übergebene Vektor hat den selben Aufbau wie der Login- und der R/O-Vektor, bezeichnet aber die zurückzusetzenden Laufwerke.
Sinn dieser Funktion ist es, einen vom Anwenderprogramm kontrollierten Diskettenwechsel zu ermöglichen.

Nummer: 38

Bezeichnung: -

Adresse: -

I: C = 26H

O: A = 00H

Ausgeführte Funktion:

keine Funktion

Nummer: 39

Bezeichnung: -

Adresse: -

I: C = 27H

O: A = 00H

Ausgeführte Funktion:

keine Funktion

Nummer: **40**

Bezeichnung: Write Random with Zero Fill
Direkter Schreibzugriff mit Initialisierung

Adresse: 0D9BH

I: C - 28H
DE -> FCB mit Random Record Nummer

O: A - Fehlercode
- = 0, kein Fehler
- - 2, Diskette voll
- - 3, der zuletzt angewählte Extend kann nicht geschlossen werden
- - 5, ein neuer Eintrag konnte nicht geöffnet werden
- - 6, die gewählte Recordnummer ist größer als 65535

Ausgeführte Funktion:

Diese Funktion entspricht der 'Write Random'-Funktion, mit der Ausnahme, daß neu belegte Blöcke initialisiert werden. Beim Erreichen eines neuen, bisher nicht belegten Blocks, wird der gesamte Block mit 00H gefüllt. Dadurch kann ein Programm bei einem späteren Lesezugriff, daß Ende des Files genau bestimmen.

BIOS-Funktionen

Alle BIOS-Funktionen werden über eine Sprungtabelle erreicht, die sich direkt an das BDOS anschliesst. Das BDOS benutzt die einzelnen Funktionen durch einen 'CALL' zu der entsprechenden Tabellenposition.

Im folgenden soll der Aufbau der Sprungtabelle und die Parameterübergabe zu den BIOS-Funktionen beschrieben werden. Da die Realisierung eines BIOS jedoch stark vom jeweiligen Computersystem abhängt, wird auf die Funktionen im einzelnen nicht weiter eingegangen.

Das BIOS bekommt Funktionsparameter grundsätzlich im Register C bzw. bei 16-Bit Werten im Registerpaar BC übergeben und teilt dem BDOS das Funktionsresultat im Register A mit.

BIOS-Sprungtabelle

Die Sprungtabelle am Anfang des BIOS bzw. am Ende des BDOS muß folgenden Aufbau haben:

```
Adresse        Sprungbefehl              Funktion
-------------------------------------------------------------
BIOS+00H       JP    CBOOT               ;Kaltstart, System-
                                         ;initialisierung
BIOS+03H       JP    WBOOT               ;Warmstart, CCP und
                                         ;BDOS nachladen
BIOS+06H       JP    CONST               ;Status der Consolen-
                                         ;eingabe testen
BIOS+09H       JP    CONIN               ;Consoleneingabe,

BIOS+0CH       JP    CONOUT              ;Consolenausgabe

BIOS+0FH       JP    LIST                ;'List'-ausgabe
                                         ;(Drucker)
BIOS+12H       JP    PUNCH               ;'Punch'-Ausgabe

BIOS+15H       JP    READER              ;'Reader'-Ausgabe

BIOS+18H       JP    HOME                ;Spur 0 anwählen

BIOS+1BH       JP    SELDSK              ;Disk anwählen

BIOS+1EH       JP    SETTRK              ;Spurnummer setzen

BIOS+21H       JP    SETSEC              ;Logische Sektornummer
                                         ;setzen
BIOS+24H       JP    SETDMA              ;DMA-Adresse fest-
                                         ;legen
```

```
BIOS+27H        JP    READ          ;Logischen Sektor
                                    ;lesen
BIOS+2AH        JP    WRITE         ;Logischen Sektor
                                    ;schreiben
BIOS+2DH        JP    LISTST        ;Status der Drucker-
                                    ;ausgabe testen
BIOS+30H        JP    SECTRAN       ;Sektornummer aus der
                                    ;Verschränkungstabelle
                                    ;holen
```


Name: CBOOT, Cold BOOT
Kaltstart ausführen

I: -

O: Sprung zum CCP oder Übergang in die Warmstart-Funktion

Funktion:

Der Kaltstart wird nur nach dem erstmaligen Laden des Betriebsystems benötigt.
Der Aufruf an den Kaltstart erfolgt meist von einem speziellen Ladeprogramm, das nach dem Einschalten des Rechners, das BIOS von der Systemdiskette geladen hat.
Aufgabe des Kaltstart ist es, die einzelnen Systemkomponenten zu initialisieren und eine Meldung über den erfolgten Systemstart auf der Console auszugeben.
Falls der CCP und das BDOS bereits im Speicher stehen, kann nach dem Kaltstart sofort der CCP aktiviert werden. In diesem Fall müssen aber die beiden Sprungbefehle an den Adressen BOOT (nach BIOS+0003h) und BOOT+0005h (nach FBASE+0006h) eingesetzt sein. Normalerweise wird nach einem Kaltstart ein Warmstart ausgeführt, der den CCP und das BDOS von der Diskette lädt und die Sprungbefehle einsetzt. Dann braucht nur die zuerst anzuwählende Disk- und Usernummer in die Speicheradresse BOOT+0004h eingetragen zu werden.

Name: WBOOT, Warm BOOT
Warmstart ausführen

I: -

O: Sprung zum CCP

Funktion:

Ein Warmstart wird ausgeführt, wenn ein Programm durch einen Sprung nach BOOT beendet wird oder das Zeichen CTRL-C von der Console erkannt wurde.

Der Warmstart muß die beiden Betriebssystem-Teile CCP und BDOS von der Diskette nachladen und die Sprungbefehle an den Adressen BOOT und BOOT+0005h neu setzen.
Vor dem Wiedereintritt zum CCP, sollte das C-Register mit dem Wert der Speicherstelle BOOT+0004H geladen werden. Dadurch ist sichergestellt, daß nach einem Warmstart der CCP wieder das selbe Laufwerk und den selben Userbereich wie vorher anwählt.

Name: CONST, CONsole STatus
Consolenstatus testen

I: -

O: A = Consolenstatus
- FFH, falls ein Zeichen von der Console ansteht
- 00H, sonst

Funktion:

Der Consolenstatus zeigt an, ob ein Zeichen von der Console ansteht, daß über die Consoleneingabe geholt werden sollte. Diese BIOS-Funktion wird vom BDOS bei jeder Zeichenausgabe zur Erkennung der Steuerzeichen CTRL-S und CTRL-C aufgerufen.

Name: CONIN, CONsole INput
Consolenzeichen holen

I: -

O: A = Zeichen von der Console

Funktion:

Diese Routine holt ein Zeichen von der Console (Tastatur) und gibt es im A-Register zurück. Falls kein Zeichen von der Console ansteht, muß auf ein Zeichen gewartet werden.

Name: CONOUT, CONsole OUTput
Zeichen zur Console ausgeben

I: C = Zeichen im ASCII-Code

O: -

Funktion:

Der im Register C befindliche Wert wird zur Console (Bildschirm) ausgegeben.

Name: LIST, LIST output
Zeichen zum 'List'-Kanal (Drucker) ausgeben

I: C - Zeichen im ASCII-Code

O: -

Funktion:

Wie Consolenausgabe, aber zum 'List'-Kanal.

Name: PUNCH, PUNCH output
Zeichen zum 'Punch'-Kanal ausgeben

I: C - Zeichen im ASCII-Code

O: -

Funktion:

Wie Consolenausgabe, aber zum 'Punch'-Kanal

Name: READER, READER input
Zeichen vom 'Reader'-Kanal holen

I: -

O: A = Zeichen vom 'Reader'-Kanal

Funktion:

Wie Consoleneingabe, aber vom 'Reader'-Kanal

Name: HOME, HOME disk
Spur 0 anwählen

I: -

O: -

Funktion:

Diese Funktion war bei älteren Laufwerken zur exakten Positionierung des Schreib/Lesekopfes gedacht.
Da das BDOS vor jedem Diskzugriff die Spurnummer über SETTRK anwählt, ist HOME bei neueren Laufwerken überflüssig.

Name: SELDSK, SELect DiSK
Laufwerk anwählen

I: C = Laufwerksnummer
= 0 für Laufwerk A:
...
= 15 für Laufwerk P:
E,0 = Login-Bit
= 0, falls das Laufwerk zum ersten Mal angewählt wird
= 1, falls das Laufwerk seit dem letzten Warmstart schonmal angewählt wurde

O: HL -> DPH des Laufwerks
HL = 0, falls das Laufwerk nicht existiert

Funktion:

Das BIOS muß die im C-Register übergebene Laufwerksnummer überprüfen und, falls ein Laufwerk mit dieser Nummer existiert, in HL die Adresse des zugehörigen DPH zurückgeben.
Im CP/M ist nicht garantiert, daß nach einem SELDSK-Aufruf auch tatsächlich auf dieses Laufwerk zugegriffen wird. Vielmehr hat der SELDSK-Aufruf nur eine 'Anmeldefunktion', damit sich das BDOS auf das Laufwerk einstellen kann.
Das BIOS muß die Laufwerksnummer aber intern speichern, da sich nachfolgende Diskzugriffe immer auf das zuletzt selektierte Laufwerk beziehen.

Name: SETTRK, SET TRacK
Spurnummer setzen

I: BC = Spurnummer

O: -

Funktion:

Der nächste Diskzugriff bezieht sich auf die im Registerpaar BC übergebene Spur. Die so gesetzte Spurnummer, errechnet sich immer aus der BDOS-internen (logischen) Spurnummer plus dem OFF-Wert im DPB. Wie auch beim SELDSK-Aufruf ist ein tatsächlicher Diskzugriff nicht garantiert.

Name: SETSEC, SET SECtor
Logische Sektornummer setzen

I: BC - Logische Sektornummer

O: -

Funktion:

Der nächste Diskzugriff bezieht sich auf den im Registerpaar BC übergebenen Sektor. Die so gesetzte Sektornummer ist immer das Ergebnis der SECTRAN-Funktion (s.u.). Auch hier ist ein tatsächlicher Diskzugriff auf diesen Sektor nicht garantiert.

Name: SETDMA, SET DMA-address
DMA-Adresse festlegen

I: BC - DMA-Adresse

O:

Funktion:

Alle nachfolgenden Diskzugriffe müssen die DMA-Adresse als Quell- (bei Schreibzugriffen) bzw. Zieladresse (bei Lesezugriffen) benutzen.
Die DMA-Adresse zeigt immer auf einen 128-Byte großen Buffer, weshalb Diskzugriffe immer in Recordgröße erfolgen.

Name: READ, READ sector
Logischen Sektor lesen

I: -

O: A = Fehlercode
= 0, kein Fehler
= 1, sonst

Funktion:

Die READ-Funktion liest einen (logischen) Sektor von der Diskette in den DMA-Buffer. Die Disknummer, Spurnummer und Sektornummer sind jeweils durch die letzten SELDSK-, SETTRK- und SETSEC-Aufrufe festgelegt.
Bei physikalischen Sektorlängen vom mehr als 128 Bytes, muß das BIOS einen Sektorbuffer entsprechender Größe selbst bereitstellen und aus diesem Buffer, 128 Bytes zum zuletzt definierten DMA-Buffer kopieren.
Falls ein Lesefehler auftritt, sollte das BIOS den Diskzugriff ein paar Mal wiederholen und, falls der Fehler bestehen bleibt, den Fehlercode 1 im A-Register zurückgeben.

Name: WRITE, WRITE sector
Logischen Sektor schreiben

I: C = Record-Flag
- 0, bei einem normalen Schreibzugriff
- 1, falls der logische Sektor ein Directory-Record ist
- 2, falls der logische Sektor der erste Sektor eines neuen Blocks ist

O: A = Fehlercode
- 0, kein Fehler
- 1, sonst

Funktion:

Die WRITE-Funktion schreibt einen (logischen) Sektor vom DMA-Buffer auf die Diskette. Die Disknummer, Spurnummer und Sektornummer sind jeweils durch die letzten SELDSK-, SETTRK- und SETSEC-Aufrufe festgelegt.
Bei physikalischen Sektorlängen von mehr als 128 Bytes, kann das Record-Flag zur Realisierung eines 'Blocking'-Algorithmus verwendet werden. Bei einem normalen Schreibzugriff reicht es, den logischen Sektor nur in den BIOS-internen Sektorbuffer zu übernehmen. Dies hat den Vorteil, daß nachfolgende Schreibzugriffe auf den selben physikalischen Sektor, keinen Diskettenzugriff verlangen. Erst wenn der neue logische Sektor in einem anderen physikalsichen Sektor liegt, muß der Sektorbuffer auf die Diskette geschrieben werden.
Directory-Schreibzugriffe sollten immer direkt auf die Diskette geleitet werden.

Name: LISTST, LIST output STatus
Status der Druckerausgabe testen

I: -

O: A = Druckerstatus
- = 00H, Drucker nicht bereit
- = FFH, Drucker frei

Funktion:

LISTST wird vom BDOS nicht benötigt, sondern dient dem CP/M Hilfsprogramm 'DESPOOL' den Status der Druckerausgabe zu bestimmen.

Name:	SECTRAN, SECtor TRANslation Sektornummer aus der Sektor-Verschränkungs-Tabelle holen
I:	BC - Logische Sektornummer DE -> Sektor-Verschränkungs-Tabelle
O:	HL - (Logische) Sektornummer aus der Tabelle

Das BDOS benutzt diese Funktion, um aus der im DPH angegebenen Sektor-Verschränkungs-Tabelle die Sektornummer zu entnehmen. Im allgemeinen genügt es, einfach die im Registerpaar BC übergebene Sektornummer ins Registerpaar HL zu kopieren.

Hinweise zu den Source-Listings

Die auf den folgenden Seiten abgedruckten Source-Listings von CCP und BDOS sind Ausgaben eines Assemblers. Beide Listings sind auf die Adresse 0000H assembliert, da die Lage von CCP und BDOS im Speicher abhängig vom Computersystem ist.
Die tatsächliche Adresse von BDOS+0006H kann jeweils über den Inhalt der Speicherplätze 0006H und 0007H bestimmt werden.

Zeichen- und Begriffserklärung

Abkürzungen:

UPRO	Abkürzung für 'UnterPROgramm'
I:	Registerbelegung beim Start des Unterprogramms
O:	Registerbelegung nach Ende des Unterprogramms
#	Abkürzung für 'Nummer'
->	Abkürzung für 'enthält Adresse von' oder 'ist Zeiger auf'

Assembler-Direktiven:

equ	Label als Konstante defininieren
defb	Konstante einmal im Speicher ablegen
defs	Konstante mehrfach im Speicher ablegen
defm	Text im Speicher ablegen

Label-Definitionen:

§label	Label ist Datenlabel
?label	Label ist Programmlabel
?§label	Label ist Adresse eines Datum
§label?	Label ist Flag ('wahr' oder 'falsch')
label?	Label für ein Unterprogramm, daß 'wahr' oder 'falsch' zurückgibt

Hexadezimale Werte sind grundsätzlich zwei- oder vierstellig und mit 'h' am Ende angegeben. Alle anderen Werte sind dezimal.

Hinweise

```
;
;
;                       CCP Source Listing
;
;
;
; Dies sind Equates für die verschiedenen ASCII-Zeichen.
;
0007   =   bell    equu      07h
0008   =   back    equu      08h
0009   =   tab     equu      09h
000A   =   lf      equu      0ah
000D   =   cr      equu      0dh
0010   =   ctrlp   equu      10h             ;Dies ist CTRL-P
001A   =   eof     equu      1ah             ;Dies ist CTRL-Z
0020   =   space   equu      20h             ;Leerzeichen
003E   =   prompt  equu      '>'             ;CCP Promptzeichen
007F   =   del     equu      7fh             ;Delete Code
;
;
; 'BOOT'-Adresse für dieses System
;
0000   =   ?boot           equu      0000h
;
;
; Adresse, an der die Nummer der aktuellen Disk
; gespeichert ist:
;
0004   =   §disk           equu      ?boot + 0004h
;
;
; Einsprung ins BDOS:
;
0005   =   ?bdos           equu      ?boot + 0005h
;
;
; Adresse des vom CCP aufgebauten FCB
;
005C   =   ?default_fcb    equu      ?boot + 005ch
;
;
; Adresse des Standard-Disk-Buffers:
;
0080   =   §buffer         equu      ?boot + 0080h
;
;
; Startadresse des Anwenderprogramm-Bereichs:
; (Transient Program Area)
;
0100   =   ?tpa_start      equu      ?boot + 0100h
```

```
;
; Hier sind die Nummern der verschiedenen vom CCP
; benutzten BDOS-Funktionen definiert:
;
; Das 'b_' am Labelanfang, kennzeichnet es als
; BDOS-Funktionsname:
;
0001   =   b_console_input          equ      1

0002   =   b_console_output         equ      2

000A   =   b_read_buffer            equ     10

000B   =   b_console_status         equ     11

000D   =   b_reset_system           equ     13

000E   =   b_select_disk            equ     14

000F   =   b_open_file              equ     15

0010   =   b_close_file             equ     16

0011   =   b_search_first           equ     17

0012   =   b_search_next            equ     18

0013   =   b_delete_file            equ     19

0014   =   b_read_sequ              equ     20

0015   =   b_write_sequ             equ     21

0016   =   b_make_file              equ     22

0017   =   b_rename_file            equ     23

0018   =   b_return_disk            equ     24

0019   =   b_return_current         equ     25

001A   =   b_set_dma                equ     26

0020   =   b_set_get_user           equ     32

0025   =   b_reset_disk            equ     37
```

```
;
; Startadresse des CCP
;
ccp:
;
;
; Dies ist der normale Einsprung zum CCP nach einem
; Warmstart.
; Falls jedoch ein AUTO-Befehl definiert ist, sollte nur
; nach dem Kaltstart hierhin gesprungen werden.
;
; I: C = Usernummer * 16 + Disknummer
;
0000  C3 035C  JP    start_disk_c
;
;
; Einsprung zum CCP ohne Ausführung eines AUTO-Befehls.
;
; I: C = Usernummer * 16 + Disknummer
;
0003  C3 0358  JP    start_no_command
;
;
; Dies ist der CCP-Befehlszeilen-Buffer.
; Jede übernommene Befehlszeile wird hier zwischengespeichert
; und ausgewertet.
; Hier kann auch beim Einsprung in den CCP ein Befehl stehen.
; Auf diese Weise lässt sich ein AUTO-Befehl realisieren, der
; direkt nach dem Systemstart ausgeführt wird.
;
;
;
cmd_buffer:
;
0006  7F       defb  127              ;Länge des Buffers

0007  00       defb  0                ;Anzahl der eingege-
                                      ;benen Zeichen im
                                      ;Buffer

0008  20       defs  16,space         ;Kein AUTO-Befehl im
                                      ;Normalfall
```

```
0018  43        defm  'COPYRIGHT (C)'   ;Copyrightvermerk der
                defm  '1979, DIGITAL '  ;Fa. Digital Research
                defm  'RESEARCH  '

003E  00        defs  74,0              ;Rest des Buffers auf-
                                        ;füllen
;
;
?buffer_address:
;
0088  0008      defw  &cmd_buffer+2     ;Adresse des nächsten
                                        ;Zeichens während der
                                        ;Auswertung des Buf-
                                        ;fers
;
;
?first_char:
;
008A  0000      defw  0                 ;Adresse des ersten
                                        ;Zeichens eines ein-
                                        ;gegebenen Befehls.
                                        ;Wird bei command_
                                        ;error benutzt.
```

```
;
; CCP-interne Konsolenausgabe
;
; I: A - auszugebendes Zeichen
;
; O: -
;
conout:
;
008C  5F        LD    E,A                ;E - auszugebendes
                                         ;Zeichen
008D  0E 02     LD    C,b_console_output ;C - BDOS-Funktions-
                                         ;nummer
008F  C3 0005   JP    ?bdos              ;BDOS-Aufruf
;
;
; Console Ausgabe
; mit Rettung des Registerpaares BC
;
; I: A - auszugebendes Zeichen
;
; O: -
;
conout_bc:
;
0092  C5        PUSH  BC                 ;BC auf Stack
0093  CD 008C   CALL  conout             ;Zeichen Ausgabe
0096  C1        POP   BC                 ;BC von Stack
0097  C9        RET
;
;
; Ausgabe von Carriage Return (cr) und Line Feed (lf)
; zur Beendigung einer Ausgabezeile.
;
print_crlf:
;
0098  3E 0D     LD    A,cr               ;A = cr-Code
009A  CD 0092   CALL  conout_bc          ;Zeichen ausgeben
009D  3E 0A     LD    A,lf               ;A = lf-Code
009F  C3 0092   JP    conout_bc          ;Zeichen ausgeben
;
;
; Ausgabe eines Leerzeichens.
;
print_space:
;
00A2  3E 20     LD    A,space            ;A = Leerzeichen
00A4  C3 0092   JP    conout_bc          ;Zeichen ausgeben
```

```
;
; Ausgabe eines Textes.
;
; I: BC -> Text, abgeschlossen mit 00H
;
; O: -
;
print_text:
;
00A7  C5        PUSH  BC                ;Textzeiger retten
00A8  CD 0098   CALL  print_crlf        ;Neue Ausgabezeile
                                        ;beginnen
00AB  E1        POP   HL                ;Textzeiger zurück
                                        ;nach HL
print_loop:
;
00AC  7E        LD    A,(HL)            ;A - nächstes Zeichen
00AD  B7        OR    A                 ;ist es 00H ?
00AE  C8        RET   Z                 ;J: fertig
00AF  23        INC   HL                ;N: Zeiger +1
00B0  E5        PUSH  HL                ;Zeiger retten
00B1  CD 008C   CALL  conout            ;Zeichen ausgeben
00B4  E1        POP   HL                ;Zeiger zurück
00B5  C3 00AC   JP    print_loop        ;nächstes Zeichen
                                        ;holen
;
;
; BDOS in den Ausgangszustand bringen
;
reset_system:
;
00B8  0E 0D     LD    C,b_reset_system  ;C = BDOS-Funktions-
                                        ;nummer
00BA  C3 0005   JP    ?bdos             ;BDOS-Aufruf
;
;
; Neue Disk anwählen
;
; I: A = Disknummer (00H = A: .. 0FH = P:)
;
select_disk:
;
00BD  5F        LD    E,A               ;E = Disknummer
00BE  0E 0E     LD    C,b_select_disk   ;C = BDOS-Funktions-
                                        ;nummer
00C0  C3 0005   JP    ?bdos             ;BDOS-Aufruf
```

```
;
; CCP-interner BDOS-Aufruf für BDOS-Funktionen, die
; einen Fehlercode zurückgeben (File-Funktionen).
;
; I: C  = BDOS-Funktionsnummer
;    DE = BDOS-Parameter (FCB-Adresse)
;
; O: A  = 0 bei fehlerhaftem BDOS-Aufruf
;      <> 0 sonst
;
bdos_call:
;
00C3  CD 0005  CALL  ?bdos              ;BDOS-Aufruf
00C6  32 07EE  LD    (return_code),A    ;Fehlercode speichern
00C9  3C       INC   A                  ;A - Fehlercode+1
00CA  C9       RET
;
;
; File eröffnen
;
; I: DE -> FCB
;
; O: A - 0 bei Fehler (File nicht gefunden)
;
open_file:
;
00CB  0E 0F    LD    C,b_open_file      ;C - BDOS-Funktions-
                                        ;nummer
00CD  C3 00C3  JP    bdos_call          ;BDOS-Aufruf
;
;
; Internen FCB eröffnen
;
; I: -
;
; O: A = 0 bei Fehler (File nicht gefunden)
;
open_fcb:
;
00D0  AF       XOR   A                  ;A = 0
                                        ;Interne Recordnummer
                                        ;auf 0 setzen
00D1  32 07ED  LD    (internal_record_#),A
00D4  11 07CD  LD    DE,internal_fcb    ;DE = Zeiger auf
                                        ;internen FCB
00D7  C3 00CB  JP    open_file          ;File eröffnen
```

```
;
; File schliessen
;
; I: DE -> FCB
;
; O: A  = 0 bei Fehler
;
close_file:
;
00DA  0E 10    LD    C,b_close_file    ;C = BDOS-Funktions-
                                       ;nummer
00DC  C3 00C3  JP    bdos_call         ;BDOS-Aufruf
;
;
; Ersten Directory-Eintrag eines Files suchen
;
; I: DE -> FCB
;
; O: A  = 0 falls kein Eintrag gefunden wurde
;
search_first:
;
00DF  0E 11    LD    C,b_search_first  ;C = BDOS-Funktions-
                                       ;nummer
00E1  C3 00C3  JP    bdos_call         ;BDOS-Aufruf
;
;
; Nächsten Directory-Eintrag eines Files suchen
;
; I: -
;
; O: A = 0 falls kein weiterer Eintrag gefunden wurde
;
search_next:
;
00E4  0E 12    LD    C,b_search_next   ;C = BDOS-Funktions-
                                       ;nummer
00E6  C3 00C3  JP    bdos_call         ;BDOS-Aufruf
;
;
; Ersten Directory-Eintrag des internen FCB suchen
;
; I: -
;
; O: A = 0 falls kein passender Eintrag gefunden wurde
;
search_fcb:
;
00E9  11 07CD  LD    DE,internal_fcb   ;DE -> interner FCB
00EC  C3 00DF  JP    search_first      ;Ersten Eintrag
                                       ;suchen
```

```
;
; File löschen
;
; I: DE -> FCB
;
; O: A = BDOS-Fehlercode
;
delete_file:
;
00EF  0E 13     LD     C,b_delete_file    ;C - BDOS-Funktions-
                                          ;nummer
00F1  C3 0005   JP     ?bdos              ;BDOS-Aufruf
;
;
; Gemeinsamer BDOS-Aufruf für Schreib- und Leseoperationen
; des CCP.
;
; I: C  -  BDOS-Funktionsnummer
;    DE -> FCB
;
; O: A - BDOS-Fehlercode
;    Flags gesetzt
;
bdos_diskio:
;
00F4  CD 0005   CALL   ?bdos              ;BDOS-Aufruf
00F7  B7        OR     A                  ;Flags setzen
00F8  C9        RET
;
;
; Nächsten Record lesen
;
; I: DE -> FCB
;
; O: A = BDOS-Fehlercode
read_sequ:
;
00F9  0E 14     LD     C,b_read_sequ      ;C = BDOS-Funktions-
                                          ;nummer
00FB  C3 00F4   JP     bdos_diskio        ;BDOS-Aufruf
;
;
; Nächsten Record des internen FCB lesen
;
read_fcb:
;
00FE  11 07CD   LD     DE,internal_fcb    ;DE -> interner FCB
0101  C3 00F9   JP     read_sequ          ;Record lesen
```

```
;
; Nächsten Record schreiben
;
; I: DE -> FCB
;
; O: A - BDOS-Fehlercode
;
write_sequ:
;
0104  0E 15    LD    C,b_write_sequ    ;C - BDOS-Funktions-
                                       ;nummer
0106  C3 00F4  JP    bdos_diskio       ;BDOS-Aufruf
;
;
; Neues File anlegen
;
; I: DE -> FCB
;
; O: A - BDOS-Fehlercode
;
make_file:
;
0109  0E 16    LD    C,b_make_file     ;C - BDOS-Funktions-
                                       ;nummer
010B  C3 00C3  JP    bdos_call         ;BDOS-Aufruf
;
;
; File umbenennen
;
; I: DE -> FCB
;
; O: A = BDOS-Fehlercode
;
rename_file:
;
010E  0E 17    LD    C,b_rename_file   ;C = BDOS-Funktions-
                                       ;nummer
0110  C3 0005  JP    ?bdos             ;BDOS-Aufruf
```

```
;
; Aktuelle Usernummer holen
;
; I: -
;
; O: A - Usernummer
;
get_user:
;
0113  1E FF    LD    E,0FFH            ;E = BDOS-Parameter
                                       ;für 'Usernummer
                                       ;holen'
;
;
; Usernummer setzen
;
; I: E - neue Usernummer
;
; O: -
;
set_user:
;
0115  0E 20    LD    C,b_set_get_user  ;C - BDOS-Funktions-
                                       ;nummer
0117  C3 0005  JP    ?bdos             ;BDOS-Aufruf
;
;
; Aktuelle Disk- und Usernummer zusammenfassen
; und in $disk setzen.
;
set_drv_usr:
;
011A  CD 0113  CALL  get_user          ;Usernummer holen
011D  87       ADD   A,A               ;*2
011E  87       ADD   A,A               ;*4
011F  87       ADD   A,A               ;*8
0120  87       ADD   A,A               ;A = Usernummer * 16
0121  21 07EF  LD    HL,ccp_disk       ;HL -> aktuelle Disk-
                                       ;nummer
0124  B6       OR    (HL)              ;Disknummer einmaskie-
0125  32 0004  LD    ($disk),A         ;ren und speichern
0128  C9       RET
```

```
;
; Aktuelle Disknummer in §disk speichern
;
set_disk:
;
0129  3A 07EF  LD    A,(ccp_disk)      ;A = aktuelle Disk-
                                        ;nummer
012C  32 0004  LD    (§disk),A         ;nach §disk speichern
012F  C9       RET
;
;
; Zeichen in Großbuchstaben wandeln
;
; I: A = ASCII-Zeichen in Groß- oder Kleinschrift
;
; O: A = ASCII-Zeichen in Großschrift
;
make_upper:
;
0130  FE 61    CP    'a'               ;Ist das Zeichen
                                        ;kleiner als 'a' ?
0132  D8       RET   C                 ;J: Zeichen ist kein
                                        ;Kleinbuchstabe
0133  FE 7B    CP    'ä'               ;N: Ist das Zeichen
                                        ;größer als 'ä' ?
0135  D0       RET   NC                ;J: Zeichen ist kein
                                        ;Kleinbuchstabe
0136  E6 5F    AND   05FH              ;N: Zeichen in Groß-
                                        ;buchstaben wandeln
0138  C9       RET
```

```
;
; Befehlszeile holen
;
; Je nach Wert des 'submit_flag' wird die nächste Befehls-
; zeile von der Console (Tastatur) oder aus dem Submit-
; file $$$.SUB geholt.
;
get_command_line:
;
0139  3A 07AB  LD    A,(submit_flag)     ;A - Submit Flag
013C  B7       OR    A                   ;Submit aktiv ?
013D  CA 0196  JP    Z,no_submit         ;N: Zeile von Console
                                         ;holen
0140  3A 07EF  LD    A,(ccp_disk)        ;J: A = aktuelle Disk-
                                         ;nummer
0143  B7       OR    A                   ;Ist A: die aktuelle
                                         ;Disk ? (Disknummer
                                         ;- 0 ?)
0144  3E 00    LD    A,0                 ;A - 0
0146  C4 00BD  CALL  NZ,select_disk      ;N: Disk A: an-
                                         ;wählen

0149  11 07AC  LD    DE,submit_fcb       ;DE -> FCB für $$$.SUB
014C  CD 00CB  CALL  open_file           ;FCB öffnen, $$$.SUB
                                         ;vorhanden ?
014F  CA 0196  JP    Z,no_submit         ;N: Zeile von Console
                                         ;holen
                                         ;J: Letzten Record
                                         ;von $$$.SUB lesen:
0152  3A 07BB  LD    A,(submit_rec_count)
                                         ;A = Anzahl der
                                         ;Records im $$$.SUB-
                                         ;File
                                         ;= Nummer des letzten
                                         ;Records in $$$.SUB+1
0155  3D       DEC   A                   ;A = A-1
                                         ;Nummer des zu lesen-
                                         ;den Records in den
                                         ;submit_fcb setzen
0156  32 07CC  LD    (current_submit_rec),A
0159  11 07AC  LD    DE,submit_fcb       ;DE -> submit_fcb
015C  CD 00F9  CALL  read_sequ           ;Record lesen
015F  C2 0196  JP    NZ,no_submit        ;Fehler ?
                                         ;J: Zeile von Console
                                         ;holen
```

```
                                        ;N: Zeile vom disk-
                                        ;buffer in den Be-
                                        ;fehlsbuffer über-
                                        ;nehmen
0162  11 0007  LD    DE,§cmd_buffer+1   ;DE -> Bufferstart
0165  21 0080  LD    HL,§buffer         ;HL -> Diskbuffer
0168  06 80    LD    B,128              ;B = Anzahl der zu
                                        ;übertragenden Bytes
016A  CD 0442  CALL  move_b_bytes       ;B Bytes von HL nach
                                        ;DE übertragen

016D  21 07BA  LD    HL,submit_fcb_na   ;HL -> 'Not altered'-
                                        ;Flag im $$$.SUB-FCB
0170  36 00    LD    (HL),0             ;auf 0 setzen.
                                        ;Damit ist das File
                                        ;als 'verändert' mark-
                                        ;iert und der folgende
                                        ;'Close File'-Aufruf
                                        ;schreibt die neue
                                        ;Filelänge zurück in
                                        ;die Directory.
                                        ;(siehe auch '_close
                                        ;_file' im BDOS)
0172  23       INC   HL                 ;HL -> Anzahl der
                                        ;Records im $$$.SUB-
                                        ;File
0173  35       DEC   (HL)               ;Recordnummer -1, da
                                        ;gerade ein Record
                                        ;gelesen wurde

0174  11 07AC  LD    DE,submit_fcb      ;DE -> submit_fcb
0177  CD 00DA  CALL  close_file         ;$$$.SUB schliessen
017A  CA 0196  JP    Z,no_submit        ;Fehler ?
                                        ;J: Zeile von Console
                                        ;holen

017D  3A 07EF  LD    A,(ccp_disk)       ;A = CCP Disknummer
0180  B7       OR    A                  ;war die CCP-Disk
                                        ;gleich A: ?
0181  C4 00BD  CALL  NZ,select_disk     ;N: Disk neu anwählen

0184  21 0008  LD    HL,§cmd_buffer+2   ;HL -> Submit Befehls-
                                        ;zeile
0187  CD 00AC  CALL  print_loop         ;Submitbefehl ausgeben
018A  CD 01C2  CALL  check_console      ;Taste gedrückt ?
018D  CA 01A7  JP    Z,set_command      ;N: Submitbefehl aus-
                                        ;führen

                                        ;J: Submit abbrechen
0190  CD 01DD  CALL  delete_submit      ;$$$.SUB löschen
0193  C3 0382  JP    ccp_prompt         ;Zurück zum CCP-Prompt
```

```
;
; Befehlszeile von der Console holen
;
no_submit:
;
0196  CD 01DD  CALL  delete_submit     ;$$$.SUB löschen
0199  CD 011A  CALL  set_drv_usr       ;Aktuelle Disk- und
                                       ;Usernummer zusammen-
                                       ;fassen und für einen
                                       ;eventuellen Warmstart
                                       ;speichern (siehe auch
                                       ;start_disk_c)
019C  0E 0A    LD    C,b_read_buffer   ;C = BDOS-Funktions-
                                       ;nummer für Eingabe-
                                       ;zeile holen
019E  11 0006  LD    DE,§cmd_buffer    ;DE = Bufferadresse
01A1  CD 0005  CALL  ?bdos             ;BDOS-Aufruf
01A4  CD 0129  CALL  set_disk          ;Disknummer speichern
;
;
; Befehlszeile im Buffer zur Auswertung vorbereiten.
; Alle Zeichen in Großbuchstaben umwandeln und Zeilenende
; mit 00H markieren.
;
set_command:
;
01A7  21 0007  LD    HL,§cmd_buffer+1  ;HL -> Anzahl der ein-
                                       ;gelesenen Zeichen
01AA  46       LD    B,(HL)            ;B = Zeichenzähler
;
convert_loop:
;
01AB  23       INC   HL                ;Zeiger +1
01AC  78       LD    A,B               ;A = Zähler
01AD  B7       OR    A                 ;Zähler = 0 ?
01AE  CA 01BA  JP    Z,reset_buffer    ;J: Fertig
01B1  7E       LD    A,(HL)            ;Zeichen holen
01B2  CD 0130  CALL  make_upper        ;In Großbuchstaben
                                       ;umwandeln
01B5  77       LD    (HL),A            ;Zeichen zurück-
                                       ;schreiben
01B6  05       DEC   B                 ;Zähler -1
01B7  C3 01AB  JP    convert_loop      ;Nächstes Zeichen
                                       ;holen
```

```
;
; Zeilenende mit 00H markieren
; Adresse des nächsten Zeichens auf Bufferanfang setzen
;
reset_buffer:
;
01BA  77        LD    (HL),A              ;Zeilenende markieren
                                          ;(A ist noch 00H vom
                                          ;Zählertest bei 01AD)
01BB  21 0008   LD    HL,§cmd_buffer+2    ;HL -> Bufferanfang
                                          ;Adresse des nächsten
                                          ;auszuwertenden Zei-
                                          ;chens auf Bufferan-
                                          ;fang setzen
01BE  22 0088   LD    (?buffer_address),HL
01C1  C9        RET
;
;
; Console auf anstehendes Zeichen überprüfen und
; Zeichen holen
;
; I: -
;
; O: A - 0, falls kein Zeichen anstand
;      - Zeichencode sonst
;
check_console:
;
01C2  0E 0B     LD    C,b_console_status;C = BDOS-Funktions-
                                          ;nummer
01C4  CD 0005   CALL  ?bdos             ;BDOS-Aufruf
01C7  B7        OR    A                 ;Zeichen vorhanden ?
01C8  C8        RET   Z                 ;N: fertig
                                          ;J: Zeichen holen
01C9  0E 01     LD    C,b_console_input ;C = BDOS Funktions-
                                          ;nummer
01CB  CD 0005   CALL  ?bdos             ;BDOS-Aufruf
01CE  B7        OR    A                 ;Flags setzen
01CF  C9        RET
;
;
; Aktuelle BDOS-Disknummer holen
;
; I: -
;
; O: A = BDOS Disknummer
;
get_disk_code:
;
01D0  0E 18     LD    C,b_return_disk   ;C = BDOS Funktions-
                                          ;nummer
01D2  C3 0005   JP    ?bdos             ;BDOS Aufruf
```

```
;
;
; DMA-Adresse auf §buffer setzen
;
set_default_dma:
;
01D5  11 0080  LD    DE,§buffer         ;DE -> §buffer
;
;
; DMA-Adress setzen
;
; I: DE - neue DMA-Adresse
;
; O: -
;
set_dma:
;
01D8  0E 1A    LD    C,b_set_dma        ;C - BDOS-Funktions-
                                        ;nummer
01DA  C3 0005  JP    ?bdos              ;BDOS-Aufruf
;
;
; Submit-File $$$.SUB löschen
;
delete_submit:
;
01DD  21 07AB  LD    HL,submit_flag     ;HL -> submit_flag
01E0  7E       LD    A,(HL)             ;A - submit_flag
01E1  B7       OR    A                  ;submit aktiv ?
01E2  C8       RET   Z                  ;N: dann braucht auch
                                        ;nichts gelöscht zu
                                        ;werden

01E3  36 00    LD    (HL),0             ;J: submit_flag auf
                                        ;inaktiv setzen
01E5  AF       XOR   A                  ;A = 0
01E6  CD 00BD  CALL  select_disk        ;Disk A: anwählen
01E9  11 07AC  LD    DE,submit_fcb      ;DE -> submit_fcb
01EC  CD 00EF  CALL  delete_file        ;$$$.SUB löschen
01EF  3A 07EF  LD    A,(ccp_disk)       ;A = CCP-Disknummer
01F2  C3 00BD  JP    select_disk        ;CCP-Disk wieder an-
                                        ;wählen
```

```
;
; Seriennummern von CCP und BDOS vergleichen
;
compare_serial:
;
01F5  11 0328  LD    DE,serial_number   ;DE -> CCP Serien-
                                        ;nummer
01F8  21 0800  LD    HL,bdos            ;HL -> BDOS Serien-
                                        ;nummer
01FB  06 06    LD    B,6                ;6 Bytes vergleichen
;
compare_loop:
;
01FD  1A       LD    A,(DE)             ;A = Byte aus CCP
01FE  BE       CP    (HL)               ;gleich dem Byte aus
                                        ;dem BDOS ?
01FF  C2 03CF  JP    NZ,wrong_serial    ;N: falsche Serien-
                                        ;nummer
0202  13       INC   DE                 ;CCP-Zeiger +1
0203  23       INC   HL                 ;BDOS-Zeiger +1
0204  05       DEC   B                  ;Zähler -1
0205  C2 01FD  JP    NZ,compare_loop    ;Nächstes Byte ver-
                                        ;gleichen
0208  C9       RET
;
;
; Fehler in der Befehlszeile erkannt.
; Ab dem ?first_char wird das nächste Wort oder der Rest
; der Zeile, gefolgt von einem Fragezeichen ausgegeben.
;
command_error:
;
0209  CD 0098  CALL  print_crlf          ;Neue Ausgabezeile
                                        ;beginnen
020C  2A 008A  LD    HL,(?first_char)   ;HL -> Fehlerhafte
                                        ;Befehlszeile
print_next_char:
;
020F  7E       LD    A,(HL)             ;Zeichen holen
0210  FE 20    CP    space              ;Leerzeichen ?
                                        ;J: Wortende erreicht,
                                        ;Fragezeichen ausgeben
0212  CA 0222  JP    Z,print_question_mark
0215  B7       OR    A                  ;00H gefunden ?
                                        ;J: Zeilenende
                                        ;erreicht, Frage-
                                        ;zeichen ausgeben
0216  CA 0222  JP    Z,print_question_mark
0219  E5       PUSH  HL                 ;N: Zeiger retten
021A  CD 008C  CALL  conout             ;Zeichen ausgeben
021D  E1       POP   HL                 ;Zeiger zurück
021E  23       INC   HL                 ;Zeiger +1
021F  C3 020F  JP    print_next_char    ;nächstes Zeichen
```

```
;
; Fragezeichen im Anschluß an den fehlerhaften Befehl
; ausgeben.
; Falls der fehlerhafte Befehl aus einem Submit-File kam,
; wird der SUBMIT-Lauf abgebrochen.
;
print_question_mark:
;
0222  3E 3F     LD    A,'?'              ;A - Fragezeichen
0224  CD 008C   CALL  conout             ;Zeichen ausgeben
0227  CD 0098   CALL  print_crlf          ;Zeile beenden
022A  CD 01DD   CALL  delete_submit      ;$$$.SUB abbrechen
022D  C3 0382   JP    ccp_prompt         ;Zurück zum CCP-Prompt
;
;
; Nächstes Zeichen aus dem Befehlsbuffer holen
; und überprüfen
;
; I: DE -> zu holendes Zeichen
;
; O: A - Zeichen
;    Z-Flag gesetzt, falls ein Trennungszeichen gefunden
;    wurde. Trennungszeichen sind: - _ . : ; < >
;
get_check_char:
;
0230  1A       LD    A,(DE)             ;Zeichen holen
0231  B7       OR    A                  ;Zeichen - 00H ?
0232  C8       RET   Z                  ;J: Zeilenende
                                        ;erreicht
0233  FE 20     CP    space              ;Ist das Zeichen ein
                                        ;Control-Code, also
                                        ;kleiner als space ?
0235  DA 0209   JP    C,command_error    ;J: Fehler
0238  C8       RET   Z                  ;ok, falls das Zeichen
                                        ;gleich space ist.
0239  FE 3D     CP    '='                ;Zeichen gleich '=' ?
023B  C8       RET   Z                  ;J: ok
023C  FE 5F     CP    '_'                ;Zeichen gleich '_' ?
023E  C8       RET   Z                  ;J: ok
023F  FE 2E     CP    '.'                ;Zeichen gleich '.' ?
0241  C8       RET   Z                  ;J: ok
0242  FE 3A     CP    ':'                ;Zeichen gleich ':' ?
0244  C8       RET   Z                  ;J: ok
0245  FE 3B     CP    ';'                ;Zeichen gleich ';' ?
0247  C8       RET   Z                  ;J: ok
0248  FE 3C     CP    '<'                ;Zeichen gleich '<' ?
024A  C8       RET   Z                  ;J: ok
024B  FE 3E     CP    '>'                ;Zeichen gleich '>' ?
024D  C8       RET   Z                  ;J: ok
024E  C9       RET                      ;N: Das Zeichen ist
                                        ;kein Trennungszeichen
```

```
;
;
; Nächstes Zeichen ungleich Leerzeichen holen
;
; I: DE -> Befehlszeile
;
; O: A = 0 falls kein Zeichen mehr gefunden wurde
;      = Zeichen sonst
;
get_next_char:
;
024F  1A       LD    A,(DE)          ;A = nächstes Zeichen
0250  B7       OR    A               ;Zeichen = 00H ?
0251  C8       RET   Z               ;J: Zeilenende
                                     ;erreicht
0252  FE 20    CP    space           ;Leerzeichen ge-
                                     ;funden ?
0254  C0       RET   NZ              ;N: Fertig
0255  13       INC   DE              ;J: Zeiger +1
0256  C3 024F  JP    get_next_char   ;Nächstes Zeichen
                                     ;testen
;
;
; A zu HL addieren
;
; I: A , HL
;
; O: HL = HL + A
;
add_hl_a:
;
0259  85       ADD   A,L             ;Erst die LSBs ad-
025A  6F       LD    L,A             ;addieren
025B  D0       RET   NC              ;Fertig, falls kein
                                     ;Überlauf
025C  24       INC   H               ;Sonst Überlauf in H
025D  C9       RET                   ;auffangen
```

```
;
; Zeichen ab ?buffer_address als Filenamen interpretieren
; und in den internen FCB übernehmen.
;
; I: ?buffer_address -> Filename
;
; O: A - Anzahl der '?', die der Filenamen enthält
;      - 0, falls der Filename eindeutig ist
;
setup_fcb:
;
025E  3E 00     LD    A,0
;
;
; Zeichen ab ?buffer_address als Filenamen interpretieren
; und in den internen FCB ab Positon A übernehmen.
; Dieser Einsprung wird vom 'REN'-Befehl benutzt, um den
; internen FCB für die BDOS-Funktions 'Rename File' vor-
; zubereiten
;
setup_fcb_offset:
;
0260  21 07CD  LD    HL,internal_fcb   ;HL -> interner FCB
0263  CD 0259  CALL  add_hl_a          ;A aufaddieren
0266  E5       PUSH  HL                ;FCB Zeiger zweimal
0267  E5       PUSH  HL                ;retten
0268  AF       XOR   A                 ;A - 0
0269  32 07F0  LD    (file_disk_no),A  ;Aktuelle Disk als
                                       ;Disk des Files an-
                                       ;nehmen
026C  2A 0088  LD    HL,(?buffer_address)
                                       ;HL -> Buffer
026F  EB       EX    DE,HL
0270  CD 024F  CALL  get_next_char     ;Nächstes Zeichen un-
                                       ;gleich 'space' holen
0273  EB       EX    DE,HL
0274  22 008A  LD    (?first_char),HL  ;Adresse dieses Zei-
                                       ;chens für die Fehler-
                                       ;routine merken
0277  EB       EX    DE,HL             ;DE -> Buffer
0278  E1       POP   HL                ;HL -> FCB
0279  1A       LD    A,(DE)            ;Zeichen holen
027A  B7       OR    A                 ;Zeilenende erreicht ?
027B  CA 0289  JP    Z,no_new_disk     ;J: keine Diskbezeich-
                                       ;nung gefunden
```

```
                                                ;N: Zeichen als Disk-
                                                ;bezeichnung auffassen
027E  DE 40     SBC   A,'@'                     ;Diskcode berechnen
0280  47        LD    B,A                       ;B = Diskcode
0281  13        INC   DE                        ;Bufferzeiger +1
0282  1A        LD    A,(DE)                    ;nächstes Zeichen
                                                ;holen
0283  FE 3A     CP    ':'                       ;Ist es ein ':' ?
0285  CA 0290   JP    Z,set_new_disk            ;J: Das Zeichen war
                                                ;eine Diskbezeichnung
0288  1B        DEC   DE                        ;N: Zeiger -1
;
;
; Keine Diskbezeichnung gefunden.
; Als Disk für den internen FCB wird die aktuelle Disk
; genommen.
;
no_new_disk:
;
0289  3A 07EF   LD    A,(ccp_disk)              ;A = aktuelle Disk-
                                                ;nummer
028C  77        LD    (HL),A                    ;A als Disknummer in
                                                ;den FCB übernehmen
028D  C3 0296   JP    setup_filename            ;Filenamen übernehmen
;
;
; Diskbezeichnung gefunden.
;
set_new_disk:
;
0290  78        LD    A,B                       ;A = Disknummer
0291  32 07F0   LD    (file_disk_no),A          ;Nummer merken
0294  70        LD    (HL),B                    ;und in FCB über-
                                                ;nehmen
0295  13        INC   DE                        ;Bufferzeiger +1
```

```
;
; Filenamen in den FCB übernehmen.
; Das Zeichen '*' wird als 'Joker' erkannt und die rest-
; lichen Zeichen des Filenamens mit '?' aufgefüllt.
;
setup_filename:
;
0296  06 08     LD    B,8                ;Der Filenamen hat
                                         ;maximal 8 Zeichen
;
setup_filename_loop:
;
0298  CD 0230   CALL  get_check_char     ;Zeichen aus Buffer
                                         ;holen
                                         ;Zeilenende erreicht ?
029B  CA 02B9   JP    Z,fill_fcb_space   ;J: restliche Zeichen
                                         ;im FCB mit Leerzei-
                                         ;chen auffüllen
029E  23        INC   HL                 ;N: FCB-Zeiger +1
029F  FE 2A     CP    '*'                ;'Joker' gefunden ?
02A1  C2 02A9   JP    NZ,set_char_to_fcb;N: Zeichen übernehmen
02A4  36 3F     LD    (HL),'?'           ;J: '?' einsetzen und
                                         ;Bufferzeiger NICHT
                                         ;erhöhen
02A6  C3 02AB   JP    next_filename_char;Das nächste Zeichen
                                         ;wird wieder '*' sein.
                                         ;Dadurch werden die
                                         ;restlichen Zeichen
                                         ;des Filenamens im FCB
                                         ;mit '?' belegt
;
; Zeichen in den FCB übernehmen und Bufferzeiger erhöhen
;
set_char_to_fcb:
;
02A9  77        LD    (HL),A             ;Zeichen in FCB über-
                                         ;nehmen
02AA  13        INC   DE                 ;Bufferzeiger +1
;
; Nächstes Zeichen vom Buffer in den FCB übernehmen, falls
; noch nicht 8 Zeichen erreicht sind.
;
next_filename_char:
;
02AB  05        DEC   B                  ;Zähler -1
                                         ;Zähler = 0 ?
                                         ;N: nächstes Zeichen
                                         ;übernehmen
02AC  C2 0298   JP    NZ,setup_filename_loop
```

```
;
; Ende des Filenamens und damit den Anfang des Filetyps im
; Buffer suchen.
;
search_filetype:
;
02AF  CD 0230  CALL  get_check_char     ;Nächstes Zeichen aus
                                        ;dem Buffer holen.
                                        ;Ist es ein Trennungs-
                                        ;zeichen ?
02B2  CA 02C0  JP    Z,setup_filetype  ;J: Ende des File-
                                        ;namens erreicht
02B5  13       INC   DE                ;N: Bufferzeiger +1
02B6  C3 02AF  JP    search_filetype   ;und weitersuchen
;
;
; Restliche Zeichen des Filenamens im FCB mit Leerzeichen
; auffüllen.
;
fill_fcb_space:
;
02B9  23       INC   HL                ;FCB-Zeiger +1
02BA  36 20    LD    (HL),space        ;Leerzeichen einsetzen
02BC  05       DEC   B                 ;Zähler -1
02BD  C2 02B9  JP    NZ,fill_fcb_space ;Weiter, bis Zähler -0
;
;
; Filetyp in internen FCB übernehmen
;
; I: A = Trennungszeichen nach dem Filenamen
;
setup_filetype:
;
02C0  06 03    LD    B,3               ;Der Filetyp hat
                                        ;maximal 3 Zeichen
02C2  FE 2E    CP    '.'               ;Ist das Trennungs-
                                        ;zeichen ein Punkt ?
02C4  C2 02E9  JP    NZ,fill_typ_space ;N: Es folgt kein
                                        ;Filetyp
                                        ;Entsprechende Stellen
                                        ;im FCB mit Leerzei-
                                        ;chen auffüllen
```

```
02C7  13        INC   DE                ;J: Filetyp folgt,
                                        ;Bufferzeiger +1
;
setup_filetype_loop:
;
02C8  CD 0230   CALL  get_check_char    ;Nächstes Zeichen aus
                                        ;dem Buffer holen
                                        ;Trennungszeichen ge-
                                        ;funden ?
02CB  CA 02E9   JP    Z,fill_typ_space  ;J: Restliche Zeichen
                                        ;mit Leerzeichen auf-
                                        ;füllen
02CE  23        INC   HL                ;FCB-Zeiger +1
02CF  FE 2A     CP    '*'               ;'Joker' gefunden ?
02D1  C2 02D9   JP    NZ,set_char_to_typ;N: Zeichen übernehmen
02D4  36 3F     LD    (HL),'?'          ;J: '?' einsetzen und
                                        ;Bufferzeiger NICHT
                                        ;erhöhen
02D6  C3 02DB   JP    next_filetype_char
;
;
; Zeichen in den FCB übernehmen und Bufferzeiger erhöhen
;
set_char_to_typ:
;
02D9  77        LD    (HL),A            ;Zeichen in den FCB
                                        ;einsetzen
02DA  13        INC   DE                ;Bufferzeiger +1
;
next_filetype_char:
;
02DB  05        DEC   B                 ;Zähler -1
                                        ;Zähler = 0 ?
                                        ;N: nächstes Zeichen
                                        ;übernehmen
02DC  C2 02C8   JP    NZ,setup_filetype_loop
;
;
; Ende des Filetyps suchen.
;
search_end_of_filetype:
;
02DF  CD 0230   CALL  get_check_char    ;Zeichen aus Buffer
                                        ;holen. Trennungs-
                                        ;zeichen gefunden ?
02E2  CA 02F0   JP    Z,clear_rest      ;J: Fertig
02E5  13        INC   DE                ;N: Bufferzeiger +1
02E6  C3 02DF   JP    search_end_of_filetype
```

```
;
; Restliche Zeichen des Filetyps im FCB mit Leerzeichen
; auffüllen.
;
fill_typ_space:
;
02E9  23        INC   HL                  ;FCB-Zeiger +1
02EA  36 20     LD    (HL),space          ;Leerzeichen einsetzen
02EC  05        DEC   B                   ;Zähler -1
02ED  C2 02E9   JP    NZ,fill_typ_space   ;Weiter, bis Zähler =0
;
; Die nächsten 3 Bytes im FCB auf 00H setzen
;
clear_rest:
;
02F0  06 03     LD    B,3                 ;3 Bytes
;
clear_rest_loop:
;
02F2  23        INC   HL                  ;Zeiger +1
02F3  36 00     LD    (HL),0              ;Byte auf 00H setzen
02F5  05        DEC   B                   ;Zähler -1
02F6  C2 02F2   JP    NZ,clear_rest_loop  ;Weiter, bis Zähler =0
02F9  EB        EX    DE,HL               ;HL - Bufferzeiger
                                          ;Aktuellen Bufferzei-
                                          ;ger für nächsten
                                          ;Aufruf abspeichern
02FA  22 0088   LD    (?buffer_address),HL
02FD  E1        POP   HL                  ;FCB-Startzeiger
                                          ;zurück
;
; Aufgebauten FCB nach '?' absuchen
;
02FE  01 000B   LD    BC,11               ;B = 0, C = 11
;
check_wildcard:
;
0301  23        INC   HL                  ;FCB-Zeiger +1
0302  7E        LD    A,(HL)              ;Zeichen holen
0303  FE 3F     CP    '?'                 ;Ist es '?' ?
0305  C2 0309   JP    NZ,no_wildcard      ;N: weiter
0308  04        INC   B                   ;J: '?' gefunden
;
no_wildcard:
;
0309  0D        DEC   C                   ;Zähler -1
030A  C2 0301   JP    NZ,check_wildcard   ;Nächstes Zeichen
                                          ;überprüfen
030D  78        LD    A,B                 ;A = Anzahl der ge-
                                          ;fundenen '?'
030E  B7        OR    A                   ;Flags setzen
030F  C9        RET
```

```
;
; Tabelle der vom CCP erkannten Befehle
;
cmd_table:
;
0310  44        defm  'DIR '
0314  45        defm  'ERA '
0318  54        defm  'TYPE'
031C  53        defm  'SAVE'
0320  52        defm  'REN '
0324  55        defm  'USER'
;
;
; CCP-Seriennummer
;
serial_number:
;
0328  00        defs  6,0
;
;
; Eingegebenen Filenamen (Befehl) in der cmd_table suchen
;
; I: Befehl, über 'setup_fcb' im internal_fcb eingetragen
;
; O: A - relative Position des Befehls in der cmd_table
;      - 0 entspricht 'DIR'
;      - 1 entspricht 'ERA'
;      - 2 entspricht 'TYPE'
;      = 3 entspricht 'SAVE'
;      = 4 entspricht 'REN'
;      = 5 entspricht 'USER'
;      = 6 Befehl nicht erkannt (Programmname ?!)
;
search_ccp_cmd:
;
032E  21 0310  LD    HL,cmd_table      ;HL -> Befehlstabelle
0331  0E 00    LD    C,0               ;C ist Befehlszähler
;
;
; Nächsten Befehl vergleichen
;
compare_next_cmd:
;
0333  79       LD    A,C               ;Ist der Befehlszähler
0334  FE 06    CP    6                 ;gleich 6 ?
0336  D0       RET   NC                ;J: Keinen der sechs
                                       ;Befehle erkannt.
```

```
                                                ;J: DE -> Filename
                                                ;(Befehl)
0337  11 07CE  LD    DE,internal_filename
033A  06 04    LD    B,4                       ;4 Zeichen werden
                                                ;maximal verglichen
compare_next_char:
;
033C  1A       LD    A,(DE)                    ;Stimmt das nächste
                                                ;Zeichen aus dem File-
                                                ;namen mit dem nächs-
033D  BE       CP    (HL)                      ;ten Zeichen der Be-
                                                ;fehlstabelle überein?
033E  C2 034F  JP    NZ,inc_to_next_cmd        ;N: Befehlstabellen-
                                                ;zeiger auf nächsten
                                                ;Befehl erhöhen
0341  13       INC   DE                        ;J: Befehlszeiger +1
0342  23       INC   HL                        ;Tabellenzeiger +1
0343  05       DEC   B                         ;Zeichenzähler -1
                                                ;Alle 4 Zeichen ver-
                                                ;glichen ?
                                                ;N: Nächstes Zeichen
                                                ;prüfen
0344  C2 033C  JP    NZ,compare_next_char
                                                ;J: Die ersten 4 Zei-
                                                ;chen des Filenamens
                                                ;wurden erkannt.
                                                ;Ist aber der einge-
                                                ;gebene Befehl damit
                                                ;abgeschlossen ?
0347  1A       LD    A,(DE)                    ;Nächstes Zeichen des
                                                ;Filenamens holen
0348  FE 20    CP    space                     ;Folgt ein Leer-
                                                ;zeichen ?
                                                ;N: Befehl nicht voll-
                                                ;ständig erkannt
034A  C2 0354  JP    NZ,inc_cmd_counter
034D  79       LD    A,C                       ;J: A = relative Num-
                                                ;mer des erkannten
                                                ;Befehls
034E  C9       RET
;
;
; Befehlstabellenzeiger auf nächsten Befehl erhöhen
;
inc_to_next_cmd:
;
034F  23       INC   HL                        ;Zeiger +1
0350  05       DEC   B                         ;Zeichenzähler -1
0351  C2 034F  JP    NZ,inc_to_next_cmd        ;Weiter, bis Zeichen-
                                                ;zähler = 0
```

```
;
; Befehlszähler erhöhen und nächsten Befehl vergleichen
;
inc_cmd_counter:
;
0354  0C       INC   C                   ;Befehlszähler +1
0355  C3 0333  JP    compare_next_cmd    ;Nächsten Befehl ver-
                                         ;gleichen
;
;
; CCP-Start ohne Ausführung eines Auto-Befehls
;
; I: C = Zusammengefasste User- und Disknummer für die
;        weitere Bearbeitung
;
start_nc_command:
;
0358  AF       XOR   A                   ;A = 0
0359  32 0007  LD    (&cmd_buffer+1),A   ;Anzahl der im Buffer
                                         ;befindlichen Zeichen
                                         ;auf Null setzen
;
;
; CCP-Start mit Ausführung eines Auto-Befehls
;
; I: C = Zusammengefasste User- und Disknummer für die
;        weitere Bearbeitung
;
start_disk_c:
;
035C  31 07AB  LD    SP,stack_area       ;Stackpointer laden
035F  C5       PUSH  BC                  ;User/Disknummer
                                         ;retten
0360  79       LD    A,C                 ;A = User/Disknummer
0361  1F       RRA                       ;A um 4 Bits rotieren
0362  1F       RRA
0363  1F       RRA
0364  1F       RRA                       ;Die oberen 4 Bits
                                         ;bilden die Disk- die
                                         ;unteren 4 Bits die
                                         ;Usernummer
0365  E6 0F    AND   0FH                 ;Disknummer aus-
                                         ;maskieren
0367  5F       LD    E,A                 ;E = gewünschte User-
                                         ;nummer
0368  CD 0115  CALL  set_user            ;Usernummer setzen
```

```
036B  CD 00B8  CALL  reset_system      ;BDOS in den Ausgangs-
                                       ;zustand bringen und
                                       ;Disk A: einloggen.
                                       ;A = FFH, falls ein
                                       ;$-File gefunden
                                       ;wurde. (siehe BDOS
                                       ;'logical_select')
036E  32 07AB  LD    (submit_flag),A   ;Wert merken, es kön-
                                       ;nte sich um ein Sub-
                                       ;mit-File ($$$.SUB)
                                       ;handeln
0371  C1       POP   BC                ;User/Disknummer
                                       ;zurück
0372  79       LD    A,C               ;A = Nummer
0373  E6 0F    AND   0FH               ;Usernummer ausmask-
                                       ;ieren
0375  32 07EF  LD    (ccp_disk),A      ;Disknummer als CCP-
                                       ;Disk speichern
0378  CD 00BD  CALL  select_disk        ;Disk selektieren
037B  3A 0007  LD    A,(&cmd_buffer+1) ;A = Anzahl der Zei-
                                       ;chen im Buffer
037E  B7       OR    A                 ;Sind bereits Zeichen
                                       ;im Buffer vorhanden ?
037F  C2 0398  JP    NZ,execute_cmd    ;J: Zeichen als Auto-
                                       ;befehl auffassen und
                                       ;den Befehl ausführen
;
; CCP-Neustart nach Ausführung eines CCP-Befehls
;
ccp_prompt:
;
0382  31 07AB  LD    SP,stack_area     ;Stackpointer laden
0385  CD 0098  CALL  print_crlf         ;Neue Ausgabezeile
                                       ;beginnen
0388  CD 01D0  CALL  get_disk_code      ;Aktuelle Disknummer
                                       ;holen
038B  C6 41    ADD   A,'A'             ;+ ASCII-Wert von 'A'
                                       ;ergibt Buchstaben
                                       ;zwischen 'A' und 'P'
038D  CD 008C  CALL  conout             ;Buchstaben ausgeben
0390  3E 3E    LD    A,'>'             ;'>'
0392  CD 008C  CALL  conout             ; ausgeben
0395  CD 0139  CALL  get_command_line   ;Befehlszeile holen
                                       ;(entweder von der
                                       ;Console oder aus
                                       ;einem Submit-File)
```

```
;
; Im §cmd_buffer stehende Befehlszeile interpretieren und
; ausführen
;
execute_cmd:
;
0398  11 0080  LD    DE,§buffer          ;DE -> §buffer
039B  CD 01D8  CALL  set_dma             ;DMA-Adresse auf
                                         ;§buffer setzen
039E  CD 01D0  CALL  get_disk_code       ;Aktuelle BDOS-Disk-
                                         ;nummer holen
03A1  32 07EF  LD    (ccp_disk),A       ;und als CCP-Disk
                                         ;speichern
03A4  CD 025E  CALL  setup_fcb           ;Eingegebenen Befehl
                                         ;als Filenamen auf-
                                         ;fassen und in den
                                         ;internal_fcb über-
                                         ;nehmen
03A7  C4 0209  CALL  NZ,command_error    ;Fehler, falls ein
                                         ;mehrdeutiger Datei-
                                         ;name (also mit '*'
                                         ;oder '?') angegeben
                                         ;wurde
03AA  3A 07F0  LD    A,(file_disk_no)   ;Wurde eine Diskbe-
                                         ;zeichnung vorange-
                                         ;stellt ?
03AD  B7       OR    A                  ;Dann ist die 'file_
                                         ;disk_no' nicht 0
                                         ;(siehe setup_fcb)
03AE  C2 06A5  JP    NZ,execute_program ;J: Befehl als File-
                                         ;namen eines Programms
                                         ;auffassen und dieses
                                         ;Programm ausführen
03B1  CD 032E  CALL  search_ccp_cmd      ;N: Wurde einer der
                                         ;CCP-Befehle benutzt ?
03B4  21 03C1  LD    HL,cmd_adresses    ;HL -> Tabelle der
                                         ;Startadressen der
                                         ;CCP-Befehlsroutinen
03B7  5F       LD    E,A                ;DE = Befehlsnummer
03B8  16 00    LD    D,0
03BA  19       ADD   HL,DE              ;Befehlsnummer zweimal
03BB  19       ADD   HL,DE              ;aufaddieren ergibt
                                         ;Zeiger auf die Adres-
                                         ;se des gewünschten
                                         ;Befehls
03BC  7E       LD    A,(HL)             ;LSB der Adresse laden
03BD  23       INC   HL                 ;Zeiger +1
03BE  66       LD    H,(HL)             ;MSB der Adresse laden
03BF  6F       LD    L,A                ;HL = Adresse
03C0  E9       JP    (HL)               ;Befehlsroutine aus-
                                         ;führen
```

```
;
; Adress-Tabelle der in der cmd_table aufgeführten
; Befehle.
; Der von 'search_ccp_cmd' gelieferte Wert entspricht
; der Position der Adresse in dieser Tabelle
;
cmd_adresses:
;
03C1  0477     defw  dir                ;'DIR'
03C3  051F     defw  era                ;'ERA'
03C5  055D     defw  type               ;'TYPE'
03C7  05AD     defw  save               ;'SAVE'
03C9  0610     defw  ren                ;'REN'
03CB  068E     defw  user               ;'USER'
03CD  06A5     defw  execute_program    ;Nicht erkannter Be-
                                        ;fehl, der Befehl
                                        ;wird als Programm-
                                        ;name aufgefasst und
                                        ;das entsprechende
                                        ;Programm gestartet
;
;
; Beim Vergleich der beiden Seriennummern von CCP und BDOS
; wurde eine Abweichung entdeckt.
; Diese Routine hier lässt den CCP 'abstürzen'
;
wrong_serial:
;
03CF  21       defb  21h                ;'LD HL,'
03D0  F3       DI                       ;HL mit den Befehlen
03D1  76       HALT                     ;'DI' und 'HALT'
                                        ;laden.
03D2  22 0000  LD    (ccp),HL           ;Diese Befehle beim
                                        ;CCP-Start eintragen
03D5  21 0000  LD    HL,ccp             ;und zum CCP-Start
03D8  E9       JP    (HL)               ;springen
;
;
; UPRO für TYPE
; Lesefehler melden, 'READ ERROR' ausgeben
;
read_error:
;
03D9  01 03DF  LD    BC,txt_read_error ;BC -> Text
03DC  C3 00A7  JP    print_text        ;Text ausgeben
;
;
txt_read_error:
;
03DF  52       defm  'READ ERROR'
03E9  00       defb  0
```

```
;
; UPRO für DIR und REN
; File nicht gefunden, 'NO FILE' ausgeben
;
no_file:
;
03EA  01 03F0  LD    BC,txt_no_file    ;BC -> Text
03ED  C3 00A7  JP    print_text        ;Text ausgeben
;
txt_no_file:
;
03F0  4E       defm  'NO FILE'
03F7  00       defb  0
;
;
; UPRO für USER und SAVE
; Die nach dem Befehl angegebene Nummer auswerten.
;
; I: ?buffer_address zeigt auf das, dem Befehlswort folgende
;    Zeichen
;
; O: A = Nummer, falls sie eindeutig ist und im Bereich
;                von 0 bis 255 liegt.
;    Sprung nach 'command_error' sonst
;
get_number:
;
03F8  CD 025E  CALL  setup_fcb         ;Zeichen ab ?buffer_
                                       ;address in den inter-
                                       ;nal_fcb übernehmen
03FB  3A 07F0  LD    A,(file_disk_no)  ;Wurde eine Diskbe-
03FE  B7       OR    A                 ;zeichnung gefunden?
03FF  C2 0209  JP    NZ,command_error  ;J: Keine Ziffer an-
                                       ;gegeben, Fehler
                                       ;N: HL -> Ziffern
0402  21 07CE  LD    HL,internal_filename
0405  01 000B  LD    BC,11             ;B = 0 (Zwischensumme)
                                       ;C = 11 (Ziffern-
                                       ;       zähler)
get_number_loop:
;
0408  7E       LD    A,(HL)            ;Nächste Ziffer holen
0409  FE 20    CP    space             ;Leerzeichen gefunden?
040B  CA 0433  JP    Z,get_number_exit ;J: Restliche Zeichen
                                       ;des Filenamens über-
                                       ;prüfen
040E  23       INC   HL                ;N: Zeiger +1
040F  D6 30    SUB   '0'               ;ASCII-Wert in Dezi-
                                       ;malwert umrechnen
0411  FE 0A    CP    0AH               ;Liegt der Wert im
                                       ;Ziffernbereich ?
0413  D2 0209  JP    NC,command_error  ;N: Keine Ziffer ge-
                                       ;funden, Fehler
```

```
0416  57        LD    D,A                ;Neuen Ziffernwert
                                         ;speichern
0417  78        LD    A,B                ;A = Zwischensumme
0418  E6 E0     AND   0E0H               ;Ist die Zwischensumme
                                         ;bereits > 31 ?
041A  C2 0209   JP    NZ,command_error   ;J: Mit der neuen Zif-
                                         ;fer würde sich ein
                                         ;Wert von mindestens
                                         ;320 ergeben, Fehler.
041D  78        LD    A,B                ;N: A = Zwischensumme
041E  07        RLCA                     ;*2
041F  07        RLCA                     ;*4
0420  07        RLCA                     ;*8
0421  80        ADD   A,B                ;+ Zwischensumme
                                         ;= Zwischensumme * 9
                                         ;Überlauf (>255) ?
0422  DA 0209   JP    C,command_error    ;J: Fehler
0425  80        ADD   A,B                ;N:+ Zwischensumme
                                         ;= Zwischensumme * 10
                                         ;Überlauf (>255) ?
0426  DA 0209   JP    C,command_error    ;J: Fehler
0429  82        ADD   A,D                ;Zwischensumme * 10
                                         ;+ neue Ziffer ergibt
                                         ;die neue Zwischen-
                                         ;summe
                                         ;Überlauf (>255) ?
042A  DA 0209   JP    C,command_error    ;J: Fehler
042D  47        LD    B,A                ;N: Neue Zwischen-
                                         ;summe speichern
042E  0D        DEC   C                  ;Ziffernzähler -1
042F  C2 0408   JP    NZ,get_number_loop;Weiter, bis Ziffern-
                                         ;zähler = 0
0432  C9        RET
;
; Leerzeichen gefunden, Ziffernzähler noch größer 0
;
get_number_exit:
;
0433  7E        LD    A,(HL)             ;Nächstes Zeichen
                                         ;holen
0434  FE 20     CP    space              ;Leerzeichen ?
0436  C2 0209   JP    NZ,command_error   ;N: Fehler
0439  23        INC   HL                 ;J: Zeiger +1
043A  0D        DEC   C                  ;Ziffernzähler -1
043B  C2 0433   JP    NZ,get_number_exit;Weiter, bis Ziffern-
                                         ;zähler = 0
043E  78        LD    A,B                ;A = Nummer
043F  C9        RET
```

```
;
; UPRO für 'load_and_execute' zur Einsetzung des Filetyps 'COM'
; in den angegebenen Filenamen.
;
; 3 Bytes von (HL) nach (DE) kopieren
;
; I: DE = Zieladresse
;    HL = Quelladresse
;
move_3_bytes:
;
0440  06 03     LD    B,3                ;B ist Zähler
;
;
; B Bytes von (HL) nach (DE) kopieren
;
; I: B  = Anzahl der zu kopierenden Bytes
;    DE = Zieladresse
;    HL = Quelladresse
;
move_b_bytes:
;
0442  7E        LD    A,(HL)             ;Byte holen
0443  12        LD    (DE),A             ;Byte speichern
0444  23        INC   HL                 ;Quellzeiger +1
0445  13        INC   DE                 ;Zielzeiger +1
0446  05        DEC   B                  ;Zähler -1
0447  C2 0442   JP    NZ,move_b_bytes    ;Weiter, bis Zähler -0
044A  C9        RET
;
;
; UPRO für DIR
; Einzelnes Byte aus dem $buffer holen
;
; I: A+C = relative Position des gewünschten Bytes innerhalb
;          des $buffer
;
; O: A   = Byte an der Position  ($buffer + A + C)
;
get_byte_from_buffer:
;
044B  21 0080   LD    HL,$buffer         ;HL -> $buffer
044E  81        ADD   A,C                ;A = A+C
044F  CD 0259   CALL  add_hl_a           ;HL = HL + A
0452  7E        LD    A,(HL)             ;Byte holen
0453  C9        RET
```

```
;
; Neue Disk selektieren, falls es nicht die aktuelle
; CCP-Disk ist.
;
select_new_disk:
;
0454   AF         XOR    A                    ;A = 0
0455   32 07CD    LD     (internal_fcb),A     ;Diskcode des FCB auf
                                               ;Null setzen, da die
                                               ;CCP-Disk jetzt ange-
                                               ;wählt wird und damit
                                               ;zur aktuellen Disk
                                               ;wird
0458   3A 07F0    LD     A,(file_disk_no)     ;Wurde wirklich eine
045B   B7         OR     A                    ;neue Disk verlangt ?
045C   C8         RET    Z                    ;N: Fertig
045D   3D         DEC    A                    ;J: Diskcode in Disk-
                                               ;nummer umwandeln
045E   21 07EF    LD     HL,ccp_disk          ;und mit der aktuellen
0461   BE         CP     (HL)                 ;CCP-Disk vergleichen.
                                               ;Ist die neue Disk
                                               ;gleich der CCP-Disk ?
0462   C8         RET    Z                    ;J: Fertig
0463   C3 00BD    JP     select_disk          ;N: Neue Disk selek-
                                               ;tieren
;
;
; Nach Anwahl einer neuen Disk (über select_new_disk)
; wieder die alte Disk selektieren.
;
select_old_disk:
;
0466   3A 07F0    LD     A,(file_disk_no)     ;Wurde eine neue Disk
0469   B7         OR     A                    ;angewählt ?
046A   C8         RET    Z                    ;N: Fertig
046B   3D         DEC    A                    ;J: Diskcode in Disk-
                                               ;nummer umrechnen
046C   21 07EF    LD     HL,ccp_disk          ;Ist die neu ange-
046F   BE         CP     (HL)                 ;wählte Disk gleich
                                               ;der CCP-Disk ?
0470   C8         RET    Z                    ;J: Fertig
0471   3A 07EF    LD     A,(ccp_disk)         ;N: CCP-Disk wieder
0474   C3 00BD    JP     select_disk          ;selektieren
```

```
;
; **** DIR
;
; Directory ausgeben
;
dir:
;
0477  CD 025E  CALL  setup_fcb          ;Die eventuell dem
                                        ;DIR-Befehl folgende
                                        ;Disk- und/oder File-
                                        ;bezeichnung in den
                                        ;internal_fcb über-
                                        ;nehmen
047A  CD 0454  CALL  select_new_disk    ;Neue Disk, falls an-
                                        ;gegeben, selektieren
047D  21 07CE  LD    HL,internal_filename
                                        ;HL -> Nach 'DIR' fol-
                                        ;gende Filebezeichnung
0480  7E       LD    A,(HL)             ;Erstes Zeichen dieser
                                        ;Filebezeichnung holen
0481  FE 20    CP    space              ;Wurde eine Filebe-
                                        ;zeichnung angegeben ?
0483  C2 048F  JP    NZ,dir_file        ;J: Zum internal_fcb
                                        ;passende Directory-
                                        ;Einträge anzeigen
                                        ;N: 'DIR' ohne File-
                                        ;bezeichnung ent-
                                        ;spricht einem
                                        ; ' DIR *.* '.
                                        ;internal_fcb ent-
                                        ;sprechend vorbe-
                                        ;reiten
;
; Filenamen und Filetyp im internal_fcb mit '?' füllen
;
0486  06 0B    LD    B,11               ;8 Zeichen Filenamen
                                        ;und 3 Zeichen File-
                                        ;typ mit '?' füllen
;
set_wildcard_loop:
;
0488  36 3F    LD    (HL),'?'           ;'?' einsetzen
048A  23       INC   HL                 ;FCB-Zeiger +1
048B  05       DEC   B                  ;Zähler -1
                                        ;Weiter, bis Zähler =0
048C  C2 0488  JP    NZ,set_wildcard_loop
```

```
;
; Alle auf den internal_fcb passenden Directory-Einträge
; anzeigen.
;
dir_file:
;
048F  1E 00     LD    E,0               ;Spaltenzähler auf 0
                                        ;setzen
0491  D5        PUSH  DE                ;Spaltenzähler retten
0492  CD 00E9   CALL  search_fcb         ;Ersten passenden Dir-
                                        ;ectory suchen
0495  CC 03EA   CALL  Z,no_file         ;Nichts gefunden ?
                                        ;J: 'NO FILE' melden
;
;
; Nächsten passenden Directory-Eintrag anzeigen
;
; I: Flags vom letzten search_next gesetzt
;
dir_loop:
;
                                        ;Wurde ein nächster
                                        ;Eintrag gefunden ?
0498  CA 051B   JP    Z,dir_exit        ;N: Fertig
049B  3A 07EE   LD    A,(return_code)   ;Directory-Code vom
                                        ;letzten 'search_next'
                                        ;holen
049E  0F        RRCA                    ;Dreimal nach rechts
049F  0F        RRCA                    ;rotiert, entspricht
04A0  0F        RRCA                    ;einer Multiplikation
                                        ;mit 32. Dies ist dann
                                        ;die relative Position
                                        ;des gefundenen Ein-
                                        ;trags im Sbuffer
04A1  E6 60     AND   060H              ;untere 5 Bits löschen
04A3  4F        LD    C,A               ;C = Position des Ein-
                                        ;trags
04A4  3E 0A     LD    A,10              ;A = relative Position
                                        ;des 'SYS'-Flags in-
                                        ;nerhalb des Eintrags.
                                        ;'SYS'-Flag holen
04A6  CD 044B   CALL  get_byte_from_buffer
04A9  17        RLA                     ;'SYS'-Flag ins Carry-
                                        ;Flag schieben.
                                        ;'SYS'-Flag gesetzt ?
04AA  DA 050F   JP    C,entry_is_sys    ;J: 'SYS'-Files werden
                                        ;nicht angezeigt
```

```
04AD  D1        POP   DE                ;Spaltenzähler zurück
04AE  7B        LD    A,E               ;A = Spaltenzähler
04AF  1C        INC   E                 ;Spaltenzähler +1
04B0  D5        PUSH  DE                ;Spaltenzähler retten
04B1  E6 03     AND   3                 ;A = Spaltenzähler
                                        ;modulo 4
04B3  F5        PUSH  AF                ;A retten
                                        ;Spaltenzähler modulo
                                        ;4 gleich Null ?
                                        ;N: Gefundenen Eintrag
                                        ;innerhalb der selben
                                        ;Zeile ausgeben
04B4  C2 04CC   JP    NZ,dir_next_in_line
                                        ;J: Es werden nur 4
                                        ;Einträge pro Zeile
                                        ;angezeigt.
04B7  CD 0098   CALL  print_crlf        ;Neue Ausgabezeile
                                        ;beginneen
04BA  C5        PUSH  BC                ;Eintrags-Position
                                        ;retten
04BB  CD 01D0   CALL  get_disk_code     ;Aktuelle Disknummer
                                        ;holen
04BE  C1        POP   BC                ;Eintrags-Position
                                        ;zurück
04BF  C6 41     ADD   A,'A'             ;Disknummer + 'A' er-
                                        ;gibt Diskbezeichnung
                                        ;('A' - 'P')
04C1  CD 0092   CALL  conout_bc         ;Diskbezeichnung aus-
                                        ;geben, BC dabei ret-
                                        ;ten
04C4  3E 3A     LD    A,':'             ;Doppelpunkt danach
04C6  CD 0092   CALL  conout_bc         ;ausgeben
04C9  C3 04D4   JP    show_file         ;Filenamen anzeigen
;
;
; Gefundenen Eintrag innerhalb der selben Zeile anzeigen
;
dir_next_in_line:
;
04CC  CD 00A2   CALL  print_space       ;Leerzeichen
04CF  3E 3A     LD    A,':'             ;und Doppelpunkt als
04D1  CD 0092   CALL  conout_bc         ;Trennung ausgeben
```

```
;
; Filenamen und Filetyp des gefundenen Eintrags anzeigen
;
show_file:
;
04D4  CD 00A2  CALL  print_space          ;Leerzeichen voran-
                                          ;stellen
04D7  06 01    LD    B,1                  ;B ist die relative
                                          ;Position des nächsten
                                          ;auszugebenden Zei-
                                          ;chens innerhalb des
                                          ;Eintrags. (Position
                                          ;0 des Eintrags ist
                                          ;die Usernummer !)
;
; Nächstes Zeichen des Filenamens ausgeben
;
; C ist immer noch die relative Position des Eintrags inner-
; halb des §buffers
;
dir_next_char:
;
04D9  78       LD    A,B                  ;A = relative Zeichen-
                                          ;position
                                          ;Nächstes Zeichen aus
                                          ;dem Buffer holen
04DA  CD 044B  CALL  get_byte_from_buffer
04DD  E6 7F    AND   07FH                 ;Bit 7 ausmaskieren
04DF  FE 20    CP    space                ;Leerzeichen ?
04E1  C2 04F9  JP    NZ,dir_print_char    ;N: Zeichen normal
                                          ;ausgeben
04E4  F1       POP   AF                   ;Spaltenzähler mod 4
04E5  F5       PUSH  AF                   ;zurück und wieder
                                          ;retten
04E6  FE 03    CP    3                    ;Sind wir in der letz-
                                          ;ten Spalte ?
04E8  C2 04F7  JP    NZ,dir_print_space   ;N: Leerzeichen normal
                                          ;ausgeben
04EB  3E 09    LD    A,9                  ;J: Erstes Zeichen vom
                                          ;Filetyp holen
04ED  CD 044B  CALL  get_byte_from_buffer
04F0  E6 7F    AND   07FH                 ;Bit 7 löschen
04F2  FE 20    CP    space                ;Ist es ein Leer-
                                          ;zeichen ?
                                          ;J: Dann existiert
                                          ;kein Filetyp
                                          ;Nächsten Eintrag
                                          ;suchen
04F4  CA 050E  JP    Z,get_next_dir_entry
```

```
;
; Leerzeichen ausgeben
;
dir_print_space:
;
04F7  3E 20     LD     A,space            ;A - Leerzeichen
;
;
; Zeichen des Filenamens ausgeben
;
dir_print_char:
;
04F9  CD 0092   CALL   conout_bc          ;Zeichen ausgeben
04FC  04        INC    B                  ;Zeichenposition +1
04FD  78        LD     A,B                ;A - Neue Zeichen-
                                          ;position
04FE  FE 0C     CP     12                 ;Ist die neue Position
                                          ;kleiner als 12 ?
                                          ;N: Filenamen komplett
                                          ;ausgegeben
                                          ;Nächsten Filenamen
                                          ;suchen
0500  D2 050E   JP     NC,get_next_dir_entry
0503  FE 09     CP     9                  ;Ist die neue Position
                                          ;das erste Zeichen des
                                          ;Filetyps ?
0505  C2 04D9   JP     NZ,dir_next_char   ;N: nächstes Zeichen
                                          ;ausgeben
0508  CD 00A2   CALL   print_space        ;J: Leerzeichen als
                                          ;Trennung zwischen
                                          ;Filenamen und File-
                                          ;typ ausgeben
050B  C3 04D9   JP     dir_next_char      ;Nächstes Zeichen aus-
                                          ;geben
;
;
; Nächsten passenden Eintrag suchen und anzeigen
;
get_next_dir_entry:
;
050E  F1        POP    AF                 ;Spaltenzähler vom
                                          ;Stack nehmen
```

```
;
; Einsprung, falls ein Eintrag mit gesetztem 'SYS'-Flag
; gefunden wurde
;
entry_is_sys:
;
050F  CD 01C2  CALL  check_console     ;Steht ein Zeichen von
                                       ;der Console an ?
                                       ;(Wurde eine Taste
                                       ;gedrückt ?)
0512  C2 051B  JP    NZ,dir_exit      ;J: DIR abbrechen
0515  CD 00E4  CALL  search_next       ;N: Nächsten passenden
                                       ;Eintrag suchen
0518  C3 0498  JP    dir_loop         ;und anzeigen
;
;
; DIR beenden
;
dir_exit:
;
051B  D1       POP   DE               ;Spaltenzähler vom
                                       ;Stack nehmen
051C  C3 0786  JP    reenter_ccp      ;Zurück zum CCP
```

```
;
; **** ERA
;
; File(s) löschen
;
era:
;
051F  CD 025E  CALL  setup_fcb          ;Angegebenen Filenamen
                                        ;übernehmen
0522  FE 0B    CP    11                 ;11 Fragezeichen ge-
                                        ;funden (ERA *.*) ?
0524  C2 0542  JP    NZ,era_not_all     ;N: Es sollen nicht
                                        ;alle Files gelöscht
                                        ;werden
                                        ;J: Nochmal nachfragen
0527  01 0552  LD    BC,txt_all_yn      ;BC -> 'ALL (Y/N)?'
052A  CD 00A7  CALL  print_text         ;Text ausgeben
052D  CD 0139  CALL  get_command_line   ;Antwort als komplette
                                        ;Zeile holen
0530  21 0007  LD    HL,$cmd_buffer+1   ;HL -> Anzahl der ein-
                                        ;gegebenen Zeichen
0533  35       DEC   (HL)               ;-1 gleich Null ?
0534  C2 0382  JP    NZ,ccp_prompt      ;N: Es darf nur ein
                                        ;Zeichen ('Y' oder
                                        ;'N') eingegeben wer-
                                        ;den; zurück zum CCP-
                                        ;Prompt
0537  23       INC   HL                 ;J: HL -> Zeichen
0538  7E       LD    A,(HL)             ;Zeichen holen
0539  FE 59    CP    'Y'                ;Ist es 'Y'
053B  C2 0382  JP    NZ,ccp_prompt      ;N: Alles andere wird
                                        ;als 'N' interpretiert
053E  23       INC   HL                 ;Bufferzeiger erhöhen
                                        ;und abspeichern
053F  22 0088  LD    (?buffer_adress),HL
;
;
era_not_all:
;
0542  CD 0454  CALL  select_new_disk    ;Evtl. angegebene Disk
                                        ;selektieren
0545  11 07CD  LD    DE,internal_fcb    ;DE -> FCB
0548  CD 00EF  CALL  delete_file        ;File(s) löschen
054B  3C       INC   A                  ;Fehlercode = 255 ?
054C  CC 03EA  CALL  Z,no_file          ;J: 'NO FILE' melden
054F  C3 0786  JP    reenter_ccp        ;zurück zum CCP
;
; Text für ERA
;
txt_all_yn:
;
0552  41       defm  'ALL (Y/N)?'
055C  00       defb  0
```

```
;
; **** TYPE
;
; (ASCII-)File ausgeben
;
type:
;
055D  CD 025E  CALL  setup_fcb          ;Angegebenen Filenamen
                                        ;übernehmen
0560  C2 0209  JP    NZ,command_error  ;Fehler, falls der
                                        ;Filenamen nicht ein-
                                        ;deutig ist
0563  CD 0454  CALL  select_new_disk   ;Angegebene Disk
                                        ;selektieren
0566  CD 00D0  CALL  open_fcb          ;File öffnen
                                        ;File vorhanden ?
0569  CA 05A7  JP    Z,type_error      ;N: Fehler
056C  CD 0098  CALL  print_crlf        ;J: Neue Ausgabezeile
                                        ;beginnen
                                        ;HL -> Zeichenzähler
056F  21 07F1  LD    HL,type_char_counter
0572  36 FF    LD    (HL),255          ;Zeichenzähler mit
                                        ;255 vorbesetzen
;
;
; Nächstes Zeichen lesen und ausgeben
;
type_loop:
;
                                        ;HL -> Anzahl der vom
                                        ;aktuellen Record be-
                                        ;reits ausgegebenen
                                        ;Zeichen
0574  21 07F1  LD    HL,type_char_counter
0577  7E        LD    A,(HL)            ;A = Anzahl der ausge-
                                        ;gebenen Zeichen
0578  FE 80     CP    128               ;schon 128 Zeichen
                                        ;ausgegeben ?
057A  DA 0587   JP    C,type_char       ;N: Nächstes Zeichen
                                        ;aus dem Buffer holen
                                        ;J: Der aktuelle Re-
                                        ;cord wurde komplett
                                        ;ausgegeben. Nächsten
                                        ;Record des Files in
                                        ;den Buffer laden.
057D  E5        PUSH  HL                ;HL retten
057E  CD 00FE   CALL  read_fcb          ;Nächsten Record lesen
0581  E1        POP   HL                ;HL zurück
0582  C2 05A0   JP    NZ,type_exit      ;Type abbrechen, falls
                                        ;ein Lesefehler aufge-
                                        ;treten ist.
```

```
0585  AF        XOR   A                 ;A = 0
0586  77        LD    (HL),A            ;Vom neu gelesenen Re-
                                        ;cord wurde noch kein
                                        ;Zeichen ausgegeben.
;
; Nächstes Zeichen vom Buffer holen und ausgeben
; A = relative Position des nächsten Zeichens
;     innerhalb des Buffers
;
type_char:
;
0587  34        INC   (HL)              ;Zeichenzähler +1
0588  21 0080   LD    HL,$buffer        ;HL -> Buffer
058B  CD 0259   CALL  add_hl_a          ;HL -> Zeichen
058E  7E        LD    A,(HL)            ;Zeichen holen
058F  FE 1A     CP    eof               ;'End Of File' er-
                                        ;reicht?
0591  CA 0786   JP    Z,reenter_ccp     ;J: Type beenden
0594  CD 008C   CALL  conout            ;N: Zeichen ausgeben
0597  CD 01C2   CALL  check_console      ;Steht ein Zeichen
                                        ;von der Console an ?
                                        ;(Wurde eine Taste
                                        ;gedrückt ?)
059A  C2 0786   JP    NZ,reenter_ccp    ;J: Type abbrechen
059D  C3 0574   JP    type_loop         ;N: Nächstes Zeichen
                                        ;ausgeben
;
;
; TYPE-Abbruch nach Lesefehler
;
type_exit:
;
05A0  3D        DEC   A                 ;Fehlercode = 1 ?
05A1  CA 0786   JP    Z,reenter_ccp     ;J: Ende des Files
                                        ;erreicht, es war kein
                                        ;'echter' Lesefehler
05A4  CD 03D9   CALL  read_error        ;N: 'READ ERROR' mel-
                                        ;den
;
;
; TYPE-Abbruch bei nicht eindeutigem Filenamen bzw.
; fehlerhafter Parameterangabe
;
type_error:
;
05A7  CD 0466   CALL  select_old_disk   ;Alte Disk wieder an-
                                        ;wählen und
05AA  C3 0209   JP    command_error     ;fehlerhaften Para-
                                        ;meter ausgeben
```

```
;
; **** SAVE
;
; Speicherbereich auf Disk schreiben
;
save:
;
05AD  CD 03F8  CALL  get_number        ;Anzahl der zu spei-
                                       ;chernden Seiten aus-
                                       ;werten
05B0  F5       PUSH  AF                ;Anzahl retten
05B1  CD 025E  CALL  setup_fcb         ;Filenamen in den in-
                                       ;ternal_fcb übernehm-
                                       ;men
05B4  C2 0209  JP    NZ,command_error  ;Fehler, falls der
                                       ;Filenamen nicht ein-
                                       ;deutig ist.
05B7  CD 0454  CALL  select_new_disk   ;Disk selektieren
05BA  11 07CD  LD    DE,internal_fcb   ;DE -> FCB
05BD  D5       PUSH  DE                ;FCB-Zeiger retten
05BE  CD 00EF  CALL  delete_file       ;Evtl. vorhandenes
                                       ;File gleichen Namens
                                       ;vorher löschen.
05C1  D1       POP   DE                ;FCB-Zeiger zurück
05C2  CD 0109  CALL  make_file         ;File neu anlegen
05C5  CA 05FB  JP    Z,no_space        ;'NO SPACE' melden,
                                       ;falls die Directory
                                       ;voll ist
05C8  AF       XOR   A                 ;A = 0
                                       ;Recordnummer auf Null
                                       ;setzen
05C9  32 07ED  LD    (internal_record_#),A

05CC  F1       POP   AF                ;Anzahl der zu spei-
                                       ;chernden Seiten zu-
                                       ;rück
05CD  6F       LD    L,A               ;HL = Anzahl
05CE  26 00    LD    H,0
05D0  29       ADD   HL,HL             ;Anzahl der Records
                                       ;berechnen (Eine Sei-
                                       ;te im Speicher ent-
                                       ;spricht zwei Records)
05D1  11 0100  LD    DE,?tpa_start     ;DE -> TPA
```

```
;
; Nächsten Record schreiben
; DE -> Speicherbereich
; HL = Anzahl der verbleibenden Records
;
save_loop:
;
05D4  7C       LD    A,H                ;Verbleibende Record-
05D5  B5       OR    L                  ;anzahl = 0 ?
05D6  CA 05F1  JP    Z,close_save_file  ;J: Fertig, File
                                        ;schliessen
05D9  2B       DEC   HL                 ;N: Anzahl -1
05DA  E5       PUSH  HL                 ;Anzahl retten
05DB  21 0080  LD    HL,128             ;HL = Länge eines
                                        ;Records in Byte
05DE  19       ADD   HL,DE              ;Nächste (!) Speicher-
                                        ;adresse berechnen
05DF  E5       PUSH  HL                 ;und retten
05E0  CD 01D8  CALL  set_dma            ;Adresse des jetzt zu
                                        ;schreibenden Records
                                        ;setzen (in DE !)
05E3  11 07CD  LD    DE,internal_fcb    ;DE -> FCB
05E6  CD 0104  CALL  write_sequ         ;Record schreiben
05E9  D1       POP   DE                 ;Neue Adresse vom
                                        ;Stack holen
05EA  E1       POP   HL                 ;Verbleibende Anzahl
                                        ;zurück
05EB  C2 05FB  JP    NZ,no_space        ;'NO SPACE' melden,
                                        ;falls beim Schreiben
                                        ;ein Fehler aufgetre-
                                        ;ten ist
05EE  C3 05D4  JP    save_loop          ;Sonst nächsten Record
                                        ;schreiben
;
;
; File schließen
;
close_save_file:
;
05F1  11 07CD  LD    DE,internal_fcb    ;DE -> FCB
05F4  CD 00DA  CALL  close_file         ;File schließen
05F7  3C       INC   A                  ;Fehler beim
                                        ;Schließen ?
05F8  C2 0601  JP    NZ,save_exit       ;N: Save beenden
```

```
;
; Fehlermeldung 'NO SPACE' ausgeben
;
no_space:
;
05FB  01 0607  LD    BC,txt_no_space   ;BC -> Text
05FE  CD 00A7  CALL  print_text        ;Text ausgeben
;
;
; Save beenden
;
save_exit:
;
0601  CD 01D5  CALL  set_default_dma   ;DMA-Adresse auf Stan-
                                       ;dardwert zurücksetzen
0604  C3 0786  JP    reenter_ccp      ;Zurück zum CCP
;
;
; Text für SAVE
;
txt_no_space:
;
0607  4E       defm  'NO SPACE'
060F  00       defb  0
```

```
;
; **** REN
;
; Filenamen ändern
;
ren:
;
0610   CD 025E   CALL   setup_fcb           ;Neuen Filenamen über-
                                            ;nehmen
0613   C2 0209   JP     NZ,command_error    ;Fehler, falls der
                                            ;Filenamen nicht ein-
                                            ;deutig ist
0616   3A 07F0   LD     A,(file_disk_no)    ;Diskcode holen
0619   F5        PUSH   AF                  ;und retten
061A   CD 0454   CALL   select_new_disk     ;Disk selektieren
061D   CD 00E9   CALL   search_fcb          ;und Directory auf
                                            ;neuen Filenamen über-
                                            ;prüfen. Ist der neue
                                            ;Filename bereits vor-
                                            ;handen ?
0620   C2 0679   JP     NZ,file_exists      ;J: 'FILE EXISTS' mel-
                                            ;den
0623   21 07CD   LD     HL,internal_fcb     ;N: Neuen Filenamen
0626   11 07DD   LD     DE,rename_fcb       ;nach rename_fcb ko-
                                            ;pieren (für BDOS-
                                            ;Funktion 'rename')
0629   06 10     LD     B,16                ;16 Bytes
062B   CD 0442   CALL   move_b_bytes        ;kopieren
                                            ;HL -> nächstes Buf-
                                            ;ferzeichen
062E   2A 0088   LD     HL,(?buffer_adress)
0631   EB        EX     DE,HL               ;DE = HL
0632   CD 024F   CALL   get_next_char       ;Nächstes Zeichen
                                            ;holen
0635   FE 3D     CP     '='                 ;Ist es '=' ?
0637   CA 063F   JP     Z,ren_delimiter_ok  ;J: Trennungszeichen
                                            ;in Ordnung
063A   FE 5F     CP     '_'                 ;Ist es '_' ?
                                            ;(Dieses Zeichen ist
                                            ;bei manchen Computern
                                            ;ein Pfeil nach links)
                                            ;N: Fehler
063C   C2 0673   JP     NZ,ren_file_not_found
```

```
;
; Trennungszeichen zwischen dem neuen und alten Filenamen
; ist in Ordnung
;
ren_delimiter_ok:
;
063F  EB        EX    DE,HL              ;HL -> Trennungs-
                                         ;zeichen
0640  23        INC   HL                 ;HL -> nächstes Zei-
                                         ;chen
                                         ;Zeiger auf das
                                         ;nächste Zeichen,
                                         ;für den folgenden
                                         ;'setup_fcb'-Aufruf
                                         ;speichern.
0641  22 0088   LD    (?buffer_adress),HL

0644  CD 025E   CALL  setup_fcb          ;Alten Filenamen über-
                                         ;nehmen
                                         ;Fehler, falls der
                                         ;alte Filenamen nicht
                                         ;eindeutig ist.
0647  C2 0673   JP    NZ,ren_file_not_found
064A  F1        POP   AF                 ;Diskcode des neuen
                                         ;Filenamens zurück
064B  47        LD    B,A                ;und in B speichern
064C  21 07F0   LD    HL,file_disk_no    ;HL -> Diskcode
                                         ;des alten Filenamens
064F  7E        LD    A,(HL)             ;Diskcode holen.
0650  B7        OR    A                  ;Wurde beim alten
                                         ;Filenamens ein Disk-
                                         ;code angegeben ?
0651  CA 0659   JP    Z,ren_disk_ok      ;N: Diskcode des neuen
                                         ;Filenamens auch für
                                         ;den alten Filenamen
                                         ;benutzen.
0654  B8        CP    B                  ;J: Sind beide Disk-
                                         ;codes identisch ?

0655  70        LD    (HL),B             ;Diskcode des neuen
                                         ;Filenamens schonmal
                                         ;speichern.
                                         ;Fehler, falls die
                                         ;beiden Filenamen un-
                                         ;terschiedliche Disk-
                                         ;codes haben.
0656  C2 0673   JP    NZ,ren_file_not_found
```

```
;
; Die Diskcodes der beiden Filenamen sind verträglich.
; Diskcode des neuen Filenamen speichern.
;
ren_disk_ok:
;
0659  70        LD    (HL),B              ;Diskcode speichern
065A  AF        XOR   A                   ;A = 0
065B  32 07CD   LD    (internal_fcb),A    ;Diskcode des alten
                                          ;Files auf Null setzen
                                          ;(Die richtige Disk
                                          ;wurde schon selek-
                                          ;tiert !)
065E  CD 00E9   CALL  search_fcb          ;Alter Filenamen vor-
                                          ;handen ?
0661  CA 066D   JP    Z,ren_no_file       ;N: 'NO FILE' melden

0664  11 07CD   LD    DE,internal_fcb     ;J: DE -> FCB
0667  CD 010E   CALL  rename_file         ;BDOS-Aufruf
066A  C3 0786   JP    reenter_ccp         ;Zurück zum CCP
;
;
; 'NO FILE' melden
;
ren_no_file:
;
066D  CD 03EA   CALL  no_file             ;'NO FILE' ausgeben
0670  C3 0786   JP    reenter_ccp         ;Zurück zum CCP
;
;
; Fehler: Alter Filename nicht vorhanden
;
ren_file_not_found:
;
0673  CD 0466   CALL  select_old_disk     ;Alte Disk wieder
                                          ;selektieren
0676  C3 0209   JP    command_error       ;und Fehler melden
;
;
; Fehler: Neuer Filename bereits vorhanden
;
file_exists:
;
0679  01 0682   LD    BC,txt_file_exists;BC -> Text
067C  CD 00A7   CALL  print_text          ;Text ausgeben
067F  C3 0786   JP    reenter_ccp         ;Zurück zum CCP
;
txt_file_exists:
;
0682  46        defm  'FILE EXISTS'
068D  00        defb  0
```

```
;
; **** USER
;
; Neuen Userbereich anwählen
;
user:
;
068E   CD 03F8   CALL   get_number          ;Neue Usernummer holen
0691   FE 10     CP     16                  ;Nummer > 15 ?
0693   D2 0209   JP     NC,command_error    ;J: Fehler
0696   5F        LD     E,A                 ;E - Usernummer
                                           ;Wurde wirklich eine
                                           ;Nummer angegeben ?
                                           ;J: Dann steht sie im
                                           ;'internal_filename'
0697   3A 07CE   LD     A,(internal_filename)
069A   FE 20     CP     space               ;Steht dort ein Leer-
                                           ;zeichen ?
069C   CA 0209   JP     Z,command_error     ;J: Keine Nummer an-
                                           ;gegeben, Fehler.
069F   CD 0115   CALL   set_user            ;N: Neue Usernummer
                                           ;setzen
06A2   C3 0789   JP     reenter_new_disk    ;Zurück zum CCP ohne
                                           ;Disk-Selektierung
                                           ;(es wurde ja auch
                                           ;keine neue Disk
                                           ;angewählt)
```

```
;
; Der eingegebene Befehl ist keiner der eingebauten
; CCP-Befehle.
; Befehl als Filename eine Programms interpretieren und
; dieses Programm laden und ausführen.
;
execute_program:
;
06A5  CD 01F5  CALL  compare_serial     ;Seriennummern des CCP
                                        ;und BDOS miteinander
                                        ;vergleichen und bei
                                        ;Ungleichheit 'System-
                                        ;absturz' auslösen.
                                        ;Wurde ein Filenamen
                                        ;angegeben ?
06A8  3A 07CE  LD    A,(internal_filename)
06AB  FE 20    CP    space              ;Ist also das erste
                                        ;Zeichen des 'inter-
                                        ;nal_filename' kein
                                        ;Leerzeichen ?
                                        ;J: Programm laden und
                                        ;ausführen
06AD  C2 06C4  JP    NZ,load_and_execute
;
;
; Der Befehl ist kein Filename.
; Wurde nur eine Diskbezeichnung angegeben ?
;
06B0  3A 07F0  LD    A,(file_disk_no)  ;Ist eine Diskbe-
06B3  B7       OR    A                 ;zeichnung erkannt
                                       ;worden ?
06B4  CA 0789  JP    Z,reenter_new_disk;N: Zurück zum CCP
;
;
; Der Befehl ist eine Diskbezeichnung.
; Neue 'ccp_disk' anwählen.
;
06B7  3D       DEC   A                 ;J: Diskcode in Disk-
                                       ;nummer umwandeln
06B8  32 07EF  LD    (ccp_disk),A      ;und als neue CCP-Disk
                                       ;speichern
06BB  CD 0129  CALL  set_disk          ;Neue Disknummer in
                                       ;Sdisk speichern
06BE  CD 00BD  CALL  select_disk       ;und neue Disk selek-
                                       ;tieren
06C1  C3 0789  JP    reenter_new_disk  ;Zurück zum CCP
```

```
;
; Als Befehl wurde weder einer der CCP-Befehle noch eine
; Diskbezeichnung angegeben.
; Es muß ein Filename sein.
;
load_and_execute:
;
                                                ;DE -> Filetyp des
                                                ;angegebenen Files
06C4  11 07D6  LD    DE,internal_filetype
06C7  1A       LD    A,(DE)                     ;Wurde ein Filetyp
                                                ;mit angegeben ?
06C8  FE 20    CP    space                      ;Ist also das erste
                                                ;Zeichen des File-
                                                ;typs kein Leerzei-
                                                ;chen ?
06CA  C2 0209  JP    NZ,command_error           ;J: Fehler, der File-
                                                ;typ wird immer vom
                                                ;CCP eingetragen

06CD  D5       PUSH  DE                         ;Filetyp-Zeiger retten
06CE  CD 0454  CALL  select_new_disk            ;Disk selektieren
06D1  D1       POP   DE                         ;Zeiger zurück

06D2  21 0783  LD    HL,default_typ             ;HL -> Filetyp 'COM'
06D5  CD 0440  CALL  move_3_bytes               ;'COM' als Filetyp
                                                ;einsetzen
06D8  CD 00D0  CALL  open_fcb                   ;File öffnen. Ist das
                                                ;File vorhanden ?

                                                ;N: Alte Disk wieder
                                                ;anwählen und Fehler
                                                ;melden
06DB  CA 076B  JP    Z,transient_not_found
;
;
; Programm laden und ausführen
;
06DE  21 0100  LD    HL,?tpa_start              ;Programme werden im-
                                                ;mer ab dem '?tpa_
                                                ;start' geladen
```

```
;
; Nächsten Record des Programms laden.
; HL = Ladeadresse
;
load_next_record:
;
06E1   E5        PUSH   HL                  ;Ladeadresse retten
06E2   EB        EX     DE,HL                ;DE = Ladeadresse
06E3   CD 01D8   CALL   set_dma              ;DMA-Adresse = Lade-
                                             ;adresse setzen
06E6   11 07CD   LD     DE,internal_fcb      ;DE -> FCB
06E9   CD 00F9   CALL   read_sequ            ;Nächsten Record zur
                                             ;DMA-Adresse laden.
                                             ;Fehler ? (Lesefehler
                                             ;oder Ende des
                                             ;Files ?)
                                             ;J: Ende des Ladevor-
                                             ;gangs
06EC   C2 0701   JP     NZ,setup_parameters
06EF   E1        POP    HL                   ;N: Ladeadresse zurück
06F0   11 0080   LD     DE,128               ;DE = Länge eines Re-
                                             ;cords
06F3   19        ADD    HL,DE                ;Neue Ladeadresse be-
                                             ;rechnen
06F4   11 0000   LD     DE,ccp               ;DE = CCP-Startadresse
                                             ;Würde das Laden des
                                             ;nächsten Records den
                                             ;CCP überschreiben ?
06F7   7D        LD     A,L                  ;Ist also die neue
06F8   93        SUB    E                    ;Ladeadresse größer
06F9   7C        LD     A,H                  ;oder gleich der CCP-
06FA   9A        SBC    A,D                  ;Startadresse ?
06FB   D2 0771   JP     NC,bad_load          ;J: 'BAD LOAD' melden

06FE   C3 06E1   JP     load_next_record     ;N: Nächsten Record
                                             ;laden
;
;
; Beim Lesen des Records ist ein Fehler aufgetreten
;
setup_parameters:
;
0701   E1        POP    HL                   ;Ladeadresse vom Stack
                                             ;nehmen
0702   3D        DEC    A                    ;Fehlercode = 1 ?
                                             ;(Ende des Files er-
                                             ;reicht ?)
0703   C2 0771   JP     NZ,bad_load          ;N: 'BAD LOAD' melden
```

```
;
; Das Programm wurde vollständig geladen.
; Jetzt müssen noch die eventuell vorhandenen Parameter
; aufgearbeitet werden.
;
0706  CD 0466  CALL  select_old_disk    ;Alte Disk wieder an-
                                        ;wählen

0709  CD 025E  CALL  setup_fcb          ;Parameter in den in-
                                        ;ternal_fcb übernehmen
070C  21 07F0  LD    HL,file_disk_no   ;HL -> Diskcode
070F  E5       PUSH  HL                ;Zeiger retten
0710  7E       LD    A,(HL)            ;Diskcode in den
0711  32 07CD  LD    (internal_fcb),A  ;internal_fcb über-
                                        ;nehmen
0714  3E 10    LD    A,16              ;Zweiten Parameter
                                        ;nach internal_fcb+16
0716  CD 0260  CALL  setup_fcb_offset  ;übernehmen
0719  E1       POP   HL                ;Zeiger auf Diskcode
                                        ;zurück
071A  7E       LD    A,(HL)            ;Diskcode des zweiten
                                        ;Parameters
071B  32 07DD  LD    (rename_fcb),A    ;im dazugehörigen fcb
                                        ;speichern
071E  AF       XOR   A                 ;A = 0
                                        ;Recordnummer des
                                        ;zweiten fcbs auf
                                        ;Null setzen
071F  32 07ED  LD    (internal_record_#),A
                                        ;Die beiden Parameter
                                        ;werden dem Programm
                                        ;im '?default_fcb' zur
                                        ;Verfügung gestellt
0722  11 005C  LD    DE,?default_fcb   ;Zum '?default_fcb'
0725  21 07CD  LD    HL,internal_fcb   ;werden vom 'inter-
                                        ;nal_fcb'
0728  06 21    LD    B,33              ;33 Bytes
072A  CD 0442  CALL  move_b_bytes      ;kopiert
```

```
;
; Die im §cmd_buffer stehenden Parameter werden dem
; Programm auch noch im §buffer zur Verfügung gestellt.
; Dazu muß zuerst die Anfangsadresse der Parameter im Buffer
; festgestellt werden.
;
072D  21 0008  LD    HL,§cmd_buffer+2  ;HL -> Befehlsbuffer
;
search_parameters:
;
0730  7E       LD    A,(HL)            ;Nächstes Zeichen
                                       ;holen
0731  B7       OR    A                 ;Ende der Zeile er-
                                       ;reicht ?
0732  CA 073E  JP    Z,setup_cmd_buffer;J: Es wurden keine
                                       ;Parameter angegeben
0735  FE 20    CP    space             ;Leerzeichen ?
0737  CA 073E  JP    Z,setup_cmd_buffer;J: Es ist das Ende
                                       ;des Programmnamens
                                       ;und damit der Anfang
                                       ;der Parameter er-
                                       ;reicht.
073A  23       INC   HL                ;N: Zeiger +1
073B  C3 0730  JP    search_parameters ;Nächstes Zeichen
                                       ;überprüfen
;
; Parameter im §buffer zur Verfügung stellen
;
setup_cmd_buffer:
;
073E  06 00    LD    B,0               ;Zeichenzähler auf
                                       ;Null setzen
0740  11 0081  LD    DE,§buffer+1      ;Zeichen ab §buffer+1
                                       ;ablegen
;
setup_cmd_buffer_loop:
;
0743  7E       LD    A,(HL)            ;Zeichen aus Befehls-
                                       ;buffer holen
0744  12       LD    (DE),A            ;und im §buffer ab-
                                       ;speichern
0745  B7       OR    A                 ;Ende der Zeile er-
                                       ;reicht ?
0746  CA 074F  JP    Z,run_transient   ;J: Programm starten

0749  04       INC   B                 ;N: Zeichenzähler +1
074A  23       INC   HL                ;Zeiger +1
074B  13       INC   DE                ;Zeiger +1
                                       ;Nächstes Zeichen
                                       ;übernehmen
074C  C3 0743  JP    setup_cmd_buffer_loop
```

```
;
; Anzahl der Parameterzeichen abspeichern und das geladene
; Programm starten.
;
run_transient:
;
074F  78        LD    A,B               ;A = Anzahl der ko-
                                        ;pierten Zeichen
0750  32 0080   LD    (§buffer),A       ;Diese Anzahl wird
                                        ;am Anfang des §buf-
                                        ;fers gespeichert.
0753  CD 0098   CALL  print_crlf        ;Neue Zeile beginnen
0756  CD 01D5   CALL  set_default_dma   ;DMA-Adresse auf Stan-
                                        ;dardwert setzen
0759  CD 011A   CALL  set_drv_usr       ;Disk- und Usernummer
                                        ;für den nächsten
                                        ;Warmstart in §disk
                                        ;speichern (Damit kann
                                        ;auch das Programm er-
                                        ;kennen, 'woher' es
                                        ;kam)
075C  CD 0100   CALL  ?tpa_start        ;Programm mit 'CALL'
                                        ;starten, so daß eine
                                        ;Rückkehr zum CCP auch
                                        ;mit 'RET' erfolgen
                                        ;kann
;
;
; Ende eines Programms mit 'RET'
;
075F  31 07AB   LD    SP,stack_area     ;Stackpointer neu
                                        ;setzen
0762  CD 0129   CALL  set_disk          ;CCP-Disknummer in
                                        ;§disk speichern
0765  CD 00BD   CALL  select_disk       ;CCP-Disk wieder
                                        ;selektieren
0768  C3 0382   JP    ccp_prompt        ;und zum CCP-Prompt
                                        ;zurückspringen
;
;
; Fehler: Gewünschtes Programm nicht gefunden
;
transient_not_found:
;
076B  CD 0466   CALL  select_old_disk   ;Alte Disk wieder
                                        ;selektieren
076E  C3 0209   JP    command_error     ;und Fehler melden
```

```
;
; Fehler beim Laden eines Programms, 'BAD LOAD' ausgeben.
; (Lesefehler oder Programm zu groß)
;
bad_load:
;
0771  01 077A  LD    BC,txt_bad_load   ;BC -> Text
0774  CD 00A7  CALL  print_text        ;Text ausgeben
0777  C3 0786  JP    reenter_ccp       ;Zurück zum CCP
;
;
txt_bad_load:
;
077A  42       defm  'BAD LOAD'
0782  00       defb  0
;
;
; Vom CCP eingesetzter Filetyp für Anwenderprogramme
;
default_typ:
;
0783  43       defm  'COM'
;
;
; Wiedereintritt in den CCP nach Anwahl einer neuen Disk
;
reenter_ccp:
;
0786  CD 0466  CALL  select_old_disk   ;Alte CCP-Disk wieder
                                       ;anwählen
;
;
; Wiedereintritt in den CCP
;
reenter_new_disk:
;
0789  CD 025E  CALL  setup_fcb         ;Eventuell weitere
                                       ;Parameter übernehmen.
                                       ;Sind noch weitere
                                       ;Parameter vorhanden ?
                                       ;Dann steht entweder
                                       ;im internal_filename
078C  3A 07CE  LD    A,(internal_filename)
078F  D6 20    SUB   space              ;ein von 'Leerzeichen'
                                       ;verschiedenes Zeichen
                                       ;oder
0791  21 07F0  LD    HL,file_disk_no   ;es wurde eine Disk-
0794  B6       OR    (HL)              ;bezeichnung erkannt.
0795  C2 0209  JP    NZ,command_error  ;J: Dies ist ein
                                       ;Fehler.
0798  C3 0382  JP    ccp_prompt        ;N: Zurück zum CCP-
                                       ;Prompt
```

```
;
; Stackbereich für CCP-Funktionen (8 Stufen)
; Da der Stack beim 8080 (bzw. Z80) nach 'unten' wächst,
; steht das Label hinter den Daten.
;
079B  00        defs  16,0
;
stack_area:                                ;Label für den Stack-
                                           ;bereich
;
;
; Parameterbereich für CCP-Funktionen
;
;
; Flag, vom BDOS-Aufruf 'Reset Disk System' kommend, ob auf
; dem Laufwerk A: ein Filenamen existiert, der mit '$' be-
; ginnt.
; Falls dieses Flag gesetzt (-FFH) ist, wird versucht die
; nächste Befehlszeile vom Submit-File  '$$$.SUB' zu lesen.
;
; - FFH, falls ein solcher Filename existiert
; - 00H, sonst
;
submit_flag:
;
07AB  00        defb  0
;
; File Control Block für das Submit-File $$$.SUB
; Da der Diskcode im FCB auf 0 gesetzt ist, wird dieses
; File auf der aktuellen Disk erwartet.
;
submit_fcb:
;
07AC  00        defb  0                    ;Diskcode
;
07AD  24        defm  '$$$     '           ;Filename
;
07B5  53        defm  'SUB'                ;Filetyp
;
07B8  00        defb  0                    ;Extendnummer
;
07B9  00        defb  0
;
submit_fcb_na:
;
07BA  00        defb  0                    ;'Not altered'-Flag
```

```
;
submit_rec_count:
;
07BB  00        defb  0                   ;Anzahl der Records
                                          ;im $$$.SUB-File
;
07BC  00        defs  16,0                ;Blocktabelle
;
current_submit_rec:
;
07CC  00        defb  0                   ;Aktuelle Recordnummer
                                          ;für Leseoperation
;
;
; Interner File Control Block für die Befehls- und
; Parameterverarbeitung
;
internal_fcb:
;
07CD  00        defb  0                   ;Diskcode
;
internal_filename:
;
07CE  00        defs  8,0                 ;Filename
;
internal_filetype:
;
07D6  00        defs  3,0                 ;Filetyp
;
07D9  00        defs  4,0                 ;Extendnummer etc.
;
;
; Start des zweiten FCBs für die 'Rename'-Funktion
; oder Blocktabelle des internal_fcb
;
rename_fcb:
;
07DD  00        defs  16,0
;
internal_record_#:
;
07ED  00        defb  0                   ;Aktuelle Recordnummer
                                          ;zum internal_fcb
```

```
;
; Fehlercode nach BDOS-Aufrufen
;
return_code:
;
07EE  00        defb  0
;
;
; CCP-Disknummer
;
ccp_disk:
;
07EF  00        defb  0
;
;
; Diskcode bei der Parameterverarbeitung
; (siehe setup_fcb)
;
file_disk_no:
;
07F0  00        defb  0
;
;
; Anzahl der bereits ausgegebenen Zeichen eines Records
; während TYPE (siehe 'TYPE')
;
type_char_counter:
;
07F1  00        defb  0
;
;
; Füllbytes, damit der CCP genau 8 Speicherseiten
; (0800H Bytes) belegt
;
07F2  00        defs  14,0
;
;
; Startadresse des BDOS bzw. der BDOS-Seriennummer
; (siehe BDOS-Sourcelisting)
;
bdos:
;
          END
```

```
;
;
;                       BDOS Source-Listing
;
;
;
; Dies sind Equates für die verschiedenen, vom BDOS genutzten
; ASCII-Werte
;
0003  =  ctrlc          equ     3       ;CTRL-C
0005  =  ctrle          equ     5       ;CTRL-E
0007  =  bell           equ     7
0008  =  back           equ     8
0009  =  tab            equ     9
000A  =  lf             equ     10
000D  =  cr             equ     13
0010  =  ctrlp          equ     16      ;CTRL-P
0011  =  xon            equ     17      ;CTRL-Q
0012  =  ctrlr          equ     18      ;CTRL-R
0013  =  xoff           equ     19      ;CTRL-S
0015  =  ctrlu          equ     21      ;CTRL-U
0018  =  ctrlx          equ     24      ;CTRL-X
0020  =  space          equ     ' '
007F  =  del            equ     127
;
;
; Anzahl der vom BDOS unterstützten Funktionen
;
0029  =  no_of_funcs    equ     41
;
;
; BDOS-Versionsnummer
;
0022  =  version_#      equ     22h     ;CP/M 2.2
;
;
;
; Warmstart-Adresse
;
0000  =  ?boot          equ     0000h
;
;
; Adresse des IOBYTE
;
0003  =  @io_byte       equ     ?boot + 0003h
;
;
; Startadresse des Standard-Diskbuffers
;
0080  =  @buffer        equ     ?boot + 0080h
```

```
;
; Startadressen der einzelnen BIOS-Funktionen in der
; BIOS-Sprungleiste
;
; 'b_' am Labelanfang, kennzeichnet das Label als
; BIOS-Adresse
;
0E00  =  b_cboot        equ     bios +  0
0E03  =  b_wboot        equ     bios +  3
0E06  =  b_const        equ     bios +  6
0E09  =  b_conin        equ     bios +  9
0E0C  =  b_conout       equ     bios + 12
0E0F  =  b_list         equ     bios + 15
0E12  =  b_punch        equ     bios + 18
0E15  =  b_reader       equ     bios + 21
0E18  =  b_home         equ     bios + 24
0E1B  =  b_seldsk       equ     bios + 27
0E1E  =  b_settrk       equ     bios + 30
0E21  =  b_setsec       equ     bios + 33
0E24  =  b_setdma       equ     bios + 36
0E27  =  b_read         equ     bios + 39
0E2A  =  b_write        equ     bios + 42
0E2D  =  b_liststst     equ     bios + 45
0E30  =  b_sectrn       equ     bios + 48
```

```
;
; Relative Positionen der verschiedenen FCB-Daten innerhalb
; des FCB bzw. Directory-Eintrags.
;
0009  =  pos_ro        equ     9       ;'Read-Only'-Flag
                                       ;in Bit 7 an dieser
                                       ;Position

000C  =  pos_ex        equ     12      ;Extendnummer

000D  =  pos_s1        equ     13      ;reserviert

000E  =  pos_eg        equ     14      ;Extend-Gruppe

000E  =  pos_na        equ     14      ;Flag für 'File unver-
                                       ;ändert ?' (Bit 7)

000F  =  pos_rc        equ     15      ;Anzahl der Records im
                                       ;letzten Extend

0010  =  pos_dx        equ     16      ;Anfang der Blockta-
                                       ;belle

0020  =  pos_cr        equ     32      ;Aktuelle Recordnummer
                                       ;im Extend

0021  =  pos_r0        equ     33      ;RRN bei direktem Zu-
                                       ;griff

;
; Abstände zwischen den FCB-Parametern
;
0002  =  diff_ex_eg    equ     2       ;Abstand der ex-Posi-
                                       ;tion zur eg-Position

0011  =  diff_rc_cr    equ     17      ;Abstand der rc-Posi-
                                       ;tion zur cr-Position
```

```
;
; FBASE-Adresse
;
bdos:
;
;
; BDOS-Seriennummer:
;
; Diese 6 Bytes werden vom CCP, mit der im CCP gespeicherten
; Seriennummer verglichen.
; Näheres ist unter 'compare_serial' (01F5) im CCP be-
; schrieben.
;
serial_#:
;
0000  00        defs  6,0
;
;
; BDOS-Einsprung für alle Programme.
;
; In ?bdos ( = ?boot+0005h) steht ein Sprung an diese Stelle.
;
bdos_entry:
;
0006  C3 0011  jp    bdos1                ;Fehlertabelle über-
                                           ;springen
;
;
; Adressentabelle für die vier möglichen Fehlermeldungen.
;
; Diese Tabelle wird von der allgemeinen Fehlerroutine
; 'error_entry' (034A) benutzt.
;
err_loc_table:
;
0009  0099      defw  ?err_bad_sector    ;Bad Sector

000B  00A5      defw  ?err_select        ;Select

000D  00AB      defw  ?err_disk_ro       ;R/O

000F  00B1      defw  ?err_file_ro       ;File R/O
```

```
;
; Startroutine für alle BDOS-Funktionen
;
; I: C  - Funktionsnummer im Bereich von 0 bis no_of_funcs-1
;    DE - Funktionsparameter
;         (= ASCII-Zeichen für zeichenbezogene Funktionen)
;         (= FCB-Adresse für filebezogene Funktionen)
;
bdos1:
;
0011  EB        EX    DE,HL              ;Parameter nach HL
0012  22 0343   LD    (§entry_word),HL   ;Parameter zwischen-
                                         ;speichern
0015  EB        EX    DE,HL              ;Vertauschung wieder
                                         ;zurücknehmen

0016  7B        LD    A,E                ;LSB des Parameters
0017  32 0DD6   LD    (§entry_byte),A    ;getrennt speichern

001A  21 0000   LD    HL,0               ;HL = 0
001D  22 0345   LD    (§exit_value),HL   ;Rückgabeparameter
                                         ;auf 0 setzen

0020  39        ADD   HL,SP              ;HL = 0 + SP
                                         ;   - Stackadresse
0021  22 030F   LD    (§user_stack),HL   ;Stackadresse zwi-
                                         ;schenspeichern

0024  31 0341   LD    SP,stack           ;Stackpointer auf
                                         ;BDOS-Stack zeigen
                                         ;lassen

0027  AF        XOR   A                  ;A = 0
0028  32 0DE0   LD    (§fcb_disk_#),A    ;Aktuelle Disk als
                                         ;Disk für Fileopera-
                                         ;tionen annehmen
                                         ;(siehe 'auto_select'
                                         ;und '?bdos_exit')
002B  32 0DDE   LD    (§file_func?),A    ;Erst einmal annehmen,
                                         ;daß es sich bei der
                                         ;gewünschten Funktion
                                         ;nicht um eine File-
                                         ;funktion handelt.
                                         ;(siehe auto_select)

002E  21 0D74   LD    HL,?bdos_exit      ;Rückkehradresse für
                                         ;alle BDOS-Funktionen
0031  E5        PUSH  HL                 ;auf den Stack legen
```

```
0032  79        LD    A,C               ;A - Funktionsnummer
0033  FE 29     CP    no_of_funcs       ;Ist die Funktionsnum-
                                        ;mer größer oder
                                        ;gleich der Anzahl der
                                        ;Funktionen ?
0035  D0        RET   NC                ;J: Die Funktionsnum-
                                        ;mer ist zu groß,
                                        ;keine Funktion aus-
                                        ;führen

0036  4B        LD    C,E               ;N: C - LSB der Ein-
                                        ;gangsparameters
                                        ;(für zeichenbezogene
                                        ;Funktionen)

0037  21 0047   LD    HL,bdos_loc_table ;HL -> Adressentabelle
003A  5F        LD    E,A               ;DE - Funktionsnummer
003B  16 00     LD    D,0
003D  19        ADD   HL,DE             ;Funktionsnummer zwei-
003E  19        ADD   HL,DE             ;mal aufaddieren
                                        ;HL -> Adresse der ge-
                                        ;wünschten Funktion
003F  5E        LD    E,(HL)            ;Funktionsadresse nach
0040  23        INC   HL                ;DE laden
0041  56        LD    D,(HL)

0042  2A 0343   LD    HL,(§entry_word)  ;HL - Eingangspara-
                                        ;meter
0045  EB        EX    DE,HL             ;DE und HL vertauschen
                                        ;DE = Eingangspara-
                                        ;meter
                                        ;HL = Adresse der ge-
                                        ;wünschten Funktion
0046  E9        JP    (HL)              ;Funktion ausführen
```

```
;
; Adresstabelle aller BDOS-Funktionen in der Reihenfolge
; ihrer Funktionsnummern
;
; Die mit 'b_' beginnenden Adress-Label weisen auf die BIOS-
; Sprungleiste.  '?' am Labelanfang kennzeichnet ein BDOS-
; Label.
;
bdos_loc_table:
;
                                                    ;Funktionsnummer:
0047  0E03     defw  b_wboot                        ;  0
0049  02C8     defw  ?console_input                 ;  1
004B  0190     defw  ?console_output                ;  2
004D  02CE     defw  ?reader_input                  ;  3
004F  0E12     defw  b_punch                        ;  4
0051  0E0F     defw  b_list                         ;  5
0053  02D4     defw  ?direct_io                     ;  6
0055  02ED     defw  ?get_io_byte                   ;  7
0057  02F3     defw  ?set_io_byte                   ;  8
0059  02F8     defw  ?print_string                  ;  9
005B  01E1     defw  ?read_console_buffer           ; 10
005D  02FE     defw  ?get_console_status            ; 11
005F  0C7E     defw  ?return_version_#              ; 12
0061  0C83     defw  ?reset_disk_system             ; 13
0063  0C45     defw  ?select_disk                   ; 14
0065  0C9C     defw  ?open_file                     ; 15
0067  0CA5     defw  ?close_file                    ; 16
0069  0CAB     defw  ?search_first                  ; 17
006B  0CC8     defw  ?search_next                   ; 18
006D  0CD7     defw  ?delete_file                   ; 19
006F  0CE0     defw  ?read_sequ                     ; 20
0071  0CE6     defw  ?write_sequ                    ; 21
0073  0CEC     defw  ?make_file                     ; 22
0075  0CF5     defw  ?rename_file                   ; 23
0077  0CFE     defw  ?return_login_vec              ; 24
0079  0D04     defw  ?return_disk                   ; 25
007B  0D0A     defw  ?set_dma                       ; 26
007D  0D11     defw  ?get_allocation_addr           ; 27
007F  052C     defw  ?write_protect_disk            ; 28
0081  0D17     defw  ?get_ro_vec                    ; 29
0083  0D1D     defw  ?set_file_attr                 ; 30
0085  0D26     defw  ?get_parameter_addr            ; 31
0087  0D2D     defw  ?set_get_user_code             ; 32
0089  0D41     defw  ?read_random                   ; 33
008B  0D47     defw  ?write_random                  ; 34
008D  0D4D     defw  ?compute_file_size             ; 35
008F  0C0E     defw  ?set_random_record             ; 36
0091  0D53     defw  ?reset_disk                    ; 37
0093  0304     defw  ?dummy                         ; 38
0095  0304     defw  ?dummy                         ; 39
0097  0D9B     defw  ?write_zero_random             ; 40
```

```
;
; BDOS Fehlerroutinen:
;
;
; Fehlerart:  Schreib/Lesefehler
;
; Ursache:    Die BIOS-Funktionen zum Lesen bzw. Schreiben
;             eines  Records haben einen Fehler gemeldet.
;
?err_bad_sector:
;
0099  21 00CA  LD    HL,txt_bad_sector ;HL -> Fehlertext
009C  CD 00E5  CALL  show_error        ;Fehlermeldung aus-
                                       ;geben und auf ein
                                       ;Zeichen von der
                                       ;Console (Tastendruck)
                                       ;warten
009F  FE 03    CP    ctrlc             ;CTRL-C gedrückt ?
00A1  CA 0000  JP    Z,?boot           ;J: Funktion abbrechen
                                       ;und Warmstart aus-
                                       ;führen
00A4  C9       RET                     ;N: Funktion wieder-
                                       ;holen
;
;
; Fehlerart:  Fehlerhafte Diskanwahl
;
; Ursache:    Die BIOS-Funktion b_seldsk hat als DPH-Adresse
;             den Wert 0000H zurückgegeben.
;
?err_select:
;
00A5  21 00D5  LD    HL,txt_select     ;HL -> Fehlertext
00A8  C3 00B4  JP    err_boot          ;Fehlermeldung aus-
                                       ;geben und Warmstart
                                       ;ausführen
;
;
; Fehlerart:  Disk schreibgeschützt
;
; Ursache:    Das entsprechende Bit im R/O-Vektor ist gesetzt
;             (Über STAT oder wegen einer Prüfsummen-
;              Differenz)
;
?err_disk_ro:
;
00AB  21 00E1  LD    HL,txt_disk_ro    ;HL -> Fehlertext
00AE  C3 00B4  JP    err_boot          ;Fehlermeldung aus-
                                       ;geben und Warmstart
                                       ;ausführen
```

```
;
; Fehlerart:  File schreibgeschützt
;
; Ursache:    Das 'Read-Only'-Bit im Directory-Eintrag ist
;             gesetzt.
;
?err_file_ro:
;
00B1  21 00DC  LD    HL,txt_file_ro    ;HL -> Fehlertext
;
;
; UPRO für Fehlerroutinen.
; Fehlertext anzeigen und Warmstart ausführen.
;
; I: HL -> Fehlertext
;
err_boot:
;
00B4  CD 00E5  CALL  show_error        ;Fehlermeldung aus-
                                       ;geben und auf ein
                                       ;Zeichen von der
                                       ;Console (Tastendruck)
                                       ;warten
00B7  C3 0000  JP    ?boot             ;Warmstart ausführen
;
;
; Texte für die verschiedenen Fehlermeldungen.
;
; Dieser Text wird bei jeder Fehlermeldung ausgegeben:
;
error_text:
;
00BA           defm  'Bdos Err On '
;
;
; Hier wird die Bezeichnung der aktuellen Disk ein-
; getragen ('A' - 'P')
;
$disk_name:
;
00C6           defm  ' : $'
```

```
;
; Fehlertexte für die verschiedenen BDOS-Fehler
;
txt_bad_sector:
;
00CA            defm  'Bad Sector$'
;
txt_select:
;
00D5            defm  'Select$'
;
txt_file_ro:
;
00DC            defm  'File '             ;Nach 'File ' steht
                                          ;kein '$', also wird
                                          ;das folgende 'R/O'
                                          ;noch mit ausgegeben !
;
txt_disk_ro:
;
00E1            defm  'R/O$'
;
;
; UPRO für Fehlerroutinen.
; Komplette Fehlermeldung ausgeben und auf ein Zeichen von
; der Console (Tastendruck) warten
;
; I: HL -> Fehlermeldung
;
show_error:
;
00E5  E5        PUSH  HL                  ;Textzeiger retten

00E6  CD 01C9   CALL  print_cr_lf         ;Neue Zeile beginnen

00E9  3A 0342   LD    A,(§bdos_disk)      ;Disknummer holen
00EC  C6 41     ADD   A,'A'               ;+'A' ergibt die Disk-
                                          ;bezeichnung
00EE  32 00C6   LD    (§disk_name),A      ;Diskbezeichnung in
                                          ;die Fehlermeldung
                                          ;einsetzen

00F1  01 00BA   LD    BC,error_text       ;BC -> Fehlermeldung
00F4  CD 01D3   CALL  print_text          ;Ersten Teil der Feh-
                                          ;lermeldung ausgeben

00F7  C1        POP   BC                  ;Textzeiger vom Stack
                                          ;holen
00F8  CD 01D3   CALL  print_text          ;und Fehlertext aus-
                                          ;geben
```

```
;
; Zeichen von der Console holen
;
; I: -
;
; O: A - Zeichen
;
get_char:
;
00FB  21 030E  LD    HL,$buffered_char ;HL -> Zeichen von
                                       ;letzter Statusabfrage
00FE  7E       LD    A,(HL)           ;A - Zeichen
00FF  36 00    LD    (HL),0           ;Zeichen löschen
0101  B7       OR    A                ;War noch ein Zeichen
                                       ;zwischengepuffert ?
0102  C0       RET   NZ               ;J: Dieses Zeichen zu-
                                       ;rückgeben
0103  C3 0E09  JP    b_conin          ;N: Neues Zeichen von
                                       ;der Console holen
;
;
; Zeichen von der Console holen und ausgeben
;
; I: -
;
; O: A - Zeichen
;
input_char:
;
0106  CD 00FB  CALL  get_char         ;Zeichen holen
0109  CD 0114  CALL  check_char       ;Ist es ein darstell-
                                       ;bares Zeichen ?
010C  D8       RET   C                ;N: Zeichen nicht
                                       ;ausgeben
010D  F5       PUSH  AF               ;J: Zeichen retten
010E  4F       LD    C,A              ;C = Zeichen
010F  CD 0190  CALL  ?console_output  ;Zeichen ausgeben
0112  F1       POP   AF               ;Zeichen zurück
0113  C9       RET                    ;Fertig
```

```
;
; Zeichen überprüfen
;
; I: A = ASCII-Zeichen
;
; O: A unverändert
;    Z = 1, falls A = 'cr', 'lf', 'tab' oder 'back'
;    C = 1, falls A der ASCII-Wert ein anderen, nicht dar-
;           stellbaren Zeichens ist
;    Z = 0, C = 0 sonst
;
check_char:
;
0114  FE 0D     CP     cr                ;A = cr ?
0116  C8        RET    Z                 ;J: ok
0117  FE 0A     CP     lf                ;A = lf ?
0119  C8        RET    Z                 ;J: ok
011A  FE 09     CP     tab               ;A = tab ?
011C  C8        RET    Z                 ;J: ok
011D  FE 08     CP     back              ;A = back ?
011F  C8        RET    Z                 ;J: ok
0120  FE 20     CP     space             ;A < 'space' ?
0122  C9        RET                      ;Zurück, mit ent-
                                         ;sprechend gesetzten
                                         ;Flags
;
;
; Console auf anstehendes Zeichen überprüfen und Zeichen
; holen.
; Falls das Zeichen CTRL-S (xoff) war, wird auf ein zweites
; Zeichen gewartet.
;
; Diese Routine ist die 'Get Console Status'-Funktion des BDOS
;
char_xoff?:
;
0123  3A 030E   LD     A,(&buffered_char);Ist noch ein zwi-
0126  B7        OR     A                 ;schengepuffertes
                                         ;Zeichen vorhanden ?
0127  C2 0145   JP     NZ,true_status    ;J: Status 'true'
                                         ;melden

012A  CD 0E06   CALL   b_const           ;N: Consolenstatus
                                         ;vom BIOS holen
012D  E6 01     AND    01H               ;Steht ein Zeichen an?
012F  C8        RET    Z                 ;N: Status 'false'
                                         ;(Null) melden
```

```
0130  CD 0E09  CALL  b_conin           ;J: Zeichen abholen
0133  FE 13    CP    xoff              ;Ist es CTRL-S ?
0135  C2 0142  JP    NZ,save_status    ;N: Zeichen zwischen-
                                       ;puffern und Status
                                       ;'true' melden

0138  CD 0E09  CALL  b_conin           ;J: Auf zweites Zei-
                                       ;chen warten. Die
                                       ;Verarbeitung ist so
                                       ;lange gestoppt !
013B  FE 03    CP    ctrlc             ;Ist das nächste Zei-
                                       ;chen CTRL-C ?
013D  CA 0000  JP    Z,?boot           ;J: Warmstart aus-
                                       ;führen

0140  AF       XOR   A                 ;N: Status 'false'
0141  C9       RET                     ;zurückmelden
;
;
; Von der Console geholtes Zeichen zwischenpuffern und
; Status 'true' zurückmelden
;
save_status:
;
                                       ;Zeichen abspeichern
0142  32 030E  LD    (§buffered_char),A
;
;
; Status 'true' zurückmelden
;
true_status:
;
0145  3E 01    LD    A,1               ;A = 'true' (<> 0)
0147  C9       RET
;
;
; ASCII-Zeichen ausgeben und Zeichenzähler aktualisieren
;
; I: C = ASCII-Zeichen
;
; O: C unverändert
;
print_char:
;
0148  3A 030A  LD    A,(§dont_print?)  ;Soll das Zeichen aus-
0149  B7       OR    A                 ;gegeben werden ?
                                       ;(siehe CTRL-H und re-
                                       ;type_line)
014C  C2 0162  JP    NZ,adjust_column  ;N: Nur den Spalten-
                                       ;zähler (§column) an-
                                       ;passen
```

```
014F  C5        PUSH  BC                ;N: Zeichen retten
0150  CD 0123   CALL  char_xoff?        ;CTRL-S abprüfen und
                                        ;Verarbeitung evtl.
                                        ;anhalten
0153  C1        POP   BC                ;Zeichen zurück

0154  C5        PUSH  BC                ;Zeichen retten
0155  CD 0E0C   CALL  b_conout          ;Zeichen ausgeben
0158  C1        POP   BC                ;Zeichen zurück

0159  C5        PUSH  BC                ;Zeichen retten
015A  3A 030D   LD    A,(§list_flag)    ;§list_flag gesetzt ?
015D  B7        OR    A
015E  C4 0E0F   CALL  NZ,b_list         ;J: Zeichen auch über
                                        ;den List-Kanal aus-
                                        ;geben
0161  C1        POP   BC                ;Zeichen zurück
;
;
; Zähler für die Anzahl der in der aktuellen Zeile aus-
; gegebenen Zeichen (§column) anpassen.
;
adjust_column:
;
0162  79        LD    A,C               ;A - Zeichen
0163  21 030C   LD    HL,§column        ;HL -> Zähler
0166  FE 7F     CP    del               ;Ist das Zeichen
                                        ;= 'del' ?
0168  C8        RET   Z                 ;J: Fertig

0169  34        INC   (HL)              ;N: Zähler +1
016A  FE 20     CP    space             ;Ist das Zeichen
                                        ;darstellbar ?
                                        ;(Also in der ASCII-
                                        ;Tabelle >= space ?)
016C  D0        RET   NC                ;J: Fertig
                                        ;N: Das zuletzt aus-
                                        ;gegebene Zeichen war
                                        ;ein nicht darstell-
                                        ;bares Zeichen
016D  35        DEC   (HL)              ;Die vorgenommene Er-
                                        ;höhung des Zählers
                                        ;wieder zurücknehmen
016E  7E        LD    A,(HL)            ;A = Zähler
016F  B7        OR    A                 ;Ist der Zähler = 0 ?
0170  C8        RET   Z                 ;J: Fertig
```

```
0171  79         LD     A,C              ;N: A - Zeichen
0172  FE 08      CP     back             ;War das Zeichen back-
                                          ;space (Rückwärts-
                                          ;schritt ?)
0174  C2 0179    JP     NZ,not_back      ;N: Weitertesten

0177  35         DEC    (HL)             ;J: Es wurde ein Zei-
                                          ;chen gelöscht; Zäh-
                                          ;ler entsprechend an-
                                          ;passen
0178  C9         RET
;
;
not_back:
;
0179  FE 0A      CP     1f               ;Wurde eine neue Zeile
                                          ;begonnen ?
017B  C0         RET    NZ               ;N: Fertig

017C  36 00      LD     (HL),0           ;J: Zähler auf Null
017E  C9         RET                     ;setzen
;
;
; ASCII-Zeichen in darstellbarer Form ausgeben
;
; I: C - ASCII-Zeichen
;
; O: -
;
show_char:
;
017F  79         LD     A,C              ;A = Zeichen
0180  CD 0114    CALL   check_char       ;Ist es ein darstell-
                                          ;bares Zeichen ?
                                          ;J: Zeichen ausgeben
0183  D2 0190    JP     NC,?console_output

0186  F5         PUSH   AF               ;N: Zeichen retten
0187  0E 5E      LD     C,'^'            ;'^' (CTRL-Kennung)
0189  CD 0148    CALL   print_char       ;ausgeben
018C  F1         POP    AF               ;Zeichen zurück
018D  F6 40      OR     '@'              ;ASCII-Wert zwischen
                                          ;0 und 31 in ent-
                                          ;sprechendes ASCII-
                                          ;zeichen '@' bis '_'
                                          ;umwandeln
018F  4F         LD     C,A              ;C = Zeichen für die
                                          ;folgende Funktion
```

```
;
;
; **** BDOS-Funktion Nummer 2  'Console Output'
;
; I: C = Zeichen
;
; O: -
;
?console_output:
;
0190  79       LD    A,C               ;A = Zeichen
0191  FE 09    CP    tab               ;Zeichen = tab ?
0193  C2 0148  JP    NZ,print_char     ;N: Zeichen normal
                                        ;ausgeben
;
;
; Der Tab-Code wird vom BDOS in soviel Leerzeichen gewandelt,
; daß die nächste, durch 8 teilbare Spaltenposition erreicht
; wird.
;
show_tab:
;
0196  0E 20    LD    C,space           ;C = Leerzeichen
0198  CD 0148  CALL  print_char        ;Leerzeichen ausgeben
019B  3A 030C  LD    A,(§column)       ;A = Spaltenposition
019E  E6 07    AND   7                 ;modulo 8
                                        ;Ist die Spaltenposi-
                                        ;tion durch 8 teilbar?
01A0  C2 0196  JP    NZ,show_tab       ;N: Noch ein Leerzei-
                                        ;chen ausgeben
01A3  C9       RET                     ;J: Fertig
;
;
; UPRO für ?read_console_buffer
;
; Ein bereits ausgegebenes Zeichen, wird durch die Zeichen-
; sequenz 'backspace' - 'space' - 'backspace' wieder vom
; Bildschirm gelöscht.
;
print_backspace:
;
01A4  CD 01AC  CALL  print_back        ;backspace ausgeben
01A7  0E 20    LD    C,space           ;space
01A9  CD 0E0C  CALL  b_conout          ;ausgeben
;
;
; 'back'-Code ausgeben
;
print_back:
;
01AC  0E 08    LD    C,back            ;backspace
01AE  C3 0E0C  JP    b_conout          ; ausgeben
```

```
;
; UPRO für ?read_console_buffer
;
; '#' ausgeben, neue Zeile anfangen und die Schreibposition
; (Cursor) in die gleiche Spalte, wie beim Start von '?read_-
; console_buffer' bringen.
;
new_line:
;
01B1  0E 23     LD    C,'#'              ;'#'
01B3  CD 0148   CALL  print_char         ; ausgeben
01B6  CD 01C9   CALL  print_cr_lf        ;Neue Zeile anfangen
;
;
; Solange Leerzeichen ausgeben, bis die Anfangs-Spalten-
; position wieder erreicht ist.
;
new_line_loop:
;
01B9  3A 030C   LD    A,(§column)        ;A = aktuelle Spalten-
                                         ;position
01BC  21 030B   LD    HL,§start_pos      ;HL -> Anfangs-Spal-
                                         ;tenposition
01BF  BE        CP    (HL)               ;Position vom Anfang
                                         ;erreicht ?
01C0  D0        RET   NC                 ;J: Fertig
01C1  0E 20     LD    C,space            ;N: Leerzeichen
01C3  CD 0148   CALL  print_char         ; ausgeben
01C6  C3 01B9   JP    new_line_loop      ;Und Position neu
                                         ;testen
;
;
; Neue Ausgabezeile beginnen
;
print_cr_lf:
;
01C9  0E 0D     LD    C,cr               ;Carriage Return
01CB  CD 0148   CALL  print_char         ; ausgeben
01CE  0E 0A     LD    C,lf               ;Line Feed
01D0  C3 0148   JP    print_char         ; ausgeben
```

```
;
; Text ausgeben.
; Das Textende ist durch '$' gekennzeichnet.
;
; I: BC -> Text
;
; O: BC -> '$' (Textende)
;
print_text:
;
01D3  0A        LD    A,(BC)              ;Zeichen holen
01D4  FE 24     CP    '$'                 ;Ende ('$') erreicht ?
01D6  C8        RET   Z                   ;J: Fertig

01D7  03        INC   BC                  ;N: Zeiger +1
01D8  C5        PUSH  BC                  ;Zeiger retten
01D9  4F        LD    C,A                 ;C - Zeichen
01DA  CD 0190   CALL  ?console_output     ;Zeichen ausgeben
01DD  C1        POP   BC                  ;Zeiger zurück
01DE  C3 01D3   JP    print_text          ;Weiter, bis das Ende
                                          ;erreicht ist
;
;
; **** BDOS-Funktion Nummer 10  'Read Console Buffer'
;
; I: (§entry_word) -> Eingabebuffer
;
; O: Eingabebuffer, gefüllt mit den eingegebenen Zeichen
;
?read_console_buffer:
;
01E1  3A 030C   LD    A,(§column)         ;Aktuelle Spaltenposi-
                                          ;tion
01E4  32 0305   LD    (§start_pos),A      ;in §start_pos merken

01E7  2A 0343   LD    HL,(§entry_word)    ;HL -> Buffer
01EA  4E        LD    C,(HL)              ;C = Bufferlänge
01EB  23        INC   HL                  ;HL -> Buffer +1

01EC  E5        PUSH  HL                  ;Bufferadresse retten
01ED  06 00     LD    B,0                 ;B = Zähler der ein-
                                          ;gegebenen Zeichen
;
;
; Bufferzeiger und Bufferzähler retten
; Nächstes Zeichen von Console holen
;
read_next1:
;
01EF  C5        PUSH  BC
01F0  E5        PUSH  HL
```

```
;
; Nächstes Zeichen von Console holen
;
read_next2:
;
01F1   CD 00FB   CALL   get_char          ;Zeichen holen (ohne
                                          ;Ausgabe !)
01F4   E6 7F     AND    07FH              ;Bit 7 löschen

01F6   E1        POP    HL                ;Bufferzeiger zurück
01F7   C1        POP    BC                ;Bufferzähler zurück
01F8   FE 0D     CP     cr                ;Eingabezeile be-
                                          ;endet ?
01FA   CA 02C1   JP     Z,end_of_input    ;J: Eingabe abschlies-
                                          ;sen

01FD   FE 0A     CP     lf                ;Eingabezeile be-
                                          ;endet ?
01FF   CA 02C1   JP     Z,end_of_input    ;J: Eingabe abschlies-
                                          ;sen

0202   FE 08     CP     back              ;CTRL-H eingegeben ?
0204   C2 0216   JP     NZ,check_del      ;N: Sprung
;
;
; CTRL-H
; Zuletzt eingegebenes Zeichen wieder löschen
;
0207   78        LD     A,B               ;A = Anzahl der be-
                                          ;reits eingegebenen
                                          ;Zeichen
0208   B7        OR     A                 ;Schon Zeichen einge-
                                          ;geben ?
0209   CA 01EF   JP     Z,read_next1      ;N: CTRL-H hat am An-
                                          ;fang einer Zeile kei-
                                          ;ne Funktion

020C   05        DEC    B                 ;J: Zeichenanzahl um
                                          ;eins verringern
020D   3A 030C   LD     A,(§column)       ;Aktuelle Spaltenposi-
                                          ;tion
0210   32 030A   LD     (§dont_print?),A  ;für back_char? spei-
                                          ;chern und währende
                                          ;des retype_loop keine
                                          ;Zeichen ausgeben
0213   C3 0270   JP     retype_line       ;Neue Spaltenposition
                                          ;brechnen und ent-
                                          ;sprechend viele Zei-
                                          ;chen von der Console
                                          ;löschen
```

```
;
check_del:
;
0216  FE 7F     CP    del                ;DEL eingegeben ?
0218  C2 0226   JP    NZ,check_ctrle     ;N: Sprung
;
;
; DEL
; Zuletzt eingegebenes Zeichen wieder löschen und ausgeben
;
021B  78        LD    A,B                ;A - Anzahl der bisher
                                         ;eingegebenen Zeichen
021C  B7        OR    A                  ;Stehen schon Zeichen
                                         ;im Buffer ?
021D  CA 01EF   JP    Z,read_next1       ;N: Dann kann auch
                                         ;nichts gelöscht wer-
                                         ;den

0220  7E        LD    A,(HL)             ;J: Letztes Zeichen
                                         ;holen
0221  05        DEC   B                  ;Zeichenzähler -1
0222  2B        DEC   HL                 ;Bufferzeiger -1
0223  C3 02A9   JP    show_input         ;Zeichen ausgeben
;
;
check_ctrle:
;
0226  FE 05     CP    ctrle              ;CTRL-E eingegeben ?
0228  C2 0237   JP    NZ,check_ctrlp     ;N: Sprung
;
;
; CTRL-E
; Neue Ausgabezeile beginnen, Buffer unverändert lassen
;
022B  C5        PUSH  BC                 ;Zeichenzähler
022C  E5        PUSH  HL                 ;und Bufferzeiger
                                         ;retten
022D  CD 01C9   CALL  print_cr_lf        ;Neue Ausgabezeile
                                         ;beginnen
0230  AF        XOR   A                  ;A = 0
0231  32 030B   LD    (start_pos),A      ;Startposition auf
                                         ;Null setzen
0234  C3 01F1   JP    read_next2         ;Nächstes Zeichen ho-
                                         ;len.
```

```
;
check_ctrlp:
;
0237  FE 10     CP     ctrlp              ;CTRL-P eingegeben ?
0239  C2 0248   JP     NZ,check_ctrlx     ;N: Sprung
;
;
; CTRL-P
; Parallelausgabe der Consolenzeichen auf dem Drucker ein-
; bzw. ausschalten.
;
023C  E5        PUSH   HL                 ;Bufferzeiger retten
023D  21 030D   LD     HL,$print_flag     ;HL -> $print_flag
0240  3E 01     LD     A,1                ;A = 1
0242  96        SUB    (HL)               ;A = 0, falls $print_
                                          ;flag 1 war
                                          ;A = 1, falls $print_
                                          ;flag 0 war
0243  77        LD     (HL),A             ;Neues $print_flag
                                          ;setzen
0244  E1        POP    HL                 ;Bufferzeiger zurück
0245  C3 01EF   JP     read_next1         ;Nächstes Zeichen ho-
                                          ;len
;
;
check_ctrlx:
;
0248  FE 18     CP     ctrlx              ;CTRL-X eingegeben ?
024A  C2 025F   JP     NZ,check_ctrlu     ;N: Sprung
;
;
; CTRL-X
; Eingegebene Zeile löschen und Eingabe neu starten.
; Die bisher auf der Console angezeigten Zeichen
; werden ebenfalls gelöscht.
;
024D  E1        POP    HL                 ;Bufferzeiger vom
                                          ;Stack nehmen
```

```
;
ctrlx_loop:
;
024E  3A 030B  LD    A,(§start_pos)     ;A - Startposition
0251  21 030C  LD    HL,§column         ;HL -> aktuelle Spal-
                                        ;tenposition
0254  BE       CP    (HL)               ;Startposition er-
                                        ;reicht ?
                                        ;J: Eingabe neu be-
                                        ;ginnen
0255  D2 01E1  JP    NC,?read_console_buffer

0258  35       DEC   (HL)               ;N: Spaltenposition -1
0259  CD 01A4  CALL  print_backspace    ;Zeichen auf der Con-
                                        ;sole löschen
025C  C3 024E  JP    ctrlx_loop         ;Weiter, bis Startpo-
                                        ;sition erreicht ist
;
;
check_ctrlu:
;
025F  FE 15    CP    ctrlu              ;CTRL-U eingegeben ?
0261  C2 026B  JP    NZ,check_ctrlr     ;N: Sprung
;
;
; CTRL-U
; Eingegebene Zeile löschen und Eingabe neu starten.
; Die bisher auf der Console angezeigten Zeichen wer-
; den nicht gelöscht, sondern eine neue Zeile begon-
; nen.
;
0264  CD 01B1  CALL  new_line           ;Neue Zeile beginnen
0267  E1       POP   HL                 ;Bufferzeiger vom
                                        ;Stack nehmen und
                                        ;Eingabe neu starten
0268  C3 01E1  JP    ?read_console_buffer
;
;
check_ctrlr:
;
026B  FE 12    CP    ctrlr              ;CTRL-R eingegeben ?
026D  C2 02A6  JP    NZ,save_char       ;N: Sprung
```

```
;
; CTRL-R
; Ausgabezeile beenden und die bisher eingegebene Zeile
; neu anzeigen.
;
; UPRO für CTRL-H
; Neue Spaltenposition nach der Rücknahme des letzten
; Zeichens berechnen. In diesem Fall, werden die Zeichen
; beim 'retype_loop' nicht angezeigt, sondern nur '$column'
; neu berechnet
;
retype_line:
;
0270  C5        PUSH  BC                ;Zeichenzähler retten
0271  CD 01B1   CALL  new_line          ;Neue Ausgabezeile be-
                                        ;ginnen
0274  C1        POP   BC                ;Zeichenzähler zurück
0275  E1        POP   HL                ;Bufferzeiger zurück

0276  E5        PUSH  HL                ;Bufferzeiger retten
0277  C5        PUSH  BC                ;Zeichenzähler retten
;
;
; Bisher eingegebene Zeile neu anzeigen
;
retype_loop:
;
0278  78        LD    A,B               ;A = Zähler
0279  B7        OR    A                 ;Noch Zeichen auszu-
                                        ;geben ?
027A  CA 028A    JP    Z,delete_char     ;N: weiter bei delete_
                                        ;char

027D  23        INC   HL                ;Zeiger +1
027E  4E        LD    C,(HL)            ;Zeichen holen
027F  05        DEC   B                 ;Zähler -1
0280  C5        PUSH  BC                ;Zähler retten
0281  E5        PUSH  HL                ;Zeiger retten
0282  CD 017F   CALL  show_char         ;Zeichen ausgeben
0285  E1        POP   HL                ;Zeiger zurück
0286  C1        POP   BC                ;Zähler zurück
0287  C3 0278   JP    retype_loop       ;Nächstes Zeichen aus-
                                        ;geben
;
;
delete_char:
;
028A  E5        PUSH  HL                ;Bufferzeiger retten
028B  3A 030A    LD    A,($dont_print?)  ;Kam der Aufruf von
028E  B7        OR    A                 ;CTRL-H ?
028F  CA 01F1    JP    Z,read_next2      ;N: Fertig, Nächstes
                                        ;Zeichen holen
```

```
                                                    ;J: Alle Zeichen bis
                                                    ;zur neuen Spalten-
                                                    ;position zurücknehmen
                                                    ;A = Alte Spaltenpo-
                                                    ;sition
0292  21 030C  LD    HL,§column                    ;HL -> Neue Spalten-
                                                    ;position
0295  96       SUB   (HL)                          ;A = Anzahl der noch
                                                    ;zu löschenden Zeichen
0296  32 030A  LD    (§back_count),A               ;Anzahl speichern
;
;
; Zeichen zwischen der alten und neuen Spaltenposition
; löschen
;
back_tab:
;
0299  CD 01A4  CALL  print_backspace               ;Zeichen löschen
029C  21 030A  LD    HL,§back_count                ;HL -> Zähler
029F  35       DEC   (HL)                          ;Zähler -1, Noch Zei-
                                                    ;chen zu löschen ?
02A0  C2 0299  JP    NZ,back_tab                   ;J: Zeichen löschen
02A3  C3 01F1  JP    read_next2                    ;N: Fertig, Nächstes
                                                    ;Zeichen holen
;
;
; Das eingegebenes Zeichen war kein Steuercode;
; Zeichen in den Buffer übernehmen und anzeigen.
;
save_char:
;
02A6  23       INC   HL                            ;Bufferzeiger +1
02A7  77       LD    (HL),A                        ;Zeichen in den Buf-
                                                    ;fer übernehmen
02A8  04       INC   B                             ;Zeichenzähler +1
;
;
; Zeichen ausgeben
;
show_input:
;
02A9  C5       PUSH  BC                            ;Zeichenzähler retten
02AA  E5       PUSH  HL                            ;Bufferzeiger retten
02AB  4F       LD    C,A                           ;C = Zeichen
02AC  CD 017F  CALL  show_char                     ;Zeichen ausgeben
02AF  E1       POP   HL                            ;Bufferzeiger zurück
02B0  C1       POP   BC                            ;Zeichenzähler zurück
```

```
02B1  7E       LD    A,(HL)          ;Zeichen aus dem Buf-
                                     ;fer holen
02B2  FE 03    CP    ctrlc           ;War es CTRL-C ?
02B4  78       LD    A,B             ;A = Zeichenzähler
02B5  C2 02BD  JP    NZ,not_ctrlc    ;N: Weiter

02B8  FE 01    CP    1               ;J: War es das erste
                                     ;Zeichen ?
02BA  CA 0000  JP    Z,?boot         ;J: Warmstart
;
;
not_ctrlc:
;
02BD  B9       CP    C               ;N: Maximale Zeichen-
                                     ;zahl erreicht ?
02BE  DA 01EF  JP    C,read_next1    ;N: Nächstes Zeichen
                                     ;holen
                                     ;J: Eingabe beenden
;
; Ende der Zeileneingabe
;
end_of_input:
;
02C1  E1       POP   HL              ;Bufferanfangszeiger
                                     ;zurück
02C2  70       LD    (HL),B          ;Anzahl der eingegebe-
                                     ;nen Zeichen abspei-
                                     ;chern
02C3  0E 0D    LD    C,cr            ;C = CR-Code
02C5  C3 0148  JP    print_char      ;Ausgabezeile ab-
                                     ;schliessen
;
;
; **** BDOS-Funktion Nummer 1  'Console Input'
;
?console_input:
;
02C8  CD 0106  CALL  input_char      ;Zeichen von der Con-
                                     ;sole holen
02CB  C3 0301  JP    return_byte     ;und zurückgeben
;
;
; **** BDOS-Funktion Nummer 3  'Reader Input'
;
?reader_input:
;
02CE  CD 0E15  CALL  b_reader        ;BIOS-Aufruf 'Reader
                                     ;Input'
02D1  C3 0301  JP    return_byte     ;Zeichen zurückgeben
```

```
;
; **** BDOS-Funktion Nummer 6  'Direct Console I/O'
;
?direct_io:
;
02D4  79       LD    A,C              ;A = Eingangspara-
                                      ;meter
02D5  3C       INC   A                ;Parameter = 0FFH ?
                                      ;(Console Input ?)
02D6  CA 02E0  JP    Z,direct_conin   ;J: Console Input

02D9  3C       INC   A                ;Parameter = 0FEH ?
                                      ;(Console Status ?)
02DA  CA 0E06  JP    Z,b_const        ;J: BIOS-Aufruf 'Con-
                                      ;sole Status'

02DD  C3 0E0C  JP    b_conout         ;N: Parameter ist Zei-
                                      ;chen; Zeichen ausge-
                                      ;ben
;
;
; UPRO für ?direct_io
; Console Eingabe
;
direct_conin:
;
02E0  CD 0E06  CALL  b_const          ;Steht ein Zeichen von
02E3  B7       OR    A                ;der Console an ?
02E4  CA 0D91  JP    Z,ret_to_user    ;N: Fertig

02E7  CD 0E09  CALL  b_conin          ;J: Zeichen holen
02EA  C3 0301  JP    return_byte      ;und zurückgeben
;
;
; **** BDOS-Funktion Nummer 7  'Get I/O Byte'
;
?get_io_byte:
;
02ED  3A 0003  LD    A,(§io_byte)     ;I/O-Byte holen
02F0  C3 0301  JP    return_byte      ;und zurückgeben
;
;
; **** BDOS-Funktion Nummer 8  'Set I/O Byte'
;
?set_io_byte:
;
02F3  21 0003  LD    HL,§io_byte      ;HL -> I/O-Byte
02F6  71       LD    (HL),C           ;Neues I/O-Byte ein-
                                      ;setzen
02F7  C9       RET
```

```
;
; **** BDOS-Funktion Nummer 9  'Print String'
;
?print_string:
;
02F8  EB        EX    DE,HL              ;HL - Textzeiger
02F9  4D        LD    C,L
02FA  44        LD    B,H                ;BC - Textzeiger
02FB  C3 01D3   JP    print_text         ;Text ausgeben
;
;
; **** BDOS-Funktion Nummer 11  'Get Console Status'
;
?get_console_status:
;
02FE  CD 0123   CALL  char_xoff?         ;Consolenstatus holen
                                         ;und zurückgeben:
;
;
; Byte-Wert an das aufrufende Programm zurückgeben
;
return_byte:
;
0301  32 0345   LD    (§exit_value),A    ;Wert abspeichern
;
;
; **** BDOS-Funktionen Nummer 38 und 39
; Keine Funktion
;
?dummy:
;
0304  C9        RET                      ;Rücksprung nach
                                         ;'?bdos_exit'
;
;
; UPRO für Dateibezogene BDOS-Funktionen
; Fehler bei Diskettenein-/ausgabe melden
;
read_write_error:
;
0305  3E 01     LD    A,1                ;Fehlercode 1
0307  C3 0301   JP    return_byte        ;zurückgeben
```

```
;
; Parameterbereich für zeichenbezogene Funktionen
;
;
; Flag, ob während 'retype_line' Zeichen ausgegeben werden
; sollen
; =  00H, Zeichen ausgeben
;         (CTRL-R Funktion)
; <> 00H, Nur den Spaltenzähler (§column) berechnen
;         (CTRL-H Funktion)
;
                §dont_print:            ;Gleiche Adresse wie
                                        ;§back_count:
;
;
; Zähler für die CTRL-H Funktion
;
030A  00        §back_count:  defb    0
;
;
; Spaltenposition beim Beginn von ?read_console_buffer
;
030B  00        §start_pos:       defb    0
;
;
; Aktuelle Spaltenposition
;
030C  00        §column:          defb    0
;
;
; Flag für 'print_char'
; = 00H, falls das Zeichen nur auf der Console ausgegeben
;        werden soll
; = 01H, falls das Zeichen auch über den List-Kanal aus-
;        gegeben werden soll
;
; Die Umschaltung dieses Flags, geschieht über CTRL-P bei
; der 'read_console_buffer'-Funktion
;
030D  00        §print_flag:      defb    0
;
;
; 1 Zeichen Zwischenpuffer für Console Status/Input
;
030E  00        §buffered_char:   defb    0
;
;
; Zwischenspeicher für den Stackpointer des aufrufenden
; Programms. Das BDOS benutzt eine eigenen Stackbereich.
;
030F  0000      §user_stack:      defw    0
```

```
;
; BDOS-Interner Stackbereich
;
0311  00                        defs    48,0
;
stack:
;
;
; Erster Parameterbereich für dateibezogene BDOS-Funktionen
;
;
; Nummer des aktuellen Userbereichs
;
0341  00        §user_#:        defb    0
;
;
; Aktuelle Disknummer
;
0342  00        §bdos_disk:     defb    0
;
;
; Zwischenspeicher für 16-Bit Eingabewert beim BDOS-Aufruf
; (meist steht hier die Adresse eines FCB)
;
0343  0000      §entry_word:    defw    0
;
; Zwischenspeicher für 8/16-Bit Rückgabeparameter vom
; BDOS-Aufruf
;
0345  0000      §exit_value:    defw    0
;
;
; Select Error
;
select_error:
;
0347  21 000B   LD    HL,err_loc_table+2;HL -> Fehlertabelle
;
;
; Allgemeiner Einsprung für Fehlermeldungen
;
; I: HL -> Adresse der gewünschten Fehlerroutine
;
error_entry:
;
034A  5E        LD    E,(HL)            ;E = LSB der Adresse
034B  23        INC   HL
034C  56        LD    D,(HL)            ;D = MSB der Adresse
034D  EB        EX    DE,HL             ;HL = Adresse der Feh-
                                        ;lerroutine
034E  E9        JP    (HL)              ;Fehlerroutine aus-
                                        ;führen
```

```
;
; Allgemeine Kopierroutine
; C Bytes von (DE) nach (HL) kopieren
;
; I: C  =  Anzahl der zu kopierenden Bytes
;    DE -> Quellbereich
;    HL -> Zielbereich
;
move_de_hl:
;
034F  0C        INC   C                 ;Zähler +1, da der
                                        ;Zähler zuerst ge-
                                        ;testet wird
;
move_loop:
;
0350  0D        DEC   C                 ;Zähler -1
                                        ;Zähler = 0 ?
0351  C8        RET   Z                 ;J: Fertig

0352  1A        LD    A,(DE)            ;N: Byte holen
0353  77        LD    (HL),A            ;Byte speichern
0354  13        INC   DE                ;Quellzeiger +1
0355  23        INC   HL                ;Zielzeiger +1
0356  C3 0350    JP    move_loop         ;Nächstes Byte
;
;
; UPRO für ?select_disk
; Disk anwählen und Parameter übernehmen
;
; I: §bdos_disk = Nummer der anzuwählenden Disk
;    E,0        = 1, falls die Disk seit dem letzten
;                 Warmstart schonmal angewählt wurde.
;
; O: Z-Flag gesetzt, falls der BIOS-Aufruf fehlerhaft war
;
physical_select:
;
0359  3A 0342   LD    A,(§bdos_disk)    ;A = Gewünschte Disk-
                                        ;nummer
035C  4F        LD    C,A               ;C = Disknummer
035D  CD 0E1B   CALL  b_seldsk          ;BIOS-Aufruf

0360  7C        LD    A,H               ;HL = 0 ?
0361  B5        OR    L
0362  C8        RET   Z                 ;J: Disk kann nicht
                                        ;selektiert werden
```

```
                                                    ;N: HL -> DPH
0363  5E        LD    E,(HL)                      ;XLTAB-Adresse holen
0364  23        INC   HL
0365  56        LD    D,(HL)
0366  23        INC   HL
0367  22 0DB3   LD    (?§last_entry),HL           ;Adresse von DPH+2
                                                    ;merken
036A  23        INC   HL
036B  23        INC   HL
036C  22 0DB5   LD    (?§track),HL                ;Adresse von DPH+4
                                                    ;merken
036F  23        INC   HL
0370  23        INC   HL
0371  22 0DB7   LD    (?§trkrecord),HL            ;Adresse von DPH+6
                                                    ;merken
0374  23        INC   HL
0375  23        INC   HL
0376  EB        EX    DE,HL                       ;HL - XLTAB-Adresse
0377  22 0DD0   LD    (?§xltab),HL                ;XLTAB-Adresse
                                                    ;speichern

037A  21 0DB9   LD    HL,?§dirbuf                 ;Nach ?§dirbuf werden
037D  0E 08     LD    C,8                         ;die nächsten 8 Bytes
037F  CD 034F   CALL  move_de_hl                  ;des DPH kopiert

0382  2A 0DBB   LD    HL,(?§dpb)                  ;HL = DPB-Adresse
0385  EB        EX    DE,HL                       ;DE -> DPB
0386  21 0DC1   LD    HL,§dpb                     ;Nach §dpb werden die
0389  0E 0F     LD    C,15                        ;15 Bytes des DPB
038B  CD 034F   CALL  move_de_hl                  ;kopiert

038E  2A 0DC6   LD    HL,(§dsm)                   ;HL = Anzahl der
                                                    ;Blöcke der selek-
                                                    ;tierten Disk
0391  7C        LD    A,H                         ;A = MSB der Block-
                                                    ;anzahl
0392  21 0DDD   LD    HL,§large_dsm?              ;HL -> Flag, ob die
                                                    ;Blocknummern als 8-
                                                    ;oder 16-Bit Werte
                                                    ;verwaltet werden
0395  36 FF     LD    (HL),0FFH                   ;Ersteinmal 8-Bit
                                                    ;Blocknummern annehmen
0397  B7        OR    A                           ;Ist das MSB der
                                                    ;Blockanzahl = 0 ?
0398  CA 039D   JP    Z,select_new_ok             ;J: Die Annahme war
                                                    ;richtig
039B  36 00     LD    (HL),0                      ;N: Die Blocknummern
                                                    ;sind 16-Bit Werte
```

```
;
select_new_ok:
;
039D  3E FF     LD     A,0FFH              ;A ungleich Null
039F  B7        OR     A                   ;Z-Flag löschen
03A0  C9        RET                        ;Fertig
;
;
; UPRO für logical_select und search_first
; Spurnummer der angewählten Disk und die dazugehörigen
; DPH-Parameter auf Null setzen.
;
home_disk:
;
03A1  CD 0E18   CALL   b_home              ;BIOS-Aufruf Track 0
                                           ;anwählen
03A4  AF        XOR    A                   ;A = 0
03A5  2A 0DB5   LD     HL,(?§track)        ;HL -> Spurnummer
03A8  77        LD     (HL),A              ;Spurnummer auf
03A9  23        INC    HL                  ; Null
03AA  77        LD     (HL),A              ; setzen
                                           ;HL -> Recordnummer
                                           ;der Spur
03AB  2A 0DB7   LD     HL,(?§trkrecord)
03AE  77        LD     (HL),A              ;Spurrecord auf Null
03AF  23        INC    HL                  ; setzen
03B0  77        LD     (HL),A
03B1  C9        RET
;
;
; Record lesen und Fehlerbehandlung durchführen
;
read_record:
;
03B2  CD 0E27   CALL   b_read              ;BIOS-Aufruf
03B5  C3 03BB   JP     test_rw_status      ;Fehler ?
;
;
; Record schreiben und Fehlerbehandlung durchführen
;
write_record:
;
03B8  CD 0E2A   CALL   b_write             ;BIOS-Aufruf
;
test_rw_status:
;
03BB  B7        OR     A                   ;Fehler beim BIOS-
                                           ;Aufruf ?
03BC  C8        RET    Z                   ;N: Fertig
```

```
;
; Fehler: 'Bad Sector' melden
;
bad_sector_error:
;
03BD  21 0009  LD    HL,err_loc_table  ;J: HL -> Fehler-
                                       ;adresse 'BAD SECTOR'
03C0  C3 034A  JP    error_entry       ;Fehlerroutine aus-
                                       ;führen
;
;
; Spur und Sektor im Directorybereich der Diskette anwählen
;
; I: (§dir_entry_#) - Nummer des gewünschten Directory-
;                     Eintrags
;
set_dir_trk_sec:
;
03C3  2A 0DEA  LD    HL,(§dir_entry_#) ;HL - Nummer des Ein-
                                       ;trags
03C6  0E 02    LD    C,002H            ;C - 2
03C8  CD 04EA  CALL  shift_right_hl    ;HL - HL/4 - Nummer
                                       ;des gewünschten Di-
                                       ;rectory-Records
03CB  22 0DE5  LD    (§abs_rec_#),HL   ;Absolute Recordnummer
                                       ;setzen
03CE  22 0DEC  LD    (§dir_record_#),HL;Direcory-Recordnummer
                                       ;speichern
;
;
; Spur und Sektor im Datenbereich der Diskette anwählen
;
; I: (§abs_rec_#) = Nummer des gewünschten Records
;
set_dsk_trk_sec:
;
03D1  21 0DE5  LD    HL,§abs_rec_#     ;HL -> Recordnummer
03D4  4E       LD    C,(HL)
03D5  23       INC   HL
03D6  46       LD    B,(HL)            ;BC = Recordnummer

03D7  2A 0DB7  LD    HL,(?§trkrecord)
03DA  5E       LD    E,(HL)
03DB  23       INC   HL
03DC  56       LD    D,(HL)            ;DE = Nummer des ers-
                                       ;ten Records der aktu-
                                       ;ellen Spur
```

```
03DD  2A 0DB5  LD    HL,(?§track)
03E0  7E       LD    A,(HL)
03E1  23       INC   HL
03E2  66       LD    H,(HL)
03E3  6F       LD    L,A                ;HL = Aktuelle Spur-
                                        ;nummer
;
;
; Liegt die gewünschte Recordnummer in einer niedrigeren
; Spur ?
;
track_dec:
;
03E4  79       LD    A,C                ;Ist die Recordnummer
03E5  93       SUB   E                  ; kleiner als der
03E6  78       LD    A,B                ; Spurrecord ?
03E7  9A       SBC   A,D
03E8  D2 03FA  JP    NC,track_inc       ;N: Sprung

03EB  E5       PUSH  HL                 ;Spurnummer retten
03EC  2A 0DC1  LD    HL,(§spt)          ;HL = Anzahl der Re-
                                        ;cords pro Spur
03EF  7B       LD    A,E                ;Spurrecord der
03F0  95       SUB   L                  ; nächstkleineren
03F1  5F       LD    E,A                ; Spur berechnen
03F2  7A       LD    A,D
03F3  9C       SBC   A,H
03F4  57       LD    D,A
03F5  E1       POP   HL                 ;Spurnummer zurück
03F6  2B       DEC   HL                 ;Spurnummer -1
03F7  C3 03E4  JP    track_dec          ;Test wiederholen
;
;
; Liegt die gewünschte Recordnummer in der aktuellen oder
; einer höheren Spur ?
;
track_inc:
;
03FA  E5       PUSH  HL                 ;Spurnummer retten
03FB  2A 0DC1  LD    HL,(§spt)          ;HL = Anzahl der Re-
                                        ;cords pro Spur
03FE  19       ADD   HL,DE              ;Spurrecord der nächs-
                                        ;ten Spur berechnen.
                                        ;Überlauf ?
03FF  DA 040F  JP    C,track_found      ;J: Die höchstmögliche
                                        ;Spur ist erreicht.
                                        ;Die auf dem Stack
                                        ;liegende Spurnummer
                                        ;muß richtig sein.
```

```
0402  79        LD    A,C               ;Ist der Spurrecord
0403  95        SUB   L                 ; der nächsten Spur
0404  78        LD    A,B               ; größer als die ge-
0405  9C        SBC   A,H               ; wünschte Recordnum-
                                        ; mer ?
0406  DA 040F   JP    C,track_found     ;J: Die Recordnummer
                                        ;liegt in der aktu-
                                        ;ellen Spur
0409  EB        EX    DE,HL             ;N: DE - Spurrecord
                                        ;der nächsten Spur
040A  E1        POP   HL                ;Spurnummer zurück
040B  23        INC   HL                ;Spurnummer +1
040C  C3 03FA   JP    track_inc         ;Test wiederholen
;
;
; Richtige Spurnummer erreicht
;
; BC = Gewünschte Recordnummer
; DE = Nummer des ersten Records der berechneten Spur
; HL = Spurnummer
;
track_found:
;
040F  E1        POP   HL                ;Spurnummer vom Stack
                                        ;holen
0410  C5        PUSH  BC                ;Recornummer retten
0411  D5        PUSH  DE                ;Spurrecord retten
0412  E5        PUSH  HL                ;Spurnummer retten
0413  EB        EX    DE,HL             ;DE = Spurnummer
0414  2A 0DCE   LD    HL,(§off)         ;HL = Anzahl der re-
                                        ;servierten Spuren
0417  19        ADD   HL,DE             ;Physikalische Spur-
                                        ;nummer berechnen
0418  44        LD    B,H
0419  4D        LD    C,L               ;BC = phys. Spurnummer
041A  CD 0E1E   CALL  b_settrk          ;BIOS-Aufruf: 'Spur
                                        ;setzen'

041D  D1        POP   DE                ;Spurnummer vom Stack
                                        ;holen
041E  2A 0DB5   LD    HL,(?§track)      ;HL -> DPH + 2
0421  73        LD    (HL),E            ;Spurnummer im DPH
0422  23        INC   HL                ; speichern
0423  72        LD    (HL),D
```

```
0424  D1        POP   DE                ;Neuen Spurrecord vom
                                        ;Stack holen
0425  2A 0DB7   LD    HL,(?§trkrecord)  ;HL -> DPH + 4
0428  73        LD    (HL),E            ;Neuen Spurrecord im
0429  23        INC   HL                ; DPH speichern
042A  72        LD    (HL),D

042B  C1        POP   BC                ;Recordnummer vom
                                        ;Stack holen
042C  79        LD    A,C               ;BC - Recordnummer
042D  93        SUB   E                 ; minus Spurrecord
042E  4F        LD    C,A               ; - Recordnummer
042F  78        LD    A,B               ; innerhalb der
0430  9A        SBC   A,D               ; berechneten Spur
0431  47        LD    B,A
0432  2A 0DD0   LD    HL,(?§xltab)      ;HL -> XLTAB
0435  EB        EX    DE,HL             ;DE -> XLTAB
0436  CD 0E30   CALL  b_sectrn           ;BIOS-Aufruf Sektor-
                                        ;nummer aus der XLTAB
                                        ;holen
0439  4D        LD    C,L
043A  44        LD    B,H               ;BC - Physikalische
                                        ;Sektornummer
043B  C3 0E21   JP    b_setsec          ;BIOS-Aufruf: 'Sektor
                                        ;setzen'
;
;
; Blockposition in der Blocktabelle berechnen
;
get_block_pos:
;
043E  21 0DC3   LD    HL,§bsh           ;HL -> Block Shift
                                        ;Factor
0441  4E        LD    C,(HL)            ;C = BSH
0442  3A 0DE3   LD    A,(§sequ_rec_#)   ;A = Aktuelle Record-
                                        ;nummer innerhalb des
                                        ;aktuellen Extends
```

```
;
; Relative Blocknummer des aktuellen Records (innerhalb des
; Extends) berechnen
;
; = Aktuelle Recordnummer / 2^BSH
;
block_loop_1:
;
0445  B7        OR    A                    ;CY-Flag löschen
0446  1F        RRA                        ;A = A/2
0447  0D        DEC   C                    ;C -1
0448  C2 0445   JP    NZ,block_loop_1      ;Weiter, bis C = 0

044B  47        LD    B,A                  ;B = Relative Block-
                                           ;nummer innerhalb des
                                           ;aktuellen Extends
044C  3E 08     LD    A,8
044E  96        SUB   (HL)
044F  4F        LD    C,A                  ;C = 8 - BSH = Anzahl
                                           ;der Blöcke pro Extend
0450  3A 0DE2   LD    A,(§extend_#)        ;A = Aktuelle Extend-
                                           ;nummer
;
;
; Relative Blocknummer des aktuellen Extends berechnen
; = Aktuelle Extendnummer * Anzahl der Blöcke pro Extend
;
block_loop_2:
;
0453  0D        DEC   C                    ;C -1, C = 0 ?
0454  CA 045C   JP    Z,add_a_b            ;J: Ende der zweiten
                                           ;Schleife

0457  B7        OR    A                    ;CY-Flag löschen
0458  17        RLA                        ;A = A * 2
0459  C3 0453   JP    block_loop_2         ;Weiter, bis C = 0
;
;
add_a_b:
;
045C  80        ADD   A,B                  ;Relative Blocknummer
                                           ;des aktuellen Extends
                                           ;zur relativen Block-
                                           ;nummer des aktuellen
                                           ;Records addieren
                                           ;Dies ergibt die Num-
                                           ;mer des Blockes rela-
                                           ;tiv zum Eintrag und
                                           ;damit die Position
                                           ;der aktuellen Block-
                                           ;nummer in der Block-
                                           ;tabelle
045D  C9        RET
```

```
;
; Aktuelle Blocknummer aus der Blocktabelle holen
;
; I: BC = Position der Blocknummer innerhalb der Block-
;         tabelle
;
get_block_#:
;
045E  2A 0343  LD    HL,(§entry_word)  ;HL -> FCB
0461  11 0010  LD    DE,pos_dx         ;DE = Position der
                                       ;Blocktabelle
0464  19       ADD   HL,DE             ;HL -> Blocktabelle
0465  09       ADD   HL,BC             ;HL -> Blocknummer
0466  3A 0DDD  LD    A,(§large_dsm?)   ;Werden die Block-
0469  B7       OR    A                 ;nummern als 16-Bit
                                       ;Werte verwaltet ?
046A  CA 0471  JP    Z,get_block_word  ;J: 16-Bit Wert holen

046D  6E       LD    L,(HL)            ;N: L = Blocknummer
046E  26 00    LD    H,0               ;HL = Blocknummer
0470  C9       RET
;
;
; 16-Bit Wert aus der Blocktabelle holen
;
get_block_word:
;
0471  09       ADD   HL,BC             ;Position nochmals
                                       ;aufaddieren, da jede
                                       ;Blocknummer zwei
                                       ;Bytes belegt
0472  5E       LD    E,(HL)            ;E = LSB der Block-
0473  23       INC   HL                ; nummer
0474  56       LD    D,(HL)            ;D = MSB
0475  EB       EX    DE,HL             ;HL = Blocknummer
0476  C9       RET
;
;
; Blocknummer des aktuellen Records dem FCB entnehmen und
; in §block_# speichern
;
set_§block_#:
;
0477  CD 043E  CALL  get_block_pos      ;Position der Block-
                                       ;nummer innerhalb der
                                       ;Blocktabelle berech-
                                       ;nen
047A  4F       LD    C,A               ;BC = Position
047B  06 00    LD    B,0
```

```
047D  CD 045E  CALL  get_block_#        ;Blocknummer aus der
                                        ;Blocktabelle holen
0480  22 0DE5  LD    (§block_#),HL     ;und speichern
0483  C9       RET
;
;
; Blocknummer überprüfen
;
; I: §block_#
;
; O: Z-Flag gesetzt, falls §block_# = 0
;
check_§block_#:
;
0484  2A 0DE5  LD    HL,(§block_#)     ;HL - Blocknummer
0487  7D       LD    A,L
0488  B4       OR    H                 ;Flags setzen
0489  C9       RET
;
;
; Absolute Recordnummer des aktuellen Records berechnen
;
; I: §block_#
;    §sequ_rec_#
;
; O: §abs_rec_#
;
set_§record#:
;
048A  3A 0DC3  LD    A,(§bsh)          ;A = Block Shift Fac-
                                        ;tor
048D  2A 0DE5  LD    HL,(§block_#)     ;HL = Blocknummer
;
;
; Absolute Recordnummer des Blockanfangs berechnen
; = Blocknummer * 2^BSH
;
blkrec_loop:
;
0490  29       ADD   HL,HL             ;HL = HL * 2
0491  3D       DEC   A                 ;A -1
0492  C2 0490  JP    NZ,blkrec_loop    ;Weiter, bis A = 0

0495  22 0DE7  LD    (§blkrecord),HL   ;Recordnummer des
                                        ;ersten Records
                                        ;speichern
```

```
0498  3A 0DC4  LD    A,(§blm)           ;A = BLM
049B  4F       LD    C,A                ;C = BLM = Anzahl der
                                        ;Records pro Block -1
049C  3A 0DE3  LD    A,(§sequ_rec_#)    ;A = Aktuelle Record-
                                        ;Nummer
049F  A1       AND   C                  ;modulo Anzahl von Re-
                                        ;cords pro Block
                                        ;= Relative Nummer des
                                        ;Records im Block
04A0  B5       OR    L                  ;+ Absolute Nummer des
04A1  6F       LD    L,A                ;ersten Records im
                                        ;Block
04A2  22 0DE5  LD    (§abs_rec_#),HL    ;ergibt die absolute
                                        ;Nummer des aktuellen
                                        ;Records
04A5  C9       RET
;
;
; Zeiger auf die Extendnummer des FCB berechnen
;
; I: §entry_word -> FCB
;
; O: HL -> Extend-Nummer
;
get_extend_pointer:
;
04A6  2A 0343  LD    HL,(§entry_word)   ;HL -> FCB
04A9  11 000C  LD    DE,pos_ex          ;DE = Extend-Position
04AC  19       ADD   HL,DE              ;HL -> Extend-Nummer
04AD  C9       RET
;
;
; Adressen der höchsten und der aktuellen Recordnummer
; bezogen auf den aktuellen Extend berechnen
;
; I: §entry_word -> FCB
;
; O: DE -> Höchste Recordnummer im Extend
;    HL -> Aktuelle Recordnummer im Extend
;
get_rc_cr_pointer:
;
04AE  2A 0343  LD    HL,(§entry_word)   ;HL -> FCB
04B1  11 000F  LD    DE,pos_rc          ;DE = Position der
                                        ;höchsten Recordnummer
04B4  19       ADD   HL,DE
04B5  EB       EX    DE,HL              ;DE -> Höchste Record-
                                        ;nummer
04B6  21 0011  LD    HL,diff_rc_cr
04B9  19       ADD   HL,DE              ;HL -> Aktuelle Re-
                                        ;cordnummer
04BA  C9       RET
```

```
;
; UPRO für OPEN, READ und WRITE
;
; Recordposition dem FCB entnehmen
;
; I: §entry_word -> FCB
;
; O: §sequ_rec_#  = Nummer des Records innerhalb des Extends
;    §max_ext_rec = Höchste Recordnummer des Extends
;    §extend_#    = Nummer des ersten Extends im FCB
;
get_record_#s:
;
04BB  CD 04AE  CALL  get_rc_cr_pointer ;FCB-Zeiger auf aktu-
                                       ;elle und höchste Re-
                                       ;cordnummer berechnen
04BE  7E       LD    A,(HL)            ;Aktuelle Recordnummer
04BF  32 0DE3  LD    (§sequ_rec_#),A   ;vom FCB holen
04C2  EB       EX    DE,HL             ;HL -> Höchste Record-
                                       ;nummer
04C3  7E       LD    A,(HL)            ;Höchste Recordnummer
04C4  32 0DE1  LD    (§max_ext_rec),A  ;von FCB holen

04C7  CD 04A6  CALL  get_extend_pointer;Adresse der Extend-
                                       ;Nummer berechnen
04CA  3A 0DC5  LD    A,(§exm)          ;A = Extend Mask
04CD  A6       AND   (HL)              ;Nummer des ersten Ex-
04CE  32 0DE2  LD    (§extend_#),A     ;tends berechnen

04D1  C9       RET
;
;
; UPRO für READ und WRITE
;
; Neue Recordposition im FCB speichern
;
update_record_#s:
;
04D2  CD 04AE  CALL  get_rc_cr_pointer ;FCB-Zeiger auf aktu-
                                       ;elle und höchste Re-
                                       ;cordnummer berechnen
04D5  3A 0DD5  LD    A,(§sequential?)  ;A = Zugriffsflag
04D8  FE 02    CP    2                 ;'Write Random with
                                       ;Zero Fill' ?
04DA  C2 04DE  JP    NZ,update_sequ_#  ;N: Zugriffsflag ist
                                       ;Abstand zum nächsten
                                       ;Record
04DD  AF       XOR   A                 ;N: Write Random zählt
                                       ;die Recordnummer
                                       ;nicht weiter
```

```
;
update_sequ_#:
;
04DE  4F        LD    C,A                ;C - Abstand zum
                                         ;nächsten Record
04DF  3A 0DE3   LD    A,(§sequ_rec_#)    ;A - zuletzt bearnbei-
                                         ;tete Recordnummer
04E2  81        ADD   A,C                ;Nächste Recordnummer
                                         ;berechnen
04E3  77        LD    (HL),A             ;und im FCB speichern
04E4  EB        EX    DE,HL              ;HL -> bisher höchste
                                         ;Recordnummer
04E5  3A 0DE1   LD    A,(§max_ext_rec)   ;A = Neue höchste
                                         ;Recordnummer
04E8  77        LD    (HL),A             ;Höchste Recordnummer
                                         ;im FCB aktualisieren
04E9  C9        RET
;
;
; HL um C Bits nach rechts schieben und von links
; 0-Bits nachrücken:
;
; HL = HL / 2^C
;
; I: C  = Anzahl der Schiebeoperationen
;    HL = Anfangswert
;
; O: HL = Anfangswert um C Bits nach rechts geschoben
;
shift_right_hl:
;
04EA  0C        INC   C                  ;C +1, da C auch 0
                                         ;sein darf und daher
                                         ;der Zähler zuerst ge-
                                         ;testet wird
;
shift_right_loop:
;
04EB  0D        DEC   C                  ;Zähler -1, Zähler=0 ?
04EC  C8        RET   Z                  ;J: Fertig

04ED  7C        LD    A,H                ;N: A = MSB
04EE  B7        OR    A                  ;CY-Flag löschen
04EF  1F        RRA                      ;MSB rechts schieben
04F0  67        LD    H,A                ;MSB zurückschreiben
04F1  7D        LD    A,L                ;A = LSB
04F2  1F        RRA                      ;LSB schieben
04F3  6F        LD    L,A                ;LSB zurückschreiben
04F4  C3 04EB   JP    shift_right_loop   ;Weiter, bis Zähler =0
```

```
;
; Prüfsumme des Directory-Buffers berechnen
;
; I: ?§dirbuf -> Directory-Buffer
;
; O: Prüfsumme des Directory-Buffers
;
checksum_dirbuf:
;
04F7  0E 80     LD    C,128              ;128 Bytes aufaddieren
04F9  2A 0DB9   LD    HL,(?§dirbuf)      ;HL -> Buffer
04FC  AF        XOR   A                  ;A = 0
;
checksum_loop:
;
04FD  86        ADD   A,(HL)             ;Nächstes Byte auf-
                                         ;addieren
04FE  23        INC   HL                 ;Bufferzeiger +1
04FF  0D        DEC   C                  ;Zähler -1
0500  C2 04FD   JP    NZ,checksum_loop   ;Weiter, bis Zähler -0
0503  C9        RET
;
;
; HL um C Bits nach links schieben und von rechts
; 0-Bits nachrücken lassen:
;
; HL = HL * 2^C
;
; I: C  = Anzahl der Schiebeoperationen
;    HL = Anfangswert
;
; O: HL = Anfangswert um C Bits nach links geschoben
;
shift_left_hl:
;
0504  0C        INC   C                  ;C +1, da C auch Null
                                         ;sein darf und daher
                                         ;der Zähler zuerst ge-
                                         ;testet wird
;
shift_left_loop:
;
0505  0D        DEC   C                  ;Zähler -1, Zähler =0?
0506  C8        RET   Z                  ;J: Fertig
0507  29        ADD   HL,HL              ;N: HL um 1 Bit nach
                                         ;links schieben
0508  C3 0505   JP    shift_left_loop    ;Weiter, bis Zähler =0
```

```
;
; UPRO für ?select_disk und ?write_protect_disk
; Vektor-Bit setzen, Die Bitnummer ist die Nummer der
; aktuellen Disk
;
; I: BC = Alter Vektor
;
; O: HL = Neuer Vektor mit - entsprechend der Disknummer -
;         gesetztem Bit
;
set_disk_vec:
;
050B  C5        PUSH  BC                  ;Alten Vektor retten
050C  3A 0342   LD    A,(§bdos_disk)      ;A - Aktuelle Disk-
                                          ;nummer
050F  4F        LD    C,A                 ;C - Disknummer
0510  21 0001   LD    HL,1                ;HL - Vektor mit Bit 0
                                          ;gesetzt
0513  CD 0504   CALL  shift_left_hl       ;HL um C Bits nach
                                          ;links schieben ergibt
                                          ;eine Vektormaske mit
                                          ;entsprechend gesetz-
                                          ;tem Bit
0516  C1        POP   BC                  ;Alten Vektor zurück
0517  79        LD    A,C                 ; und Vektormaske
0518  B5        OR    L                   ; einkopieren
0519  6F        LD    L,A
051A  78        LD    A,B
051B  B4        OR    H
051C  67        LD    H,A
051D  C9        RET
;
;
; UPRO für disk_ro?
; Entsprechendes Bit aus dem R/O-Vektor holen
;
; I: -
;
; O: A = 1, falls die Disk schreibgeschützt ist
;    A = 0 sonst
;
get_ro_bit:
;
051E  2A 0DAD   LD    HL,(§ro_vec)        ;HL = R/O-Vektor

0521  3A 0342   LD    A,(§bdos_disk)
0524  4F        LD    C,A                 ;C = Disknummer
0525  CD 04EA   CALL  shift_right_hl      ;HL um C Bits nach
                                          ;rechts schieben
0528  7D        LD    A,L                 ;A = LSB von HL
0529  E6 01     AND   1                   ;Bit 0 maskieren
052B  C9        RET
```

```
;
; **** BDOS-Funktion Nummer 28  'Write Protect Disk'
;
?write_protect_disk:
;
052C  21 0DAD  LD    HL,§ro_vec          ;HL -> R/O-Vektor
052F  4E       LD    C,(HL)
0530  23       INC   HL
0531  46       LD    B,(HL)              ;BC = R/O-Vektor

0532  CD 050B  CALL  set_disk_vec        ;Entsprechendes Bit
                                         ;setzen
0535  22 0DAD  LD    (§ro_vec),HL        ;Neuen Vektor spei-
                                         ;chern

0538  2A 0DC8  LD    HL,(§drm)           ;HL = Anzahl der mög-
053B  23       INC   HL                  ;lichen Directory-Ein-
                                         ;träge +1
053C  EB       EX    DE,HL               ;DE = HL

053D  2A 0DB3  LD    HL,(?§last_entry)   ;HL -> Höchste belegte
                                         ;Eintragsnummer
0540  73       LD    (HL),E              ;Höchste belegte Ein-
0541  23       INC   HL                  ; tragsnummer gleich
0542  72       LD    (HL),D              ; der höchsten mög-
                                         ; lichen Eintragsnum-
                                         ; mer setzen und damit
                                         ; 'Directory voll'
                                         ; vortäuschen.
0543  C9       RET
;
;
; UPRO für WRITE
; Directory-Eintrag auf Schreibschutz testen
;
entry_ro?:
;
0544  CD 055E  CALL  get_entry_pointer ;HL -> Eintrag
;
;
; UPRO für WRITE
; FCB auf Schreibschutz testen
;
fcb_ro?:
;
0547  11 0009  LD    DE,pos_ro           ;DE = Position des
                                         ;R/O-Flags
054A  19       ADD   HL,DE               ;HL -> R/O Flag
054B  7E       LD    A,(HL)              ;Flag holen und
054C  17       RLA                       ;Bit nach CY schieben
                                         ;CY (Bit) gesetzt ?
054D  D0       RET   NC                  ;N: Fertig
                                         ;J: Fehler melden
```

```
;
; 'FILE R/O' melden
;
file_ro_error:
;
054E  21 000F  LD    HL,err_loc_table+6;HL -> Fehlertabelle
0551  C3 034A  JP    error_entry       ;Fehlerroutine auf-
                                        ;rufen
;
;
; UPRO für MAKE, OPEN, CLOSE und DELETE
; Disk auf Schreibschutz testen
;
disk_ro?:
;
0554  CD 051E  CALL  get_ro_bit        ;R/O Bit holen
                                        ;Bit gesetzt ?
0557  C8       RET   Z                 ;N: Fertig
                                        ;J: Fehler melden
;
;
; Disk ist schreibgeschützt, 'R/O' melden
;
disk_ro_error:
;
0558  21 000D  LD    HL,err_loc_table+4;HL -> Fehlertabelle
055B  C3 034A  JP    error_entry       ;Fehlerroutine auf-
                                        ;rufen
;
;
; Adresse des aktuellen Eintrags im DIRBUF berechnen
;
get_entry_pointer:
;
055E  2A 0DB9  LD    HL,(?§dirbuf)     ;HL -> DIRBUF
0561  3A 0DE9  LD    A,(§dir_pos)      ;A = Position des
                                        ;Eintrags im DIRBUF
;
; HL = HL + A
;
add_hl_a:
;
0564  85       ADD   A,L               ;LSBs addieren
0565  6F       LD    L,A               ;LSB setzen
0566  D0       RET   NC                ;Fertig, falls kein
                                        ;Übertrag
0567  24       INC   H                 ;Übertrag in MSB über-
0568  C9       RET                     ;nehmen
```

```
;
; Nummer (und Adresse) der Extendgruppe holen
;
; I: §entry_word -> FCB
;
; O: A  =  Extendgruppen-Nummer bzw. 'NA'-Flag
;    HL -> Extendgruppen-Nummer bzw. 'NA'-Flag
;
get_eg:
;
0569  2A 0343  LD    HL,(§entry_word)  ;HL -> FCB
056C  11 000E  LD    DE,pos_eg         ;DE = Position der Ex-
                                       ;tendgruppennummer
056F  19       ADD   HL,DE             ;HL -> Extendgruppe
0570  7E       LD    A,(HL)            ;Extendgruppe holen
0571  C9       RET
;
;
; Extendgruppen-Nummer auf Null setzen
;
zero_eg:
;
0572  CD 0569  CALL  get_eg            ;Adresse der Extend-
                                       ;gruppennummer holen
0575  36 00    LD    (HL),0           ;Extendgruppe auf Null
                                       ;setzen
0577  C9       RET
;
;
; UPRO für OPEN und MAKE
;
; 'NA'-Flag setzen
; Damit erhält der FCB die Kennung 'File unverändert'
;
not_altered:
;
0578  CD 0569  CALL  get_eg            ;Adresse des Flags
                                       ;holen
057B  F6 80    OR    080H              ;Flag setzen
057D  77       LD    (HL),A            ;und zurückschreiben
057E  C9       RET
```

```
;
; Aktueller Eintrag frei ?
;
; I: §dir_entry_#
;
; O: HL -> ?§last_entry + 1
;    CY-Flag gelöscht, falls die Eintragsnummer größer als
;          die Nummer des höchsten belegten Eintrags ist.
;
entry_#_free?:
;
057F  2A 0DEA  LD    HL,(§dir_entry_#)
0582  EB       EX    DE,HL                ;DE - Eintragsnummer
0583  2A 0DB3  LD    HL,(?§last_entry)    ;HL -> Nummer des
                                          ;höchsten belegten
                                          ;Eintrags +1
0586  7B       LD    A,E                  ;Eintragsnummer mit
0587  96       SUB   (HL)                 ;der Nummer des
0588  23       INC   HL                   ;höchsten belegten
0589  7A       LD    A,D                  ;Eintrags verglei-
058A  9E       SBC   A,(HL)               ;chen und CY-Flag
058B  C9       RET                        ;setzen
;
;
; Nummer des höchsten belegten Eintrags aktualisieren
;
new_end_of_entries?:
;
058C  CD 057F  CALL  entry_#_free?        ;Ist der aktuelle Ein-
                                          ;trag neuer höchster
                                          ;Eintrag ?
058F  D8       RET   C                    ;N: Fertig

0590  13       INC   DE                   ;J: Eintragsnummer + 1
0591  72       LD    (HL),D               ;als neue höchste be-
0592  2B       DEC   HL                   ;belegte Eintragsnum-
0593  73       LD    (HL),E               ;mer + 1 speichern
0594  C9       RET
```

```
;
; Differenz zwischen DE und HL berechnen
; HL = DE - HL
;
sub_de_hl:
;
0595  7B        LD    A,E
0596  95        SUB   L                  ;LSBs subtrahieren
0597  6F        LD    L,A
0598  7A        LD    A,D
0599  9C        SBC   A,H                ;MSBs subtrahieren
059A  67        LD    H,A
059B  C9        RET
;
;
; CSV aktualisieren
;
update_csv:
;
059C  0E FF     LD    C,0FFH             ;C ist Flag für
                                         ;'aktualisieren'
;
;
; CSV testen oder aktualisieren
;
; I: C =  0FFH für CSV aktualisieren
;    C <> 0FFH für CSV testen
;
test_csv:
;
059E  2A 0DEC   LD    HL,(§dir_record_#)
05A1  EB        EX    DE,HL              ;DE = Directory-Re-
                                         ;cordnummer
05A2  2A 0DCC   LD    HL,(§cks)          ;HL = Höchste zu prü-
                                         ;fende Recordnummer
05A5  CD 0595   CALL  sub_de_hl          ;Soll die aktuelle
                                         ;Recordnummer noch ge-
                                         ;prüft werden ?
05A8  D0        RET   NC                 ;N: Fertig

05A9  C5        PUSH  BC                 ;Flag retten
05AA  CD 04F7   CALL  checksum_dirbuf    ;Prüfsumme berechnen
05AD  2A 0DBD   LD    HL,(?§csv)
05B0  EB        EX    DE,HL              ;DE -> CSV
05B1  2A 0DEC   LD    HL,(§dir_record_#);HL = Directory-Re-
                                         ;cordnummer
05B4  19        ADD   HL,DE              ;HL -> Prüfsumme im
                                         ;CSV
05B5  C1        POP   BC                 ;Flag zurück
05B6  0C        INC   C                  ;CSV aktualisieren?
05B7  CA 05C4   JP    Z,update_csv_byte  ;J: Neue Prüfsumme
                                         ;eintragen
```

```
05BA  BE        CP    (HL)                ;N: Neue Prüfsumme
                                          ;gleich dem Vektor-
                                          ;eintrag ?
05BB  C8        RET   Z                   ;J: Ok, fertig

05BC  CD 057F   CALL  entry_#_free?       ;N: Ist der aktuelle
                                          ;Eintrag im freien Be-
                                          ;reich ?
05BF  D0        RET   NC                  ;J: Der aktuelle Ein-
                                          ;trag wurde neu ange-
                                          ;legt, Prüfsummendif-
                                          ;ferenz ist in Ordnung
                                          ;N: Disk vorsichtshal-
                                          ;ber schreibschützen
05C0  CD 052C   CALL  ?write_protect_disk
05C3  C9        RET
;
;
; CSV aktualisieren
;
update_csv_byte:
;
05C4  77        LD    (HL),A              ;Neue Prüfsumme in den
                                          ;CSV eintragen
05C5  C9        RET
;
;
; Directory-Record schreiben
;
dir_write:
;
05C6  CD 059C   CALL  update_csv          ;CSV aktualisieren
05C9  CD 05E0   CALL  set_dma_to_dirbuf   ;DMA-Adresse des BIOS
                                          ;= DIRBUF setzen
05CC  0E 01     LD    C,1                 ;Flag für BIOS-Aufruf
05CE  CD 03B8   CALL  write_record        ;Record schreiben
05D1  C3 05DA   JP    restore_dma         ;und BIOS-DMA-Adresse
                                          ;wieder auf den alten
                                          ;Wert setzen
;
;
; Directory-Record lesen
;
dir_read:
;
05D4  CD 05E0   CALL  set_dma_to_dirbuf   ;BIOS-DMA-Adresse
                                          ;= DIRBUF setzen
05D7  CD 03B2   CALL  read_record         ;Record lesen
```

```
;
; BIOS-DMA-Adresse auf BDOS-DMA-Adresse setzen
;
restore_dma:
;
05DA  21 0DB1  LD    HL,§dma_address   ;HL -> BDOS-DMA-
                                       ;Adresse
05DD  C3 05E3  JP    set_dma_to_hl     ;BIOS-DMA-Adresse
                                       ;= (HL) setzen
;
;
; BIOS-DMA-Adresse = DIRBUF setzen
;
set_dma_to_dirbuf:
;
05E0  21 0DB9  LD    HL,?§dirbuf       ;HL -> DIRBUF-Adresse
;
;
; BIOS-DMA-Adresse = (HL) setzen
;
set_dma_to_hl:
;
05E3  4E       LD    C,(HL)
05E4  23       INC   HL
05E5  46       LD    B,(HL)            ;BC = (HL)
05E6  C3 0E24  JP    b_setdma          ;BIOS-Aufruf
;
;
; UPRO für 'Search for First'
; DIRBUF zum aktuellen DMA-Bereich kopieren
;
move_dirbuf_to_dma:
;
05E9  2A 0DB9  LD    HL,(?§dirbuf)
05EC  EB       EX    DE,HL             ;DE -> DIRBUF
05ED  2A 0DB1  LD    HL,(§dma_address) ;HL -> DMA-Bereich
05F0  0E 80    LD    C,128             ;128 Bytes
05F2  C3 034F  JP    move_de_hl        ;von DE nach HL ko-
                                       ;pieren
```

```
;
; UPRO für Directory-Operationen
;
; §dir_entry_# = Startwert (0FFFFH) ?
;
; I: §dir_entry_#
;
; O: A =  0, falls §dir_entry_# = 0FFFFH
;    A <> 0 sonst
;
all_entries_compared?:
;
05F5  21 0DEA  LD    HL,§dir_entry_#   ;HL -> §dir_entry_#
05F8  7E       LD    A,(HL)            ;A = LSB
05F9  23       INC   HL
05FA  BE       CP    (HL)              ;MSB = LSB ?
05FB  C0       RET   NZ                ;N: Fertig

05FC  3C       INC   A                 ;LSB = MSB = 0FFH ?
05FD  C9       RET                     ;J: Z-Flag gesetzt
;
;
; UPRO für Directory-Operationen
;
; §dir_entry_# auf Startwert (0FFFFH) setzen
;
reset_entry_#:
;
05FE  21 FFFF  LD    HL,0FFFFH         ;HL = 0FFFFH
0601  22 0DEA  LD    (§dir_entry_#),HL ;§dir_entry_# = HL
0604  C9       RET
;
;
; UPRO für logical_select
; Nächsten Directory Record prüfen
;
; I: C = 00H, falls der Record geprüft werden soll
;    C = FFH, falls die Prüfsumme neu gebildet werden soll
;
;
check_dir_record:
;
0605  2A 0DC8  LD    HL,(§drm)
0608  EB       EX    DE,HL             ;DE = Höchste Ein-
                                       ;tragsnummer
0609  2A 0DEA  LD    HL,(§dir_entry_#) ;HL = aktuelle Ein-
                                       ;tragsnummer
060C  23       INC   HL                ;Eintragsnummer +1
060D  22 0DEA  LD    (§dir_entry_#),HL ;Neue Nummer speichern
0610  CD 0595  CALL  sub_de_hl         ;Ist die neue Nummer
                                       ;größer als die
                                       ;höchste Nummer ?
```

```
0613  D2 0619  JP    NC,check_dir_pos  ;N: §dir_pos und
                                       ;Prüfsumme berechnen
0616  C3 05FE  JP    reset_entry_#     ;J: Eintragsnummer auf
                                       ;Startwert setzen
;
;
; §dir_pos berechnen und bei Erreichen eines neuen
; Directory-Records die neue Prüfsumme berechnen.
;
check_dir_pos:
;
0619  3A 0DEA  LD    A,(§dir_entry_#)  ;A = LSB der Eintrags-
                                       ;nummer
061C  E6 03    AND   3                 ;A = Nummer mod 4
061E  06 05    LD    B,5               ;Nun A * 2^5 berechnen
;
dir_pos_loop:
;
0620  87       ADD   A,A               ;A = A * 2
0621  05       DEC   B                 ;B -1
0622  C2 0620  JP    NZ,dir_pos_loop   ;Weiter, bis B = 0

0625  32 0DE9  LD    (§dir_pos),A      ;§dir_pos speichern
0628  B7       OR    A                 ;A = 0 ?
0629  C0       RET   NZ                ;N: Fertig
                                       ;J: Es wurde ein neuer
                                       ;Directory Record er-
                                       ;reicht
062A  C5       PUSH  BC                ;Flag (C) retten
062B  CD 03C3  CALL  set_dir_trk_sec   ;Directory Track und
                                       ;Sektor anwählen
062E  CD 05D4  CALL  dir_read          ;und Record lesen
0631  C1       POP   BC                ;Flag zurück
0632  C3 059E  JP    test_csv          ;Prüfsumme testen bzw.
                                       ;neu berechnen
```

```
;
; UPRO für WRITE
;
; Belegungsstatus der übergebenen Blocknummer feststellen.
; Zurückgegeben wird  das der Blocknummer entsprechende Bit
; aus dem Belegungsvektor (ALV).
;
; I: BC   = Blocknummer
;
; O: A,0 = 0, falls der Block noch frei ist
;         = 1, falls der Block belegt ist
;    D    = Bitnummer (siehe change_alv_bit)
;
get_alv_bit:
;
0635  79        LD    A,C          ;A = LSB der Block-
0636  E6 07     AND   7            ; nummer modulo 8
0638  3C        INC   A            ; + 1 = Bitnummer im
                                   ; Vektor-Byte
0639  5F        LD    E,A          ;E = Bitnummer
063A  57        LD    D,A          ;D = Bitnummer
063B  79        LD    A,C          ;A = LSB der Block-
063C  0F        RRCA               ; nummer geteilt
063D  0F        RRCA               ; durch 8
063E  0F        RRCA
063F  E6 1F     AND   01FH         ;Die höchsten 3 Bits
                                   ;löschen, da gerade
                                   ;rotiert (!) wurde
0641  4F        LD    C,A          ;Wert in C speichern
0642  78        LD    A,B          ;A = MSB der Block-
0643  87        ADD   A,A          ;nummer *  32
0644  87        ADD   A,A          ;(Dadurch werden die
0645  87        ADD   A,A          ; unteren 3 Bits des
0646  87        ADD   A,A          ; MSB nach 'oben' ge-
0647  87        ADD   A,A          ; schoben)
0648  B1        OR    C            ;Diese 3 Bits in C
                                   ;einmaskieren
0649  4F        LD    C,A          ;C = LSB der Block-
                                   ;nummer / 8

064A  78        LD    A,B          ;A = MSB der Block-
064B  0F        RRCA               ; nummer / 8
064C  0F        RRCA
064D  0F        RRCA
064E  E6 1F     AND   01FH         ;Obere 3 Bits wieder
                                   ;löschen
0650  47        LD    B,A          ;und B = MSB der
                                   ;Blocknummer / 8
                                   ;setzen
```

```
0651  2A 0DBF  LD    HL,(?Salv)          ;HL -> ALV-Vektor
0654  09        ADD   HL,BC               ;+ Blocknummer / 8

0655  7E        LD    A,(HL)              ;Byte aus dem ALV-
                                          ;Vektor holen und
                                          ;jetzt noch das der
                                          ;Blocknummer ent-
                                          ;sprechende Bit nach
                                          ;A,0 bringen:
;
get_alv_loop:
;
0656  07        RLCA                      ;A links rotieren
0657  1D        DEC   E                   ;Bitzähler -1
0658  C2 0656   JP    NZ,get_alv_loop     ;Weiter, bis Zähler =0
065B  C9        RET
;
;
; UPRO für WRITE
;
; Belegungsvektor (ALV) aktualisieren
;
; I: BC  = Blocknummer, dessen Status geändert werden soll
;    E   = Neues Bit für den Belegungsvektor (00H oder 01H)
;
change_alv_bit:
;
065C  D5        PUSH  DE                  ;Neues Bit retten
065D  CD 0635   CALL  get_alv_bit         ;Vektoreintrag holen

0660  E6 FE     AND   0FEH                ;Bit löschen
0662  C1        POP   BC                  ;Neues Bit zurück
0663  B1        OR    C                   ;und einmaskieren
;
set_alv_loop:
;
0664  0F        RRCA                      ;Bit nach rechts
                                          ;schieben
0665  15        DEC   D                   ;Bitzähler -1
                                          ;Alte Bitposition
                                          ;wieder erreicht ?
0666  C2 0664   JP    NZ,set_alv_loop     ;N: Weiter

0669  77        LD    (HL),A              ;J: ALV-Eintrag zu-
                                          ;rückschreiben
066A  C9        RET
```

```
;
; UPRO für logical_select und DELETE
;
; Blocktabelle des Directory-Eintrags auswerten und die
; Belegungsvektor (ALV) aktualisieren.
;
; I: Eintrag im DIRBUF, §dir_pos entsprechend gesetzt
;    C = 00H, für Blocknummern freigeben (bei DELETE)
;    C = 01H, für Blocknummern als 'belegt' markieren
;             (bei logical_select)
;
; O: ALV entsprechend aktualisiert
;
update_alv:
;
066B  CD 055E  CALL  get_entry_pointer ;HL -> Eintrag
066E  11 0010  LD    DE,pos_dx        ;DE = Position der
                                      ;Blocktabelle
0671  19       ADD   HL,DE            ;HL -> Blocktabelle
0672  C5       PUSH  BC               ;Flag (C) retten

0673  0E 11    LD    C,17             ;C = Maximale Anzahl
                                      ;von Blöcken in der
                                      ;Blocktabelle +1
;
update_alv_loop:
;
0675  D1       POP   DE               ;Flag zurück nach E
0676  0D       DEC   C                ;Blockzähler -1
                                      ;Zähler = 0 ?
0677  C8       RET   Z                ;J: Fertig

0678  D5       PUSH  DE               ;N: Flag retten
0679  3A 0DDD  LD    A,(§large_dsm?)  ;Sind die Blocknummern
067C  B7       OR    A                ;in der Blocktabelle
                                      ;16-Bit Werte ?
067D  CA 0688  JP    Z,word_value     ;J: 16-Bit Wert holen

0680  C5       PUSH  BC               ;N: Blockzähler retten
0681  E5       PUSH  HL               ;Tabellenzeiger retten
0682  4E       LD    C,(HL)           ;8-Bit Blocknummer
0683  06 00    LD    B,0              ; holen
0685  C3 068E  JP    test_block_#     ;Blocknummer testen
```

```
;
; 16-Bit Blocknummer aus der Blocktabelle holen
;
word_value:
;
0688  0D        DEC   C                   ;Blockzähler -1
0689  C5        PUSH  BC                  ;Blockzähler retten
068A  4E        LD    C,(HL)              ;LSB der Blocknummer
068B  23        INC   HL                  ; holen
068C  46        LD    B,(HL)              ;MSB der Blocknummer
068D  E5        PUSH  HL                  ;holen und Tabellen-
                                          ;zeiger retten
;
;
; Blocktabelleneintrag testen
;
test_block_#:
;
068E  79        LD    A,C                 ;Ist die Blocknummer
068F  B0        OR    B                   ;gleich Null ?
0690  CA 069D   JP    Z,no_block_alloc    ;J: Der Tabelleneintrag
                                          ;trag ist unbelegt
0693  2A 0DC6   LD    HL,(§dsm)           ;N: HL = maximale
                                          ;Blocknummer
0696  7D        LD    A,L                 ;Ist die Blocknummer
0697  91        SUB   C                   ;kleiner oder gleich
0698  7C        LD    A,H                 ;der maximalen Block-
0699  98        SBC   A,B                 ;nummer ?
069A  D4 065C   CALL  NC,change_alv_bit   ;J: ALV anpassen
;
;
; Nächsten Blocktabelleneintrag bearbeiten
;
no_block_alloc:
;
069D  E1        POP   HL                  ;Tabellenzeiger zurück
069E  23        INC   HL                  ;Tabellenzeiger +1
069F  C1        POP   BC                  ;Blockzähler zurück
06A0  C3 0675   JP    update_alv_loop     ;Nächsten Tabelleneintrag
                                          ;trag bearbeiten
```

```
;
; UPPO für ?select_disk
;
; Belegungs- und Prüfsummenvektor (ALV und CSV) neu setzen
;
logical_select:
;
06A3  2A 0DC6  LD    HL,(§dsm)          ;HL = Maximale Block-
                                        ;nummer
06A6  0E 03    LD    C,3
06A8  CD 04EA  CALL  shift_right_hl     ;HL = Maximale Block-
                                        ;nummer / 2^3
06AB  23       INC   HL                 ; +1
06AC  44       LD    B,H                ;BC = (DSM / 8) + 1
06AD  4D       LD    C,L                ; = Länge des ALV
06AE  2A 0DBF  LD    HL,(?§alv)         ;HL -> ALV
;
; Belegungsvektor komplett löschen
;
clr_alv_loop:
;
06B1  36 00    LD    (HL),0             ;Byte löschen
06B3  23       INC   HL                 ;Zeiger +1
06B4  0B       DEC   BC                 ;Zähler -1
06B5  78       LD    A,B                ;Zähler = 0 ?
06B6  B1       OR    C
06B7  C2 06B1  JP    NZ,clr_alv_loop    ;N: Weiter

06BA  2A 0DCA  LD    HL,(§al0_al1)
06BD  EB       EX    DE,HL              ;DE = Erste zwei Bytes
                                        ;des neuen ALV
                                        ;(von der Directory
                                        ;belegte Blöcke)
06BE  2A 0DBF  LD    HL,(?§alv)         ;HL -> ALV
06C1  73       LD    (HL),E             ;Erstes
06C2  23       INC   HL                 ;und
06C3  72       LD    (HL),D             ;zweites Byte ein-
                                        ;tragen
06C4  CD 03A1  CALL  home_disk          ;DPH-Parameter
                                        ;initialisieren

06C7  2A 0DB3  LD    HL,(?§last_entry)  ;Nummer des höchsten
06CA  36 03    LD    (HL),3             ;belegten Eintrags+1
06CC  23       INC   HL                 ;auf 3 setzen
06CD  36 00    LD    (HL),0

06CF  CD 05FE  CALL  reset_entry_#      ;Eintragsnummer auf
                                        ;Startwert setzen
```

```
;
login_next_entry:
;
06D2  0E FF     LD    C,0FFH              ;C = Flag für CSV
                                          ;aktualisieren
06D4  CD 0605   CALL  check_dir_record    ;Nächsten Eintrag in
                                          ;den CSV übernehmen
                                          ;Alle Einträge über-
                                          ;nommen ?
06D7  CD 05F5   CALL  all_entries_compared?
06DA  C8        RET   Z                   ;J: Fertig

06DB  CD 055E   CALL  get_entry_pointer   ;N: Adresse des gerade
                                          ;übernommenen Eintrags
                                          ;holen
06DE  3E E5     LD    A,0E5H
06E0  BE        CP    (HL)                ;Ist der Eintrag be-
                                          ;legt ?
06E1  CA 06D2   JP    Z,login_next_entry  ;N: Nächsten Eintrag
                                          ;holen
06E4  3A 0341   LD    A,(&user_#)         ;J: Gehört der Eintrag
06E7  BE        CP    (HL)                ;zum aktuellen Userbe-
                                          ;reich ?
06E8  C2 06F6   JP    NZ,entry_not_right  ;N: weiter

06EB  23        INC   HL                  ;J: HL -> Filename
06EC  7E        LD    A,(HL)              ;A = Erstes Zeichen
                                          ;des Filenamens
06ED  D6 24     SUB   '$'                 ;Ist das erste Zeichen
                                          ;ein '$' ?
06EF  C2 06F6   JP    NZ,entry_not_right  ;N: weiter

06F2  3D        DEC   A                   ;J: A = 0FFH
06F3  32 0345   LD    (&exit_value),A     ;und A als Rückgabe-
                                          ;parameter speichern
                                          ;(siehe CCP, 'submit_
                                          ;flag')
;
entry_not_right:
;
06F6  0E 01     LD    C,1                 ;C = Flag für 'ALV-
                                          ;setzen'
06F8  CD 066B   CALL  update_alv          ;ALV aktualisieren
                                          ;Höchste belegte Ein-
                                          ;tragsnummer bestimmen
06FB  CD 058C   CALL  new_end_of_entries?
06FE  C3 06D2   JP    login_next_entry    ;Nächsten Eintrag
                                          ;übernehmen
```

```
;
; UPRO für 'Delete', 'Rename' und 'Set File Attr'
;
; Parameter zurückgeben
;
return_found_flag:
;
0701  3A 0DD4  LD    A,(§file_found?)  ;A = Flag von Open,
                                       ;ob das File gefunden
                                       ;wurde
0704  C3 0301  JP    return_byte       ;Flag als Byte-Wert
                                       ;zurückgeben
;
;
; UPRO für 'Search for First' und 'Search for Next'
;
; Extendnummer überprüfen
;
; I: A = 'Gewünschte' Extendnummer (aus dem FCB)
;    C = 'Gefundene' Extendnummer (aus dem Directory-Eintrag)
;
; O: Z-Flag gesetzt, falls der gewünschte Extend vom Eintrag
;                    abgedeckt wird
;
check_extend_#:
;
0707  C5       PUSH  BC                 ;Gefundene und
0708  F5       PUSH  AF                 ;gewünschte Extend-
                                        ;nummer retten
0709  3A 0DC5  LD    A,(§exm)           ;A = Extend Mask
070C  2F       CPL                      ;Maske negieren
070D  47       LD    B,A                ;B = Maske

070E  79       LD    A,C                ;A = Extendnummer
070F  A0       AND   B                  ;Extendnummer maskie-
0710  4F       LD    C,A                ;ren und speichern

0711  F1       POP   AF                 ;Gewünschte Extend-
                                        ;nummer zurück
0712  A0       AND   B                  ;Ebenfalls maskieren
0713  91       SUB   C                  ;Sind beide Extend-
                                        ;nummern im gleichen
                                        ;Bereich ?
0714  E6 1F    AND   01FH               ;Über/Unterlauf ver-
                                        ;nachlässigen
0716  C1       POP   BC                 ;gefundene Extendnum-
0717  C9       RET                      ;mer zurück
```

```
;
; UPRO für 'Open' und 'Search for First'
;
; Ersten passenden Directory-Eintrag suchen
;
; I: §entry_word -> FCB
;    C = Anzahl der Bytes, in denen der Eintrag und der FCB
;        übereinstimmen müssen
;
; O: §file_found? = 00H, falls ein passender Eintrag gefunden
;                   wurde
;                 = 0FFH, sonst
;
;    §exit_value  = Directory-Code (0 - 3), falls ein Eintrag
;                   gefunden wurde
;                 = 0FFH, sonst
;
search_dir_first:
;
0718  3E FF     LD    A,0FFH              ;Noch wurde kein pas-
071A  32 0DD4   LD    (§file_found?),A    ;sender Eintrag gefun-
                                          ;den
071D  21 0DD8   LD    HL,§compare_count   ;HL -> Vergleichs-
                                          ;zähler
0720  71        LD    (HL),C              ;Zähler vorbesetzen
0721  2A 0343   LD    HL,(§entry_word)    ;HL -> FCB
0724  22 0DD9   LD    (?§search_fcb),HL   ;FCB-Zeiger für
                                          ;'search_dir_next'
                                          ;speichern
0727  CD 05FE   CALL  reset_entry_#       ;Eintragsnummer auf
                                          ;Startwert setzen
072A  CD 03A1   CALL  home_disk           ;DPH-Parameter initi-
                                          ;alisieren
;
;
; UPRO für 'Open', 'Search for First' und 'Search for Next'
;
; Nächsten passenden Directory-Eintrag suchen
;
; I: ?§search_fcb   -> FCB (vom letzten 'search_dir_first')
;    §compare_count =  Anzahl der zu überprüfenden Zeichen
;
search_dir_next:
;
072D  0E 00     LD    C,0                 ;C = Flag für CSV
                                          ;überprüfen
072F  CD 0605   CALL  check_dir_record    ;Eintragsnummer erhö-
                                          ;hen und CSV testen
                                          ;Letzten Eintrag er-
                                          ;reicht ?
0732  CD 05F5   CALL  all_entries_compared?
```

```
0735  CA 0794  JP    Z,entry_not_found ;J: Eintrag nicht ge-
                                       ;funden
0738  2A 0DD9  LD    HL,(?§search_fcb)
073B  EB       EX    DE,HL             ;DE -> Zu suchender
                                       ;FCB
073C  1A       LD    A,(DE)            ;A = Drivecode des FCB
073D  FE E5    CP    0E5H              ;Drivecode - 0E5H ?
073F  CA 074A  JP    Z,search_make     ;J: Der Aufruf kam von
                                       ;'Make File', es soll
                                       ;nur ein freier Ein-
                                       ;trag gesucht werden.

0742  D5       PUSH  DE                ;FCB-Zeiger retten
0743  CD 057F  CALL  entry_#_free?     ;Folgen nur noch freie
                                       ;Einträge ?
0746  D1       POP   DE                ;FCB-Zeiger zurück
0747  D2 0794  JP    NC,entry_not_found;J: Eintrag nicht ge-
                                       ;funden
;
search_make:
;
074A  CD 055E  CALL  get_entry_pointer ;Adresse des aktuellen
                                       ;Eintrags holen
074D  3A 0DD8  LD    A,(§compare_count)
0750  4F       LD    C,A               ;C - Vergleichszähler
0751  06 00    LD    B,0               ;B = Position des
                                       ;nächsten zu verglei-
                                       ;chenden Bytes
;
compare_next_loop:
;
0753  79       LD    A,C               ;A = Zähler
0754  B7       OR    A                 ;Alle Bytes ver-
                                       ;glichen ?
0755  CA 0783  JP    Z,return_dir_code ;J: Eintrag gefunden

0758  1A       LD    A,(DE)            ;N: Nächstes Byte
                                       ;aus dem FCB holen
0759  FE 3F    CP    '?'               ;'Joker' ?
                                       ;J: Byte nicht ver-
                                       ;gleichen
075B  CA 077C  JP    Z,compare_next_char

075E  78       LD    A,B               ;N: A = aktuelle
                                       ;Position
075F  FE 0D    CP    pos_s1            ;S1 erreicht ?
                                       ;J: Byte nicht ver-
                                       ;gleichen
0761  CA 077C  JP    Z,compare_next_char
```

```
0764  FE 0C     CP    pos_ex              ;Extendnummer er-
                                          ;reicht ?
0766  1A        LD    A,(DE)              ;A = Extendnummer aus
                                          ;dem FCB
0767  CA 0773   JP    Z,compare_extend    ;J: Extendnummern ver-
                                          ;gleichen

076A  96        SUB   (HL)                ;N: Stimmen die Bytes
                                          ;überein ?
076B  E6 7F     AND   07FH                ;Bit 7 interessiert
                                          ;nicht
076D  C2 072D   JP    NZ,search_dir_next  ;N: Nächsten Eintrag
                                          ;überprüfen
0770  C3 077C   JP    compare_next_char   ;J: Nächstes Zeichen
                                          ;vergleichen
;
;
; Extendnummern vergleichen
;
compare_extend:
;
0773  C5        PUSH  BC                  ;Zähler retten
0774  4E        LD    C,(HL)              ;C = Extendnummer des
                                          ;Eintrags
0775  CD 0707   CALL  check_extend_#      ;Wird der gewünschte
                                          ;Extend vom gefundenen
                                          ;Eintrag abgedeckt ?
0778  C1        POP   BC                  ;Zähler zurück
0779  C2 072D   JP    NZ,search_dir_next  ;N: Nächsten Eintrag
                                          ;überprüfen
                                          ;J: Nächstes Zeichen
                                          ;vergleichen
;
compare_next_char:
;
077C  13        INC   DE                  ;FCB-Zeiger +1
077D  23        INC   HL                  ;Eintrag-Zeiger +1
077E  04        INC   B                   ;Position +1
077F  0D        DEC   C                   ;Zähler -1
0780  C3 0753   JP    compare_next_loop   ;Weiter, bis Zähler =0
;
;
; Eintrag gefunden
;
; Directory-Code berechnen und zurückgeben
;
return_dir_code:
;
0783  3A 0DEA   LD    A,(§dir_entry_#)    ;A = LSB der Eintrags-
0786  E6 03     AND   03H                 ;nummer modulo 4
0788  32 0345   LD    (§exit_value),A     ;Dies ist der Direc-
                                          ;tory-Code
```

```
078B  21 0DD4  LD    HL,§file_found?   ;HL -> Flag
078E  7E       LD    A,(HL)            ;A = Letzter Wert
078F  17       RLA                     ;Bit 7 = 1 ?
0790  D0       RET   NC                ;N: Es war nicht der
                                       ;erste Versuch, es
                                       ;wurde schon vorher
                                       ;ein Eintrag gefunden
0791  AF       XOR   A                 ;J: A = 0
0792  77       LD    (HL),A            ;Flag auf 'Eintrag
                                       ;vorhanden' setzen
0793  C9       RET
;
;
; Keinen passenden Eintrag gefunden
;
entry_not_found:
;
0794  CD 05FE  CALL  reset_entry_#     ;Eintragsnummer auf
                                       ;Startwert setzen
0797  3E FF    LD    A,0FFH            ;und Fehlercode 0FFH
0799  C3 0301  JP    return_byte       ;zurückgeben
;
;
; UPRO für 'Delete File'
;
; Alle passenden Einträge löschen
;
delete_files:
;
079C  CD 0554  CALL  disk_ro?          ;Disk schreibge-
                                       ;schützt ?
                                       ;J: 'R/O' melden
079F  0E 0C    LD    C,12              ;N: Es müssen nur die
                                       ;ersten 12 Zeichen
                                       ;(Usernummer, Filename
                                       ;und Filetype) über-
                                       ;einstimmen
07A1  CD 0718  CALL  search_dir_first  ;Ersten passenden Ein-
                                       ;trag suchen
;
delete_loop:
;
                                       ;Passenden Eintrag ge-
                                       ;funden ?
07A4  CD 05F5  CALL  all_entries_compared?
07A7  C8       RET   Z                 ;N: Fertig
```

```
07A8  CD 0544  CALL  entry_ro?          ;J: Eintrag schreibge-
                                        ;schützt ?
07AB  CD 055E  CALL  get_entry_pointer  ;N: Zeiger auf den ge-
                                        ;fundenen Eintrag
                                        ;holen
07AE  36 E5    LD    (HL),0E5H          ;Eintrag auf 'frei'
                                        ;setzen
07B0  0E 00    LD    C,0                ;C = 0
07B2  CD 066B  CALL  update_alv         ;Alle vom Eintrag be-
                                        ;legten Blöcke aus dem
                                        ;ALV löschen
07B5  CD 05C6  CALL  dir_write          ;Eintrag auf Disk zu-
                                        ;rückschreiben
07B8  CD 072D  CALL  search_dir_next    ;Nächsten passenden
                                        ;Eintrag suchen
07BB  C3 07A4  JP    delete_loop       ;und löschen
;
;
; UPRO für 'Write'
;
; Nächstgelegenen freien Block suchen und belegen
;
; I: BC - Nummer des zuletzt belegten Blocks
;
; O: HL - 0, falls kein Block mehr frei ist
;       - Nummer des neu belegten Blocks sonst
;
allocate_next_block:
;
07BE  50       LD    D,B
07BF  59       LD    E,C                ;DE = BC
;
; Im folgenden ist BC die Nummer des nächst-niedrigeren und
; DE die Nummer des nächst-höheren Blocks, jeweils gemessen
; zu der Nummer des zuletzt belegten Blocks.
; Dadurch wird erreicht, daß ein File möglichst immer zusam-
; menhängende Teile auf der Diskette belegt.
;
allocate_next_loop:
;
07C0  79       LD    A,C                ;Ist noch ein nied-
07C1  B0       OR    B                  ;rigerer Block vor-
                                        ;handen ?
07C2  CA 07D1  JP    Z,allocate_higher  ;N: Höhere Blöcke
                                        ;überprüfen
07C5  0B       DEC   BC                 ;J: BC = Nummer des
                                        ;nächst-niedrigeren
                                        ;Blocks
```

```
07C6  D5        PUSH  DE                  ;Höhere und
07C7  C5        PUSH  BC                  ;niedrigere Blocknum-
                                          ;mer retten
07C8  CD 0635   CALL  get_alv_bit         ;Bit aus dem ALV holen
07CB  1F        RRA                       ;Ist der Block frei ?
07CC  D2 07EC   JP    NC,allocate_block   ;J: Block belegen

07CF  C1        POP   BC                  ;N: Niedrigere und
07D0  D1        POP   DE                  ;höhere Blocknummer
                                          ;zurück
;
; Ist der nächst-höhere Block frei ?
;
allocate_higher:
;
07D1  2A 0DC6   LD    HL,(§dsm)           ;HL = Nummer des
                                          ;höchsten Blocks
07D4  7B        LD    A,E                 ;Nächst-höhere Block-
07D5  95        SUB   L                   ;nummer mit der
07D6  7A        LD    A,D                 ;höchsten Blocknummer
07D7  9C        SBC   A,H                 ;vergleichen. Sind
                                          ;noch höhere Blöcke
                                          ;vorhanden ?
                                          ;N: 'Oberes Ende' der
                                          ;Diskette erreicht:
07D8  D2 07F4   JP    NC,allocate_lower   ;'Untere' Blöcke
                                          ;testen

07DB  13        INC   DE                  ;J: DE = Nächst-
                                          ;höhere Blocknummer
07DC  C5        PUSH  BC                  ;Niedrigere und
07DD  D5        PUSH  DE                  ;höhere Blocknummer
                                          ;retten
07DE  42        LD    B,D                 ;BC = Nummer des zu
07DF  4B        LD    C,E                 ;testenden Blocks
07E0  CD 0635   CALL  get_alv_bit         ;ALV-Bit holen
07E3  1F        RRA                       ;Ist der Block frei ?
07E4  D2 07EC   JP    NC,allocate_block   ;J: Block belegen

07E7  D1        POP   DE                  ;N: Höhere und
07E8  C1        POP   BC                  ;niedrigere Block-
                                          ;nummer zurück
07E9  C3 07C0   JP    allocate_next_loop  ;und weitertesten
```

```
;
; Gefundenen Block belegen
;
; Die Blocknummer liegt zuoberst auf dem Stack
;
allocate_block:
;
07EC  17        RLA                     ;RRA vom Test zurück-
                                        ;nehmen
07ED  3C        INC   A                 ;Bit 0 setzen
07EE  CD 0664   CALL  set_alv_loop      ;und ALV-Eintrag zu-
                                        ;rückschreiben
07F1  E1        POP   HL                ;Blocknummer vom
                                        ;Stack nehmen
07F2  D1        POP   DE                ;zweite Blocknummer
                                        ;ebenfalls vom Stack
                                        ;nehmen
07F3  C9        RET
;
;
; Es sind keine höheren Blöcke mehr frei
;
allocate_lower:
;
07F4  79        LD    A,C               ;Sind noch niedrigere
07F5  B0        OR    B                 ;Blöcke vorhanden ?
                                        ;J: Weitertesten
07F6  C2 07C0   JP    NZ,allocate_next_loop

07F9  21 0000   LD    HL,0              ;N: Es konnte kein
                                        ;freier Block mehr
                                        ;gefunden werden
07FC  C9        RET
;
;
; UPRO für 'Make File'
;
; Kompletten Directory-Eintrag neu anlegen
;
; I: §entry_word -> FCB des neuen Eintrags
;
write_complete_entry:
;
07FD  0E 00     LD    C,0               ;FCB ab Position 0 in
                                        ;den Eintrag überneh-
                                        ;men
07FF  1E 20     LD    E,32              ;32 Bytes des FCB ab
                                        ;der Startposition
                                        ;übernehmen
```

```
;
; UPRO für 'Rename File' und 'Set File Attr'
;
; Directory-Eintrag teilweise neu anlegen
;
; I: §entry_word -> FCB
;    C - Position, ab der der FCB in den Eintrag übernommen
;        werden soll
;    E - Anzahl der Bytes die übernommen werden sollen
;
write_partial_entry:
;
0801  D5        PUSH  DE                  ;Anzahl retten
0802  06 00     LD    B,0                 ;BC - Startposition

0804  2A 0343   LD    HL,(§entry_word)    ;HL -> FCB
0807  09        ADD   HL,BC
0808  EB        EX    DE,HL               ;DE -> Startposition
0809  CD 055E   CALL  get_entry_pointer    ;HL -> Eintrag

080C  C1        POP   BC                  ;C - Anzahl
080D  CD 034F   CALL  move_de_hl          ;C Bytes von DE nach
                                          ;HL kopieren
                                          ;und Directory Record
                                          ;schreiben
;
;
; UPRO für 'Close'
;
; Directory Record auf Disk schreiben
;
write_dir_record:
;
0810  CD 03C3   CALL  set_dir_trk_sec     ;Track und Sektor an-
                                          ;wählen
0813  C3 05C6   JP    dir_write           ;und Record schreiben
;
;
; Fortsetzung von 'Rename File'
;
_rename_file:
;
0816  CD 0554   CALL  disk_ro?            ;Disk schreibge-
                                          ;schützt ?
0819  0E 0C     LD    C,12                ;Nur Usernummer, File-
                                          ;namen und Filetype
                                          ;vergleichen
081B  CD 0718   CALL  search_dir_first    ;Ersten Eintrag mit
                                          ;altem Filenamen
                                          ;suchen
```

```
081E  2A 0343  LD    HL,(Sentry_word)  ;HL -> FCB
0821  7E       LD    A,(HL)            ;A = Usernummer des
                                       ;alten Filenamens
0822  11 0010  LD    DE,pos_dx         ;DE = Position des
                                       ;neuen Namens
0825  19       ADD   HL,DE             ;HL -> Usernummer des
                                       ;neuen Filenamens
0826  77       LD    (HL),A            ;Die alte Usernummer
                                       ;wird übernommen
;
rename_loop:
;
                                       ;Eintrag mit altem Na-
                                       ;men gefunden ?
0827  CD 05F5  CALL  all_entries_compared?
082A  C8       RET   Z                 ;N: Fertig

082B  CD 0544  CALL  entry_ro?         ;J: Eintrag schreib-
                                       ;geschützt ?
082E  0E 10    LD    C,16              ;N: C = Position des
                                       ;neuen Filenamen und
                                       ;Filetyps im FCB
0830  1E 0C    LD    E,12              ;Nur Usernummer, File-
                                       ;namen und Filetype
                                       ;kopieren und
                                       ;Eintrag auf Disk
                                       ;schreiben
0832  CD 0801  CALL  write_partial_entry

0835  CD 072D  CALL  search_dir_next    ;Nächsten passenden
                                       ;Eintrag suchen
0838  C3 0827  JP    rename_loop       ;und umbennen
;
;
; Fortsetzung von 'Set File Attr'
;
_set_file_attr:
;
083B  0E 0C    LD    C,12              ;Nur die ersten 12
                                       ;Zeichen (Usernummer,
                                       ;Filename und Filetyp)
                                       ;vergleichen
083D  CD 0718  CALL  search_dir_first  ;Ersten passenden Ein-
                                       ;trag suchen
;
set_attr_loop:
;
                                       ;Ersten/nächsten Ein-
                                       ;trag gefunden ?
0840  CD 05F5  CALL  all_entries_compared?
0843  C8       RET   Z                 ;N: Fertig
```

```
0844  0E 00     LD    C,0                 ;J: Vom FCB ab Posi-
                                          ;tion 0
0846  1E 0C     LD    E,12                ;12 Bytes in den Ein-
                                          ;trag übernehmen und
                                          ;den Eintrag wieder
                                          ;zurückschreiben
0848  CD 0801   CALL  write_partial_entry

084B  CD 072D   CALL  search_dir_next     ;Nächsten passenden
                                          ;Eintrag suchen
084E  C3 0840   JP    set_attr_loop       ;und neu setzen
;
;
; Fortsetzung von 'Open File'
;
_open_file:
;
0851  0E 0F     LD    C,15                ;Die ersten 15 Bytes
                                          ;(also auch die Ex-
                                          ;tendnummer) verglei-
                                          ;chen
0853  CD 0718   CALL  search_dir_first    ;Ersten passenden Ein-
                                          ;trag suchen
                                          ;Eintrag gefunden ?
0856  CD 05F5   CALL  all_entries_compared?
0859  C8       RET   Z                   ;N: Fertig
;
;
; Gefundenen Eintrag in den FCB übernehmen.
;
; Der gefundene Eintrag deckt entweder den gewünschten
; Extend ab, oder der Extend liegt 'hinter' dem File-
; ende.
;
open_entry:
;
085A  CD 04A6   CALL  get_extend_pointer;Adresse der Extend-
                                        ;nummer des FCBs holen
085D  7E        LD    A,(HL)            ;A = Gewünschte Ex-
                                        ;tendnummer
085E  F5        PUSH  AF                ;Extendnummer retten
085F  E5        PUSH  HL                ;Zeiger retten
0860  CD 055E   CALL  get_entry_pointer ;Zeiger auf den Ein-
                                        ;trag holen
```

```
0863  EB        EX    DE,HL              ;DE -> Gefundener Ein-
                                         ;trag
0864  2A 0343   LD    HL,(Sentry_word)   ;HL -> FCB
0867  0E 20     LD    C,32               ;32 Bytes
0869  D5        PUSH  DE                 ;Zeiger auf Eintrag
                                         ;retten
086A  CD 034F   CALL  move_de_hl         ;Kompletten Eintrag
                                         ;(32 Bytes) in den
                                         ;FCB kopieren
086D  CD 0578   CALL  not_altered        ;'File unverändert'
                                         ;markieren
0870  D1        POP   DE                 ;Zeiger auf Eintrag
                                         ;zurück
0871  21 000C   LD    HL,pos_ex
0874  19        ADD   HL,DE              ;HL -> Extendnummer
                                         ;des Eintrags
0875  4E        LD    C,(HL)             ;C - Nummer des letz-
                                         ;ten Extends im gefun-
                                         ;denen Eintrag
0876  21 000F   LD    HL,pos_rc
0879  19        ADD   HL,DE              ;HL -> Anzahl der Re
                                         ;cords in diesem letz-
                                         ;ten Extend
087A  46        LD    B,(HL)             ;B - Anzahl der Re-
                                         ;cords im letzten
                                         ;Extend
087B  E1        POP   HL                 ;Zeiger auf die FCB-
                                         ;Extendnummer zurück
087C  F1        POP   AF                 ;Gewünschte Extend-
                                         ;nummer zurück
087D  77        LD    (HL),A             ;Die gewünschte Ex-
                                         ;tendnummer wieder
                                         ;in den FCB setzen
087E  79        LD    A,C                ;A = Höchste Extend-
                                         ;nummer des Eintrags
087F  BE        CP    (HL)               ;Ist die gewünschte
                                         ;Extendnummer, die
                                         ;Nummer des letzten
                                         ;Extends ?
0880  78        LD    A,B                ;A = Anzahl der Re-
                                         ;cords im letzten
                                         ;Extend
0881  CA 088B   JP    Z,set_record_count ;J: Die Anzahl der Re-
                                         ;cords im letzten Ex-
                                         ;tend, ist die Anzahl
                                         ;der Records im ge-
                                         ;wünschten Extend
```

```
0884  3E 00     LD    A,0                 ;N: A = 0
                                          ;Liegt der gewünschte
                                          ;Extend über dem letz-
                                          ;ten Extend ?
0886  DA 088B   JP    C,set_record_count  ;J: Der gewünschte Ex-
                                          ;tend liegt 'hinter'
                                          ;dem Ende des Files
                                          ;Dieser Extend enthält
                                          ;also noch keine
                                          ;(Null !) Records
0889  3E 80     LD    A,128               ;N: Der gewünschte Ex-
                                          ;tend liegt 'vor' dem
                                          ;letzten Extend des
                                          ;Eintrags und wird
                                          ;vom Eintrag komplett
                                          ;abgedeckt.
                                          ;Dieser Extend ist
                                          ;also mit 128 Records
                                          ;gefüllt.
;
;
; Anzahl der Records im geöffneten Extend, in den FCB
; eintragen.
;
set_record_count:
;
088B  2A 0343   LD    HL,(§entry_word)    ;HL -> FCB
088E  11 000F   LD    DE,pos_rc
0891  19        ADD   HL,DE               ;HL -> Recordnummer
0892  77        LD    (HL),A              ;Recordnummer ein-
                                          ;setzen
0893  C9        RET
;
;
; UPRO für 'Close File'
;
; 16-Bit Blocknummer übernehmen
;
close_word_block:
;
0894  7E        LD    A,(HL)              ;Ist der Blocktabel-
0895  23        INC   HL                  ; leneintrag frei ?
0896  B6        OR    (HL)
0897  2B        DEC   HL
0898  C0        RET   NZ                  ;N: Fertig
```

```
0899  1A        LD    A,(DE)              ;J: LSB
089A  77        LD    (HL),A              ; übernehmen
089B  13        INC   DE
089C  23        INC   HL
089D  1A        LD    A,(DE)              ;MSB übernehmen
089E  77        LD    (HL),A
089F  1B        DEC   DE
08A0  2B        DEC   HL
08A1  C9        RET
;
;
; Fortsetzung von 'Close File'
;
_close_file:
;
08A2  AF        XOR   A                   ;Rückgabeparameter auf
08A3  32 0345   LD    (§exit_value),A     ;Null setzen
08A6  32 0DEA   LD    (§dir_entry_#),A    ;Eintragsnummer auf
08A9  32 0DEB   LD    (§dir_entry_#+1),A;Null setzen

08AC  CD 051E   CALL  get_ro_bit          ;R/O-Bit der Disk
                                          ;testen.
                                          ;Disk schreibge-
                                          ;schützt ?
08AF  C0        RET   NZ                  ;J: Fertig, keinen
                                          ;Fehler melden

08B0  CD 0569   CALL  get_eg              ;'NA'-Flag holen
08B3  E6 80     AND   080H                ;Wurde das File ge-
                                          ;ändert ?
08B5  C0        RET   NZ                  ;N: Fertig, Ein un-
                                          ;veränderter FCB wird
                                          ;nicht in die Directo-
                                          ;ry zurückgeschrieben

08B6  0E 0F     LD    C,15                ;J: Ersten passenden
08B8  CD 0718   CALL  search_dir_first    ;Eintrag suchen
                                          ;Eintrag gefunden ?
08BB  CD 05F5   CALL  all_entries_compared?
08BE  C8        RET   Z                   ;N: Fertig

08BF  01 0010   LD    BC,pos_dx           ;BC = Position der
                                          ;Blocktabelle
08C2  CD 055E   CALL  get_entry_pointer   ;Adresse des gefunde-
                                          ;nen Eintrags holen
08C5  09        ADD   HL,BC
08C6  EB        EX    DE,HL               ;DE -> Blocktabelle
                                          ;des Eintrags
```

```
08C7  2A 0343  LD    HL,(§entry_word)  ;HL -> FCB
08CA  09       ADD   HL,BC             ;HL -> Blocktabelle
                                       ;des FCBs
08CB  0E 10    LD    C,16              ;Alle 16 Einträge der
                                       ;Blocktabelle über-
                                       ;prüfen
;
close_block_loop:
;
08CD  3A 0DDD  LD    A,(§large_dsm?)   ;Sind die Blocknummern
08D0  B7       OR    A                 ;16-Bit Werte ?
08D1  CA 08E8  JP    Z,word_block_close;J: 16-Bit Blocknum-
                                       ;mern übernehmen
;
;
; Die Blocknummern sind 8-Bit Werte
; DE -> Blocktabelle des Eintrags
; HL -> Blocktabelle des FCB
;
byte_block_close:
;
08D4  7E       LD    A,(HL)            ;A = Blocknummer aus
                                       ;dem FCB
08D5  B7       OR    A                 ;Ist der Blocktabel-
                                       ;leneintrag belegt ?
08D6  1A       LD    A,(DE)            ;A = Blocknummer aus
                                       ;dem Eintrag
                                       ;J: Blocknummern tes-
                                       ;ten
08D7  C2 08DB  JP    NZ,close_byte_block

08DA  77       LD    (HL),A            ;N: Blocknummern
                                       ;gleichsetzen
;
close_byte_block:
;
08DB  B7       OR    A                 ;Blocknummer aus dem
                                       ;Eintrag = 0 ?
                                       ;N: Blocknummern ver-
                                       ;gleichen
08DC  C2 08E1  JP    NZ,same_byte_block?
08DF  7E       LD    A,(HL)            ;J: Blocknummer vom
                                       ;FCB
08E0  12       LD    (DE),A            ;in den Eintrag über-
                                       ;nehmen
```

```
;
same_byte_block?:
;
08E1  BE        CP    (HL)                ;Sind die beiden
                                          ;Blocktabelleneinträge identisch ?
08E2  C2 091F   JP    NZ,cant_close       ;N: Fehler
08E5  C3 08FD   JP    close_next_block    ;J: Nächsten Blocktabelleneintrag überprüfen
;
;
; Die Blocknummern sind 16-Bit Werte
;
; DE -> Blocktabelle des Eintrags
; HL -> Blocktabelle des FCB
;
word_block_close:
;
08E8  CD 0894   CALL  close_word_block    ;Blocknummer vom Eintrag in den FCB übernehmen
08EB  EB        EX    DE,HL               ;Blocknummer vom FCB
08EC  CD 0894   CALL  close_word_block    ;in den Eintrag übernehmen
08EF  EB        EX    DE,HL               ;
;
08F0  1A        LD    A,(DE)              ;A = LSB der Blocknummer des Eintrags
08F1  BE        CP    (HL)                ;gleich dem LSB der Blocknummer des FCB ?
08F2  C2 091F   JP    NZ,cant_close       ;N: Fehler
;
08F5  13        INC   DE                  ;J: Beide Zeiger
08F6  23        INC   HL                  ;erhöhen
08F7  1A        LD    A,(DE)              ;A = MSB der Blocknummer des Eintrags
08F8  BE        CP    (HL)                ;gleich dem MSB der Blocknummer des FCB ?
08F9  C2 091F   JP    NZ,cant_close       ;N: Fehler
;
08FC  0D        DEC   C                   ;J: Blockzähler -1
;
; Nächsten Blocktabelleneintrag überprüfen
;
close_next_block:
;
08FD  13        INC   DE                  ;Beide Zeiger
08FE  23        INC   HL                  ;erhöhen
08FF  0D        DEC   C                   ;Blockzähler -1
                                          ;Weiter, bis Blockzähler = 0
0900  C2 08CD   JP    NZ,close_block_loop
```

```
0903  01 FFEC  LD    BC,-20              ;Die 16 Positionen
                                         ;Blocktabelle und
                                         ;noch 4 weitere
0906  09       ADD   HL,BC               ;Positionen abziehen

0907  EB       EX    DE,HL               ;DE -> Extendnummer
                                         ;des FCB
0908  09       ADD   HL,BC               ;HL -> Extendnummer
                                         ;des Eintrags
0909  1A       LD    A,(DE)              ;Ist die Extendnummer
090A  BE       CP    (HL)                ;des FCB kleiner als
                                         ;die Extendnummer des
                                         ;Eintrags ?
                                         ;J: Das File wurde
                                         ;verkürzt. Neuen Entry
                                         ;auf Disk schreiben
090B  DA 0917  JP    C,write_closed_entry

090E  77       LD    (HL),A              ;N: Neue Extendnummer
                                         ;in den Eintrag über-
                                         ;nehmen
090F  01 0003  LD    BC,3
0912  09       ADD   HL,BC
0913  EB       EX    DE,HL               ;DE -> Höchste Record-
                                         ;nummer des letzten
                                         ;Extend im Eintrag
0914  09       ADD   HL,BC               ;HL -> Aktuelle Re-
                                         ;cordnummer im FCB
0915  7E       LD    A,(HL)              ;Recordnummer des FCB
0916  12       LD    (DE),A              ;als Recordnummer des
                                         ;Eintrags übernehmen
                                         ;Dadurch wird das File
                                         ;an der 'aktuellen
                                         ;Position' geschlossen
;
;
; Geschlossenen Eintrag auf Disk schreiben
;
write_closed_entry:
;
0917  3E FF    LD    A,0FFH              ;Flag für 'open_next
0919  32 0DD2  LD    (Sentry_closed?),A  ;extend' setzen

091C  C3 0810  JP    write_dir_record    ;Directory-Record
                                         ;schreiben
```

```
;
; Fehlerausgang, falls die Blocktabelleneinträge nicht mehr
; stimmen
;
cant_close:
;
091F  21 0345  LD    HL,§exit_value     ;HL -> Rückgabepara-
                                        ;meter (= 0 !)
0922  35       DEC   (HL)               ;Parameter = 0FFH
                                        ;setzen
0923  C9       RET
;
;
; Fortsetzung von 'Make File'
; UPRO für 'Write' bei Fileerweiterung
;
; Neuen Directory-Eintrag erzeugen
;
_make_file:
;
0924  CD 0554  CALL  disk_ro?           ;Disk schreibge-
                                        ;schützt ?
0927  2A 0343  LD    HL,(§entry_word)   ;HL -> FCB
092A  E5       PUSH  HL                 ;FCB-Zeiger retten

092B  21 0DAC  LD    HL,§make_fcb       ;HL -> 0E5H
092E  22 0343  LD    (§entry_word),HL   ;Adresse für 'search_
                                        ;dir_first' setzen
0931  0E 01    LD    C,1                ;Nur 1 Byte verglei-
                                        ;chen
0933  CD 0718  CALL  search_dir_first   ;Also einen Eintrag
                                        ;dessen erstes Byte
                                        ;gleich 0E5H ist
                                        ;(Dies ist die Ken-
                                        ;nung für einen frei-
                                        ;en Eintrag !) suchen
                                        ;Wurde ein freier Ein-
                                        ;trag gefunden ?
0936  CD 05F5  CALL  all_entries_compared?

0939  E1       POP   HL                 ;FCB-Zeiger zurück
093A  22 0343  LD    (§entry_word),HL   ;und wieder einsetzen
093D  C8       RET   Z                  ;N: Fertig
```

```
093E  EB       EX    DE,HL              ;DE -> FCB
093F  21 000F  LD    HL,pos_rc
0942  19       ADD   HL,DE              ;HL -> Recordnummer
                                        ;des FCB
0943  0E 11    LD    C,17               ;Die nächsten 17 Bytes
                                        ;(also die Recordnum-
                                        ;mer und die Blockta-
                                        ;belle)
0945  AF       XOR   A                  ;auf 0 setzen
;
make_clear_loop:
;
0946  77       LD    (HL),A             ;0 einsetzen
0947  23       INC   HL                 ;Zeiger +1
0948  0D       DEC   C                  ;Zähler -1
0949  C2 0946  JP    NZ,make_clear_loop;Weiter, bis Zähler =0
;
; 'S1'-Position des neuen Eintrags auf Null setzen
; und den Eintrag in die Directory zurückschreiben.
;
094C  21 000D  LD    HL,pos_s1
094F  19       ADD   HL,DE              ;HL -> S1
0950  77       LD    (HL),A             ;S1 auf Null setzen
                                        ;Nummer des höchsten
                                        ;belegten Eintrags
                                        ;aktualisieren
0951  CD 058C  CALL  new_end_of_entries?
                                        ;FCB komplett in den
                                        ;Eintrag übernehmen,
                                        ;Eintrag auf Disk
                                        ;schreiben
0954  CD 07FD  CALL  write_complete_entry
0957  C3 0578  JP    not_altered        ;und File auf 'unver-
                                        ;ändert' setzen
;
;
; UPRO für 'Read' und 'Write'
;
; Nächsten Extend des Files eröffnen
;
open_next_extend:
;
095A  AF       XOR   A                  ;Ersteinmal annehmen,
095B  32 0DD2  LD    (§entry_closed?),A;daß kein neuer Ein-
                                        ;trag eröffnet werden
                                        ;muß.
095E  CD 08A2  CALL  _close_file        ;Aktuellen Extend
                                        ;schliessen
```

```
                                        ;Wurde der zu schlies-
                                        ;sende Eintrag gefun-
                                        ;den ?
0961  CD 05F5  CALL  all_entries_compared?
0964  C8       RET   Z                  ;N: Fehler

0965  2A 0343  LD    HL,(§entry_word)
0968  01 000C  LD    BC,pos_ex
096B  09       ADD   HL,BC              ;HL -> Alte (gerade
                                        ;geschlossene) Extend-
                                        ;nummer des FCB
096C  7E       LD    A,(HL)             ;A - Extendnummer
096D  3C       INC   A                  ;A - Nächste (zu
                                        ;öffnende) Extend-
                                        ;nummer
096E  E6 1F    AND   01FH               ;A - Neue Extendnummer
                                        ;modulo 32
0970  77       LD    (HL),A             ;Neue Extendnummer
                                        ;in den FCB eintragen.
                                        ;Ist die neue Extend-
                                        ;nummer ein Viel-
                                        ;faches von 32 ?
0971  CA 0983  JP    Z,next_ext_group   ;J: Nächste Extend-
                                        ;gruppe eröffnen

0974  47       LD    B,A                ;N: B - neue Extend-
                                        ;nummer
0975  3A 0DC5  LD    A,(§exm)           ;A = Extend Mask
0978  A0       AND   B                  ;Neue Extendnummer
                                        ;maskieren
                                        ;A = 0, falls der
                                        ;neue Extend in einem
                                        ;neuen Eintrag liegt
0979  21 0DD2  LD    HL,§entry_closed?  ;HL -> Flag
097C  A6       AND   (HL)               ;A <> 0, falls der
                                        ;neue Extend in dem
                                        ;Eintrag liegt, der
                                        ;gerade geschlossen
                                        ;wurde. Ist dies der
                                        ;Fall ?
097D  CA 098E  JP    Z,search_extend    ;N: Entsprechenden
                                        ;Eintrag suchen
0980  C3 09AC  JP    entry_found        ;J: Ok, Eintrag ge-
                                        ;funden
```

```
;
; Die neuen Extendnummer liegt in einer neuen Extendgruppe.
; Neue Extendgruppe eröffnen
;
next_ext_group:
;
0983  01 0002  LD    BC,diff_ex_eg
0986  09       ADD   HL,BC                ;HL -> Extendgruppe
0987  34       INC   (HL)                 ;Extendgruppen-
                                          ;Nummer +1
0988  7E       LD    A,(HL)               ;A = Neue Extendgruppe
0989  E6 0F    AND   0FH                  ;Extendgruppe > 15 ?
098B  CA 09B6  JP    Z,cant_extend_file;J: Maximale File-
                                          ;größe überschrit-
                                          ;ten
;
; Eintrag des neuen Extends suchen
;
search_extend:
;
098E  0E 0F    LD    C,15                 ;Extendnummer mit ver-
                                          ;gleichen
0990  CD 0718  CALL  search_dir_first     ;Eintrag suchen
                                          ;Wurde der Eintrag ge-
                                          ;funden ?
0993  CD 05F5  CALL  all_entries_compared?
0996  C2 09AC  JP    NZ,entry_found       ;J: ok
;
; Für den neuen Extend konnte kein Eintrag gefunden werden
;
0999  3A 0DD3  LD    A,(&read_or_write?)
099C  3C       INC   A                    ;Kam der Aufruf von
                                          ;'Read' ?
099D  CA 09B6  JP    Z,cant_extend_file;J: Fehler, Ende des
                                          ;Files erreicht

09A0  CD 0924  CALL  _make_file           ;N: Eintrag für den
                                          ;neuen Extend anle-
                                          ;gen.
                                          ;Eintrag erfolgreich
                                          ;engelegt ?
09A3  CD 05F5  CALL  all_entries_compared?

09A6  CA 09B6  JP    Z,cant_extend_file;N: Fehler melden
09A9  C3 09AF  JP    new_extend_opened ;J: Neuer Extend ist
                                          ;eröffnet
```

```
;
; Es wurde ein passender Eintrag gefunden
;
entry_found:
;
09AC  CD 085A  CALL  open_entry          ;Eintrag mit ent-
                                         ;sprechender Extend-
                                         ;nummer eröffnen
;
; Es wurde ein neuer Extend geöffnet
;
new_extend_opened:
;
09AF  CD 04BB  CALL  get_record_#s       ;Parameter übernehmen
09B2  AF       XOR   A                   ;Keinen Fehler
09B3  C3 0301  JP    return_byte         ; zurückmelden
;
;
; Es konnte kein nächster Extend eröffnet werden
;
cant_extend_file:
;
09B6  CD 0305  CALL  read_write_error    ;Fehler melden und
09B9  C3 0578  JP    not_altered         ;'File unverändert'
                                         ;setzen
;
;
; Fortsetzung von 'Read Sequential'
;
_read_sequ:
;
09BC  3E 01    LD    A,1                 ;Sequential-Flag
09BE  32 0DD5  LD    (§sequential?),A    ;auf 1 setzen
                                         ;(Die Recordnummer
                                         ;wird nach dem Lese-
                                         ;zugriff um 1 erhöht)
;
;
;
; Einsprung für 'Read Random'
;
read_rnd_seq:
;
09C1  3E FF    LD    A,0FFH              ;Flag auf 'READ'
                                         ;setzen
09C3  32 0DD3  LD    (§read_or_write?),A
09C6  CD 04BB  CALL  get_record_#s       ;Parameter aus dem
                                         ;FCB übernehmen
```

```
09C9  3A 0DE3  LD    A,(§sequ_rec)       ;A = Gewünschte Re-
                                         ;cordnummer
09CC  21 0DE1  LD    HL,§max_ext_rec     ;HL -> Höchste Record-
                                         ;nummer des letzten
                                         ;Extends im FCB
09CF  BE       CP    (HL)                ;Wird die gewünschte
                                         ;Recordnummer noch vom
                                         ;FCB abgedeckt ?
09D0  DA 09E6  JP    C,read_sequ_record  ;J: Record lesen

09D3  FE 80    CP    128                 ;Folgen noch mehr
                                         ;Extends ? (Ist die
                                         ;höchste Recordnummer
                                         ;des letzten Extends
                                         ;gleich 128 ?)
09D5  C2 09FB  JP    NZ,end_of_file      ;N: Fehler, Ende des
                                         ;Files erreicht
09D8  CD 095A  CALL  open_next_extend    ;J: Nächsten Extend
                                         ;öffnen
09DB  AF       XOR   A                   ;Und Record Nummer 0
09DC  32 0DE3  LD    (§sequ_rec),A       ;des neuen Extends
                                         ;lesen

09DF  3A 0345  LD    A,(§exit_value)     ;Wurde überhaupt ein
09E2  B7       OR    A                   ;nächster Extend ge
                                         ;funden ?
09E3  C2 09FB  JP    NZ,end_of_file      ;N: Fehler, Ende des
                                         ;Files erreicht
;
;
; Record lesen
;
read_sequ_record:
;
09E6  CD 0477  CALL  set_§block_#        ;Blocknummer des Re-
                                         ;cords aus der Block-
                                         ;tabelle holen.
09E9  CD 0484  CALL  check_§block_#      ;Ist der entsprechende
                                         ;Eintrag in der Block-
                                         ;tabelle belegt ?
09EC  CA 09FB  JP    Z,end_of_file       ;N: Fehler, Ende des
                                         ;Files erreicht

09EF  CD 048A  CALL  set_§record#        ;J: Absolute Record-
                                         ;nummer berechnen
09F2  CD 03D1  CALL  set_dsk_trk_sec     ;Entsprechende Spur
                                         ;und Sektor anwählen,
09F5  CD 03B2  CALL  read_record         ;Record lesen
09F8  C3 04D2  JP    update_record_#s    ;und Parameter im FCB
                                         ;aktualisieren
```

```
;
; Ende des Files erreicht
;
end_of_file:
;
09FB  C3 0305  JP    read_write_error  ;Fehler zurückmelden
;
;
; Fortsetzung von 'Write Sequential'
;
_write_sequ:
;
09FE  3E 01    LD    A,1               ;Sequential-Flag
0A00  32 0DD5  LD    (§sequential?),A  ;auf 1 setzen
                                       ;(Die Recordnummer
                                       ;wird nach dem Lese-
                                       ;zugriff um 1 erhöht)
;
;
; Einsprung für 'Write Random'
;
write_rnd_seq:
;
0A03  3E 00    LD    A,0               ;Flag auf 'Write'
                                       ;setzen
0A05  32 0DD3  LD    (§read_or_write?),A

0A08  CD 0554  CALL  disk_ro?          ;Disk schreibge-
                                       ;schützt ?
0A0B  2A 0343  LD    HL,(§entry_word)  ;N: HL -> FCB
0A0E  CD 0547  CALL  fcb_ro?           ;File schreibge-
                                       ;schützt ?

0A11  CD 04BB  CALL  get_record_#s     ;N: Parameter aus dem
                                       ;FCB übernehmen
0A14  3A 0DE3  LD    A,(§sequ_rec)     ;Ist die gewünschte
0A17  FE 80    CP    128               ;Recordnummer kleiner
                                       ;als 128 ?
                                       ;N: Fehler, Falsche
                                       ;Recordnummer ange-
                                       ;wählt
0A19  D2 0305  JP    NC,read_write_error

0A1C  CD 0477  CALL  set_§block_#      ;J: Entsprechende
                                       ;Blocknummer der
                                       ;Blocktabelle ent-
                                       ;nehmen
0A1F  CD 0484  CALL  check_§block_#    ;Ist der Block schon
                                       ;belegt ?
```

```
0A22  0E 00     LD    C,0               ;C - BIOS-Flag für
                                        ;normalen Schreib-
                                        ;zugriff
0A24  C2 0A6E   JP    NZ,normal_write   ;J: Normaler Schreib-
                                        ;zugriff
0A27  CD 043E   CALL  get_block_pos     ;N: Position der neuen
                                        ;Blocknummer in der
                                        ;Blocktabelle berech-
0A2A  32 0DD7   LD    (&block_pos),A    ;nen und merken

0A2D  01 0000   LD    BC,0              ;BC - zuletzt belegte
                                        ;Blocknummer, falls
                                        ;ein neuer Eintrag an-
                                        ;gelegt werden muß
0A30  B7        OR    A                 ;Ist die Tabellenposi-
                                        ;tion der neuen Block-
                                        ;nummer gleich Null ?
0A31  CA 0A3B   JP    Z,get_new_block   ;J: Die aktuelle
                                        ;Blocktabelle ist be-
                                        ;legt, es muß ein
                                        ;neuer Eintrag ge-
                                        ;schaffen werden
0A34  4F        LD    C,A               ;N: BC - Neue Block-
0A35  0B        DEC   BC                ;position - 1
0A36  CD 045E   CALL  get_block_#       ;Letzte belegte Block-
                                        ;nummer der Blockta-
                                        ;belle entnehmen
0A39  44        LD    B,H               ;BC = Nummer des zu-
0A3A  4D        LD    C,L               ;letzt belegten Blocks
;
;
; Nächsten freien Block belegen
; BC = zuletzt belegte Blocknummer
;
get_new_block:
;
                                        ;Nächstgelegenen,
                                        ;freien Block suchen
0A3B  CD 07BE   CALL  allocate_next_block
0A3E  7D        LD    A,L               ;Freien Block gefun-
0A3F  B4        OR    H                 ;den ?
0A40  C2 0A48   JP    NZ,write_new_block;J: Weiter
;
;
; Es konnte kein freier Block mehr gefunden werden.
; 'Disk voll' melden
;
disk_full:
;
0A43  3E 02     LD    A,2               ;A = Fehlercode für
                                        ;'Disk voll'
0A45  C3 0301   JP    return_byte       ;Fehlercode zurück-
                                        ;geben
```

```
;
; Record in den neu belegten Block schreiben
;
write_new_block:
;
0A48   22 0DE5   LD     (§block_#),HL      ;Blocknummer merken
0A4B   EB        EX     DE,HL              ;DE - Blocknummer
0A4C   2A 0343   LD     HL,(§entry_word)   ;HL -> FCB
0A4F   01 0010   LD     BC,pos_dx
0A52   09        ADD    HL,BC              ;HL -> Blocktabelle

0A53   3A 0DDD   LD     A,(§large_dsm?)    ;Sind die Blocknummern
0A56   B7        OR     A                  ;16-Bit Werte ?
0A57   3A 0DD7   LD     A,(§block_pos)     ;A = Position des neu-
                                           ;en Blocks in der
                                           ;Blocktabelle
                                           ;J: Blocknummer als
                                           ;16-Bit Wert eintragen
0A5A   CA 0A64   JP     Z,new_block_word
                                           ;N: Blocknummer als
                                           ;8-Bit Wert eintragen
0A5D   CD 0564   CALL   add_hl_a           ;HL -> Blocktabelle +
                                           ;Position
0A60   73        LD     (HL),E             ;Blocknummer eintragen
0A61   C3 0A6C   JP     first_write        ;und Record als ersten
                                           ;Record des neuen
                                           ;Blocks schreiben
;
;
; Neue Blocknummer als 16-Bit Wert eintragen
;
new_block_word:
;
0A64   4F        LD     C,A                ;BC = Position des
0A65   06 00     LD     B,0                ;neuen Blocks
0A67   09        ADD    HL,BC              ;zweimal aufaddieren
0A68   09        ADD    HL,BC              ; da jeder Block zwei
                                           ; Bytes in der Block-
                                           ; tabelle belegt
0A69   73        LD     (HL),E             ;LSB der Blocknummer
0A6A   23        INC    HL                 ; eintragen
0A6B   72        LD     (HL),D             ;MSB der Blocknummer
                                           ; eintragen
```

```
;
; Der zu schreibende Record, ist der erste Record eines neuen
; Blocks
;
first_write:
;
0A6C  0E 02     LD    C,2                ;C = BIOS-Flag für
                                         ;Zugriff auf neuen
                                         ;Block
;
;
; Record schreiben
;
; C = Record-Flag
;   = 2, falls der Record in einem neuen Block liegt
;   = 0 sonst
;
normal_write:
;
0A6E  3A 0345   LD    A,(§exit_value)    ;A = vorheriger Feh-
                                         ;lercode
0A71  B7        OR    A                  ;War vorher ein Feh-
                                         ;ler ?
0A72  C0        RET   NZ                 ;J: Schreibzugriff
                                         ;abbrechen

0A73  C5        PUSH  BC                 ;N: Record-Flag retten
0A74  CD 048A   CALL  set_§record#       ;Absolute Recordnummer
                                         ;berechnen. HL = Re-
                                         ;cordnummer
0A77  3A 0DD5   LD    A,(§sequential?)   ;A = Zugriffsart
0A7A  3D        DEC   A                  ;Ist die Zugriffsart
0A7B  3D        DEC   A                  ;gleich 2 ?
                                         ;N: Daten schreiben
0A7C  C2 0ABB   JP    NZ,write_actual_data
;
;
; Der Block soll vorher mit 00H gefüllt werden
; ('Write Random with Zero Fill')
;
0A7F  C1        POP   BC                 ;Record-Flag zurück
0A80  C5        PUSH  BC                 ;und wieder retten
0A81  79        LD    A,C                ;A = Record-Flag
0A82  3D        DEC   A                  ;Gilt der Schreib-
0A83  3D        DEC   A                  ;zugriff dem ersten
                                         ;Record eines neuen
                                         ;Blocks ?
                                         ;(Record-Flag = 2 ?)
                                         ;N: Der Block ist be-
                                         ;reits initialisiert
0A84  C2 0ABB   JP    NZ,write_actual_data
```

```
                                                    ;J: Kompletten Block
                                                    ;schreiben
0A87  E5        PUSH  HL                            ;Recordnummer retten
0A88  2A 0DB9   LD    HL,(?§dirbuf)                 ;HL -> DIRBUF
0A8B  57        LD    D,A                           ;D = 0 (Zähler)
;
;
; DIRBUF mit 00H füllen
;
zero_write_buffer:
;
0A8C  77        LD    (HL),A                        ;00H einsetzen
0A8D  23        INC   HL                            ;Bufferzeiger +1
0A8E  14        INC   D                             ;Zähler +1
                                                    ;Zähler < 128 ?
                                                    ;Weiter, bis
                                                    ;Zähler = 128
0A8F  F2 0A8C   JP    P,zero_write_buffer
;
;
; Den auf 0 initialisierten DIRBUF in den neuen Block
; schreiben
;
0A92  CD 05E0   CALL  set_dma_to_dirbuf             ;DMA-Adresse = DIRBUF
                                                    ;setzen
0A95  2A 0DE7   LD    HL,(§blkrecord)               ;HL = Absolute
                                                    ;Recordnummer des
                                                    ;ersten Records im
                                                    ;neuen Block
0A98  0E 02     LD    C,2                           ;C = BIOS-Flag für
                                                    ;'Erster Schreibzu-
                                                    ;griff auf einen
                                                    ;neuen Block'
;
block_write_loop:
;
0A9A  22 0DE5   LD    (§abs_rec_#),HL               ;Recordnummer spei-
                                                    ;chern
0A9D  C5        PUSH  BC                            ;BIOS-Flag retten
0A9E  CD 03D1   CALL  set_dsk_trk_sec               ;Track und Sektor an-
                                                    ;wählen
0AA1  C1        POP   BC                            ;BIOS-Flag zurück

0AA2  CD 03B8   CALL  write_record                  ;(Null-)Record schrei-
                                                    ;ben

0AA5  2A 0DE5   LD    HL,(§abs_rec_#)               ;HL = Nummer des ge-
                                                    ;schriebenen Records
0AA8  0E 00     LD    C,0                           ;C = BIOS-Flag für den
                                                    ;nächsten Schreibzu-
                                                    ;griff
```

```
0AAA  3A 0DC4  LD    A,(§blm)           ;A = Anzahl der Re-
                                        ;cords pro Block -1
0AAD  47       LD    B,A                ;B = A

0AAE  A5       AND   L                  ;A = Geschriebene
                                        ;Recordnummer modulo
                                        ;Anzahl der Records
                                        ;pro Block
0AAF  B8       CP    B                  ;Alle Records des
                                        ;neuen Blockes ge-
                                        ;schrieben ?
0AB0  23       INC   HL                 ;Recordnummer +1
                                        ;N: Nächsten (Null-)
                                        ;Record schreiben
0AB1  C2 0A9A  JP    NZ,block_write_loop

0AB4  E1       POP   HL                 ;J: Recordnummer
                                        ;zurück
0AB5  22 0DE5  LD    (§abs_rec_#),HL    ;Recordnummer spei-
                                        ;chern
0AB8  CD 05DA  CALL  restore_dma        ;DMA-Adresse auf alten
                                        ;Wert setzen
;
; Tatsächliche Record-Daten schreiben
;
write_actual_data:
;
0ABB  CD 03D1  CALL  set_dsk_trk_sec    ;Track und Sektor an-
                                        ;wählen
0ABE  C1       POP   BC                 ;Record-Flag zurück
0ABF  C5       PUSH  BC                 ;und wieder retten
0AC0  CD 03B8  CALL  write_record       ;Record schreiben
0AC3  C1       POP   BC                 ;Record-Flag zurück

0AC4  3A 0DE3  LD    A,(§sequ_rec)      ;A = Nummer des ge-
                                        ;schriebenen Records
                                        ;innerhalb des aktu-
                                        ;ellen Extends
0AC7  21 0DE1  LD    HL,§max_ext_rec    ;HL -> maximale Re-
                                        ;cordnummer des aktu-
                                        ;ellen Extends
0ACA  BE       CP    (HL)               ;Liegt der geschrie-
                                        ;bene Record noch im
                                        ;aktuellen Extend ?
                                        ;J: Die Filelänge wur-
                                        ;de nicht verändert
0ACB  DA 0AD2  JP    C,still_in_current_ext
```

```
0ACE  77        LD    (HL),A          ;N: Neue Recordnummer
                                      ;im FCB merken
0ACF  34        INC   (HL)            ;und Filelänge um
                                      ;einen Record er-
                                      ;weitern
0AD0  0E 02     LD    C,2             ;C = Record-Flag für
                                      ;den folgenden Test
;
still_in_current_ext:
;
                                      ;In neueren BDOS-Ver-
                                      ;sionen, stehen an den
                                      ;nächsten 5 Bytes
                                      ;'NOP' (!)

(0AD2  0D       DEC   C               )  ;Wurde ein neuer Block
(0AD3  0D       DEC   C               )  ;begonnen ? (C - 2 ?)
(0AD4  C2 0ADF  JP    NZ,test_rec_#)   ;N: weiter
                                       ;J: 'File verändert'
                                       ;setzen
;
0AD7  F5        PUSH  AF              ;Recordnummer retten
0AD8  CD 0569   CALL  get_eg          ;Adresse des 'NA'-Flag
                                      ;holen
0ADB  E6 7F     AND   07FH            ;'File geändert' set-
                                      ;zen
0ADD  77        LD    (HL),A          ;Flag zurückschreiben
0ADE  F1        POP   AF              ;Recordnummer zurück
;
test_rec_#:
;
0ADF  FE 7F     CP    127             ;War der geschriebene
                                      ;Record, der letzte Re-
                                      ;cord des aktuellen
                                      ;Extends ?
                                      ;N: Write beenden
0AE1  C2 0B00   JP    NZ,jp_update_record_#s

0AE4  3A 0DD5   LD    A,(§sequential?) ;War der Schreibzu-
0AE7  FE 01     CP    1               ;griff sequentiell ?
                                      ;N: Write beenden
0AE9  C2 0B00   JP    NZ,jp_update_record_#s
```

```
;
; Nächsten Extend für den nächsten sequentiellen Schreib-
; zugriff eröffnen
;
0AEC  CD 04D2  CALL  update_record_#s   ;Neue Recordnummer im
                                        ;FCB speicnern
0AEF  CD 095A  CALL  open_next_extend   ;Nächsten Extend er-
                                        ;öffnen

0AF2  21 0345  LD    HL,§exit_value     ;HL -> Fehlercode vom
                                        ;'open'-Aufruf
0AF5  7E       LD    A,(HL)             ;A = Fehlercode
0AF6  B7       OR    A                  ;Konnte ein nächster
                                        ;Extend eröffnet wer-
                                        ;den ?
                                        ;N: Fertig, der Fehler
                                        ;wird erst beim näch-
                                        ;sten 'Write'-Aufruf
                                        ;gemeldet
0AF7  C2 0AFE  JP    NZ,clear_exit_value

0AFA  3D       DEC   A                  ;J: Recordnummer auf
0AFB  32 0DE3  LD    (§sequ_rec),A      ;-1 setzen, da der
                                        ;'update_record_#s'-
                                        ;Aufruf die Record-
                                        ;nummer um 1 erhöht
                                        ;und somit die Record-
                                        ;nummer auf 0 setzt
;
; Fehlercode löschen
;
clear_exit_value:
;
0AFE  36 00    LD    (HL),0             ;Fehlercode auf 'kein
                                        ;Fehler' setzen
;
; Neue Recordposition wieder in den FCB einsetzten.
; Nach eine sequentiellen Zugriff, wird eine um 1 erhöhte
; Recordposition eingesetzt.
;
jp_update_record_#s:
;
0B00  C3 04D2  JP    update_record_#s
```

```
;
; UPRO für 'Read Random', 'Write Random'
;
; Random Record Nummer (RRN) in die entsprechende
; sequentielle Record Nummer (SRN) umrechnen.
;
; I: §entry_word -> FCB mit gesetzter Random Record Nummer
;    C = 00H, falls der Aufruf von 'Write Random' kam
;    C = FFH, falls der Aufruf von 'Read Random' kam
;
; O: §entry_word -> FCB mit Sequentieller Record Nummer
;    §exit_value - Fehlercode
;
trans_rnd_to_seq:
;
0B03  AF        XOR   A                    ;Zugriffsflag auf
0B04  32 0DD5   LD    (§sequential?),A     ;'Random' setzen
;
;
; Einsprung für 'Write Random With Zero Fill'
;
zero_trans_r_to_s:
;
0B07  C5        PUSH  BC                   ;Schreib/Lesekennung
                                           ;retten
0B08  2A 0343   LD    HL,(§entry_word)
0B0B  EB        EX    DE,HL                ;DE -> FCB
0B0C  21 0021   LD    HL,pos_r0
0B0F  19        ADD   HL,DE                ;HL -> LSB der RRN

0B10  7E        LD    A,(HL)               ;A = LSB der RRN
0B11  E6 7F     AND   07FH                 ;RRN modulo 128 be-
                                           ;rechen. Dies ist die
                                           ;Recordnummer inner-
                                           ;halb des berechneten
                                           ;Extends
0B13  F5        PUSH  AF                   ;Recordnummer retten

0B14  7E        LD    A,(HL)               ;A = LSB der RRN
0B15  17        RLA                        ;Höchstes Bit ins
                                           ;CY-Flag schieben
0B16  23        INC   HL                   ;HL -> MSB der RRN
0B17  7E        LD    A,(HL)               ;A = MSB der RRN
0B18  17        RLA                        ;Höchstes Bit des
                                           ;LSBs nach Bit 0
                                           ;schieben. Dies er-
                                           ;gibt RRN / 128
0B19  E6 1F     AND   01FH                 ;(RRN / 128) mod 32
                                           ;ist die Extendnummer
0B1B  4F        LD    C,A                  ;C = Extendnummer
```

```
0B1C  7E        LD    A,(HL)            ;A - MSB der RRN
0B1D  1F        RRA                     ; /2
0B1E  1F        RRA                     ; /4
0B1F  1F        RRA                     ; /8
0B20  1F        RRA                     ; /16 (=RRN / 4096)
0B21  E6 0F     AND   00FH              ;Oberste 4 Bits lö-
                                        ;schen, da rotiert
                                        ;wurde
0B23  47        LD    B,A               ;B - RRN / 4096
                                        ;  - RRN / (32 * 128)
                                        ;  - Extendgruppen-
                                        ;    Nummer der SRN
0B24  F1        POP   AF                ;Recordnummer im Ex-
                                        ;tend zurück

0B25  23        INC   HL                ;HL -> Überlauf
                                        ; der RRN
0B26  6E        LD    L,(HL)            ;Ist das Überlauf-
0B27  2C        INC   L                 ; byte ungleich
0B28  2D        DEC   L                 ; Null ?
0B29  2E 06     LD    L,6               ;L - Fehlercode 'Seek
                                        ;past physical end of
                                        ;Disk'
0B2B  C2 0B8B   JP    NZ,rnd_seek_error ;J: Fehler melden
;
;
; Errechnete SRN im FCB abspeichern
; DE -> FCB
;
0B2E  21 0020   LD    HL,pos_cr
0B31  19        ADD   HL,DE             ;HL -> Recordnummer
                                        ;im FCB
0B32  77        LD    (HL),A            ;Recordnummer im FCB
                                        ;speichern
0B33  21 000C   LD    HL,pos_ex
0B36  19        ADD   HL,DE             ;HL -> Extendnummer
                                        ;im FCB
0B37  79        LD    A,C               ;A = Errechnete Ex-
                                        ;tendnummer
0B38  96        SUB   (HL)              ;Ist der errechnete
                                        ;Extend schon ge-
                                        ;öffnet ?
                                        ;N: Extend neu öffnen
0B39  C2 0B47   JP    NZ,different_extend
```

```
0B3C  21 000E  LD    HL,pos_eg
0B3F  19       ADD   HL,DE              ;HL -> Extendgruppe
0B40  78       LD    A,B                ;A - Errechnete Ex-
                                        ;tendgruppen-Nummer
0B41  96       SUB   (HL)               ;Ist diese Extend-
                                        ;gruppe schon
                                        ;geöffnet ?
0B42  E6 7F    AND   07FH               ;Bit 7 (FCB-Flag ver-
                                        ;nachlässigen)
                                        ;J: Der gewünschte Re-
                                        ;cord liegt im aktuell
                                        ;geöffneten Extend
0B44  CA 0B7F  JP    Z,correct_rnd_access
;
;
; Die gewünschte Recordnummer liegt in einem anderen Extend.
; Dieser Extend muß noch geöffnet werden.
;
different_extend:
;
0B47  C5       PUSH  BC                 ;Extendnummer und Ex-
                                        ;tendgruppennummer
                                        ;retten
0B48  D5       PUSH  DE                 ;FCB-Zeiger retten
0B49  CD 08A2  CALL  _close_file        ;Aktuellen Extend
                                        ;schliessen
0B4C  D1       POP   DE                 ;FCB-Zeiger zurück
0B4D  C1       POP   BC                 ;Extendnummer und Ex-
                                        ;tendgruppen-Nummer
                                        ;zurück
0B4E  2E 03    LD    L,3                ;L = Fehlercode
                                        ;'Cannot Close Current
                                        ;Extend'
0B50  3A 0345  LD    A,(&exit_value)    ;Ist ein Fehler beim
0B53  3C       INC   A                  ;Close-Aufruf aufge-
                                        ;treten ?
0B54  CA 0B84  JP    Z,bad_rnd_access   ;J: Fehler melden

0B57  21 000C  LD    HL,pos_ex
0B5A  19       ADD   HL,DE              ;HL -> Extendnummer
                                        ;im FCB
0B5B  71       LD    (HL),C             ;Neue Extendnummer
                                        ;einsetzen
0B5C  21 000E  LD    HL,pos_eg
0B5F  19       ADD   HL,DE              ;HL -> Extendgruppen-
                                        ;Nummer im FCB
0B60  70       LD    (HL),B             ;Neue Extendgruppen-
                                        ;Nummer einsetzen
```

```
0B61  CD 0851  CALL  _open_file          ;Neuen Extend öffnen
0B64  3A 0345  LD    A,(§exit_value)     ;Konnte der neue Ex-
0B67  3C       INC   A                   ;tend geöffnet wer-
                                         ;den ?
                                         ;J: 'Kein Fehler'
                                         ;melden
0B68  C2 0B7F  JP    NZ,correct_rnd_access
;
;
; Der neue Extend konnte nicht geöffnet werden
;
0B6B  C1       POP   BC                  ;Schreib/Lesekennung
                                         ;zurück
0B6C  C5       PUSH  BC                  ;und wieder retten
0B6D  2E 04    LD    L,4                 ;L = Fehlercode 'Seek
                                         ;To Unwritten Extend'
0B6F  0C       INC   C                   ;Handelt es sich um
                                         ;eine Lesezugriff ?
0B70  CA 0B84  JP    Z,bad_rnd_access    ;J: Fehler melden
;
;
; Neuen Eintrag zur Erweiterung des Files erzeugen
; (Nur bei einem Schreibzugriff)
;
0B73  CD 0924  CALL  _make_file          ;N: Neuen Eintrag er-
                                         ;öffnen, der neue Ex-
                                         ;tend wird vom alten
                                         ;Eintrag nicht mehr
                                         ;abgedeckt
0B76  2E 05    LD    L,5                 ;L = Fehlercode 'Cant
                                         ;Create New Extend'
0B78  3A 0345  LD    A,(§exit_value)     ;Konnte ein neuer Ein-
0B7B  3C       INC   A                   ;trag geschaffen wer-
                                         ;den ?
0B7C  CA 0B84  JP    Z,bad_rnd_access    ;N: Fehler melden
;
;
; Der gewünschte Record bzw. Extend wurde gefunden
; und ist geöffnet
;
correct_rnd_access:
;
0B7F  C1       POP   BC                  ;Schreib/Lesekennung
                                         ;zurück
0B80  AF       XOR   A                   ;Fehlercode 0
0B81  C3 0301  JP    return_byte         ;zurückmelden
```

```
;
;
; Der letzte Extend konnte nicht geschlossen oder
; der neue Extend nicht eröffnet werden
;
bad_rnd_access:
;
0B84  E5        PUSH  HL                ;Fehlercode (L) retten
0B85  CD 0569   CALL  get_eg            ;Adresse des FCB-Flags
                                        ;holen
0B88  36 C0     LD    (HL),0C0H         ;Obere zwei Bit setzen
                                        ;(File als 'unverän-
                                        ;dert' markieren und
                                        ;als Fehlerkennung die
                                        ;Extendgruppennummer
                                        ;auf 64 setzen)
0B8A  E1        POP   HL                ;Fehlercode (L) zurück
;
;
; Fehlerhafte Random Record Nummer angegeben
;
rnd_seek_error:
;
0B8B  C1        POP   BC                ;Schreib/Lesekennung
                                        ;zurück
0B8C  7D        LD    A,L               ;A = Fehlercode
0B8D  32 0345   LD    (exit_value),A    ;Fehlercode zurück-
                                        ;melden
0B90  C3 0578   JP    not_altered       ;und 'File unver-
                                        ;ändert' setzen
;
;
; Fortsetzung von 'Read Random'
;
_read_random:
;
0B93  0E FF     LD    C,0FFH            ;C = Lesekennung
0B95  CD 0B03   CALL  trans_rnd_to_seq  ;RRN in SRN umrechnen
                                        ;Fehler ?
0B98  CC 09C1   CALL  Z,read_rnd_seq    ;N: Record lesen
0B9B  C9        RET
;
;
; Fortsetzung von 'Write Random'
;
_write_random:
;
0B9C  0E 00     LD    C,0               ;C = Schreibkennung
0B9E  CD 0B03   CALL  trans_rnd_to_seq  ;RRN in SRN umrechnen
                                        ;Fehler ?
0BA1  CC 0A03   CALL  Z,write_rnd_seq   ;N: Record schreiben
0BA4  C9        RET
```

```
;
; UPRO für 'Set Random Record' und 'Compute File Size'
;
; Sequentielle Record Nummer (SRN) in die entsprechende
; Random Record Nummer (RRN) umrechnen und in den FCB ein-
; setzen.
;
; I: DE  = Position der umzurechnenden Recordnummer im FCB
;        = Position der aktuellen Recordnummer (pos_cr)
;          (für 'Set Random Record')
;        = Position der höchsten Recordnummer (pos_rc)
;          (für 'Compute File Size')
;    HL -> FCB
;
; O: BC  - Errechnete Random Record Nummer
;        = 32 * 128 * Extendgruppe + 128 * Extendnummer
;          + Recordnummer
;    A   = Überlaufbyte der RRN
;        = 0, falls die RRN kleiner als 65536 ist
;        = 1, sonst
;
trans_seq_to_rnd:
;
0BA5  EB        EX    DE,HL
0BA6  19        ADD   HL,DE             ;HL -> aktuelle oder
                                        ;höchste Recordnummer
0BA7  4E        LD    C,(HL)
0BA8  06 00     LD    B,0               ;BC - Recordnummer

0BAA  21 000C   LD    HL,pos_ex
0BAD  19        ADD   HL,DE             ;HL -> Extendnummer
0BAE  7E        LD    A,(HL)            ;A = Extendnummer
0BAF  0F        RRCA                    ;Bit 0 nach Bit 7 ro-
                                        ;tieren
0BB0  E6 80     AND   080H              ;und restliche Bits
                                        ;löschen ergibt:
                                        ;Extendnummer * 128
0BB2  81        ADD   A,C               ;BC = BC + A
0BB3  4F        LD    C,A               ;Niedrigstes Bit der
0BB4  3E 00     LD    A,0               ;Extendnummer zur
0BB6  88        ADC   A,B               ;bisherigen RRN
0BB7  47        LD    B,A               ;hinzuaddieren
0BB8  7E        LD    A,(HL)            ;A = Extendnummer
0BB9  0F        RRCA                    ; / 2
0BBA  E6 0F     AND   00FH              ;Restliche Bits lö-
                                        ;schen, da rotiert
                                        ;wurde
0BBC  80        ADD   A,B               ;Restliche Bits der
0BBD  47        LD    B,A               ;Extendnummer zur
                                        ;RRN addieren
                                        ;BC = Recordnummer
                                        ;+ 128 * Extendnummer
```

```
0BBE  21 000E  LD    HL,pos_eg
0BC1  19       ADD   HL,DE          ;HL -> Extendgruppe
0BC2  7E       LD    A,(HL)         ;Nummer der Extend-
                                    ;gruppe holen

0BC3  87       ADD   A,A            ;A = Extendgruppe *  2
0BC4  87       ADD   A,A            ;                 *  4
0BC5  87       ADD   A,A            ;                 *  8
0BC6  87       ADD   A,A            ;                 * 16
0BC7  F5       PUSH  AF             ;Eventuelle Überlauf-
                                    ;kennung im CY-Bit
                                    ;retten
0BC8  80       ADD   A,B            ;Extendgruppe * 16 zum
0BC9  47       LD    B,A            ;MSB der bisher er-
                                    ;rechneten Recordnum-
                                    ;mer addieren.
                                    ;Dies entspricht der
                                    ;Addition von 16*256*
                                    ;Extendgruppe und er-
                                    ;gibt in BC die tat-
                                    ;sächliche RRN

0BCA  F5       PUSH  AF             ;CY-Flag von der letz-
                                    ;ten Addition über den
0BCB  E1       POP   HL             ;Stack nach L kopieren
0BCC  7D       LD    A,L            ;A = CY-Flag
0BCD  E1       POP   HL             ;CY-Flag von der S2-
                                    ;Multiplikation zurück
0BCE  B5       OR    L              ;Trat bei einer der
                                    ;letzten beiden arith-
                                    ;metischen Operationen
                                    ;ein Überlauf auf ?
0BCF  E6 01    AND   01H            ;J: A = 1
                                    ;N: A = 0
0BD1  C9       RET
;
;
; Fortsetzung von 'Compute File Size'
;
; Random Record Nummer des letzten (!) Records im File
; berechnen
;
_compute_file_size:
;
0BD2  0E 0C    LD    C,12           ;Nur Usernummer, File-
                                    ;namen und Filetype
                                    ;vergleichen
0BD4  CD 0718  CALL  search_dir_first ;Ersten passenden Ein-
                                    ;trag suchen
```

```
0BD7  2A 0343  LD    HL,(§entry_word)  ;HL -> FCB
0BDA  11 0021  LD    DE,pos_r0         ;D = 0 (!)
0BDD  19       ADD   HL,DE             ;HL -> LSB der Random
                                       ;Record Nummer des FCB

0BDE  E5       PUSH  HL                ;RRN-Zeiger retten
0BDF  72       LD    (HL),D            ;RRN im FCB
0BE0  23       INC   HL                ;mit
0BE1  72       LD    (HL),D            ;Null
0BE2  23       INC   HL                ;vorbe-
0BE3  72       LD    (HL),D            ;setzen
;
; Alle zum FCB passenden Einträge durchsuchen und die
; höchste gefundene sequentielle Record Nummer als
; Random Record Nummer in den FCB einsetzen
;
compute_size_loop:
;
                                       ;Noch einen weiteren
                                       ;Eintrag gefunden ?
0BE4  CD 05F5  CALL  all_entries_compared?
0BE7  CA 0C0C  JP    Z,pop_hl          ;N: RRN-Zeiger vom
                                       ;Stack nehmen, Fertig

0BEA  CD 055E  CALL  get_entry_pointer ;J: Zeiger auf den ge-
                                       ;fundenen Eintrag
                                       ;holen
0BED  11 000F  LD    DE,pos_rc         ;DE = Position der
                                       ;höchsten Recordnummer
                                       ;im gefundenen Eintrag
0BF0  CD 0BA5  CALL  trans_seq_to_rnd  ;Dazugehörige RRN be-
                                       ;rechnen
0BF3  E1       POP   HL                ;RRN-Zeiger zurück
0BF4  E5       PUSH  HL                ;und wieder retten
0BF5  5F       LD    E,A               ;EBC = Errechnete RRN

                                       ;Ist die errechnete
                                       ;RRN größer als die
                                       ;bisher im FCB ein-
                                       ;getragene RRN ?
0BF6  79       LD    A,C               ;LSB der RRN ver-
0BF7  96       SUB   (HL)              ;gleichen
0BF8  23       INC   HL
0BF9  78       LD    A,B               ;MSB der RRN ver-
0BFA  9E       SBC   A,(HL)            ;gleichen
0BFB  23       INC   HL
0BFC  7B       LD    A,E               ;Überlaufbyte der
0BFD  9E       SBC   A,(HL)            ;RRN vergleichen
                                       ;Größere RRN gefunden?
0BFE  DA 0C06  JP    C,not_greater     ;N: Nächsten Eintrag
                                       ;überprüfen
```

```
0C01  73        LD    (HL),E              ;J: Errechnete RRN
0C02  2B        DEC   HL                  ; als neue
0C03  70        LD    (HL),B              ; größte RRN
0C04  2B        DEC   HL                  ; im FCB
0C05  71        LD    (HL),C              ; speichern
;
;
; Nächsten Eintrag suchen
;
not_greater:
;
0C06  CD 072D   CALL  search_dir_next     ;Nächsten passenden
                                          ;Eintrag suchen
0C09  C3 0BE4   JP    compute_size_loop   ;und dessen höchste
                                          ;Recordnummer über-
                                          ;prüfen
;
;
; Ende von _compute_file_size
;
pop_hl:
;
0C0C  E1        POP   HL                  ;RRN-Zeiger vom Stack
                                          ;nehmen
0C0D  C9        RET                       ;Fertig
;
;
; **** BDOS-Funktion Nummer 36  'Set Random Record'
;
?set_random_record:
;
0C0E  2A 0343   LD    HL,(&entry_word)    ;HL -> FCB
0C11  11 0020   LD    DE,pos_cr           ;DE = Position der
                                          ;aktuellen Record-
                                          ;nummer im FCB
0C14  CD 0BA5   CALL  trans_seq_to_rnd    ;Random Record Nummer
                                          ;des aktuellen Records
                                          ;berechnen
0C17  21 0021   LD    HL,pos_r0
0C1A  19        ADD   HL,DE               ;HL -> RRN des FCB
0C1B  71        LD    (HL),C              ;LSB der RRN
0C1C  23        INC   HL                  ;speichern
0C1D  70        LD    (HL),B              ;MSB der RRN
0C1E  23        INC   HL                  ;speichern
0C1F  77        LD    (HL),A              ;Überlaufbyte
                                          ;speichern
0C20  C9        RET
;
```

```
;
; UPRO für 'Select Disk'
; Neue Disk anwählen
;
; I: §bdos_disk - Nummer der anzuwählenden Disk
;
; O: -
;
login_disk:
;
0C21  2A 0DAF  LD    HL,(§login_vec)   ;HL = Login-Vektor
0C24  3A 0342  LD    A,(§bdos_disk)
0C27  4F       LD    C,A               ;C = Disknummer
0C28  CD 04EA  CALL  shift_right_hl    ;Entsprechendes Bit
                                       ;des Login-Vektors
                                       ;nach L,0 schieben.
0C2B  E5       PUSH  HL                ;Login-Bit retten
0C2C  EB       EX    DE,HL             ;E,0 = Login-Bit
0C2D  CD 0359  CALL  physical_select   ;Disk selektieren
                                       ;und DPH/DPB über-
                                       ;nehmen
0C30  E1       POP   HL                ;Login-Bit zurück
                                       ;Fehler beim physical_
                                       ;select ?
0C31  CC 0347  CALL  Z,select_error    ;J: Fehler melden

0C34  7D       LD    A,L               ;A,0 = Login-Bit
0C35  1F       RRA                     ;Login-Bit ins CY-Flag
                                       ;schieben.
                                       ;War die Disk bereits
                                       ;vorher selektiert ?
0C36  D8       RET   C                 ;J: Fertig

                                       ;N: ALV und CSV neu
                                       ;setzen
0C37  2A 0DAF  LD    HL,(§login_vec)
0C3A  4D       LD    C,L               ;BC = Alter Login-
0C3B  44       LD    B,H               ;Vektor
0C3C  CD 050B  CALL  set_disk_vec      ;Bit der neuen Disk im
                                       ;Login-Vektor setzen
0C3F  22 0DAF  LD    (§login_vec),HL   ;und neuen Login-
                                       ;Vektor speichern
0C42  C3 06A3  JP    logical_select    ;CSV und ALV setzen
```

```
;
; **** BDOS-Funktion Nummer 14  'Select Disk'
;
; I: §entry-byte = Disknummer
;
?select_disk:
;
0C45  3A 0DD6  LD    A,(§entry_byte)   ;A = Disknummer
0C48  21 0342  LD    HL,§bdos_disk     ;HL -> aktuelle Disk-
                                       ;nummer
0C4B  BE       CP    (HL)              ;Soll die aktuelle
                                       ;Disk angewählt wer-
                                       ;den ?
0C4C  C8       RET   Z                 ;J: Fertig

0C4D  77       LD    (HL),A            ;N: Neue Disknummer
                                       ;als aktuelle Disk-
                                       ;nummer speichern
0C4E  C3 0C21  JP    login_disk        ;und neue Disk an-
                                       ;wählen
;
;
; UPRO für alle Filefunktionen
;
; Die im FCB angegebene Disk selektieren und die aktuelle
; Usernummer in den FCB eintragen
;
; I: §entry_word -> Diskcode (im FCB)
;
; O: §entry_word -> Aktuelle Usernummer (im FCB)
;
auto_select:
;
0C51  3E FF    LD    A,0FFH            ;Flag für ?bdos_exit
0C53  32 0DDE  LD    (§file_func?),A   ;auf 'File-Funktion'
                                       ;setzen

0C56  2A 0343  LD    HL,(§entry_word)  ;HL -> Diskcode
0C59  7E       LD    A,(HL)            ;A = Diskcode
0C5A  E6 1F    AND   01FH              ;Obere 3 Bits löschen
0C5C  3D       DEC   A                 ;und Diskcode in Disk-
                                       ;nummer umrechnen
0C5D  32 0DD6  LD    (§entry_byte),A   ;Disknummer für even-
                                       ;tuellen 'Select'-Auf-
                                       ;ruf speichern
0C60  FE 1E    CP    01EH              ;War der Diskcode
                                       ;= '?' oder 0 ?
0C62  D2 0C75  JP    NC,no_auto_slct   ;J: Keine Disk extra
                                       ;anwählen
```

```
0C65   3A 0342   LD     A,(§bdos_disk)     ;Aktuelle Disknummer
0C68   32 0DDF   LD     (§old_bdos_disk),A;für '?bdos_exit'
                                           ;speichern

0C6B   7E        LD     A,(HL)             ;Neue Disknummer
0C6C   32 0DE0   LD     (§old_fcb_disk),A  ;für '?bdos_exit'
                                           ;speichern
0C6F   E6 E0     AND    0E0H               ;Disknummer auf 0
                                           ;oder, falls der
                                           ;Diskcode '?' war,
                                           ;auf 20H setzen
0C71   77        LD     (HL),A             ;Disknummer in FCB
                                           ;einsetzen
0C72   CD 0C45   CALL   ?select_disk       ;Neue Disk anwählen
;
no_auto_slct:
;
0C75   3A 0341   LD     A,(§user_#)        ;A - Usernummer
0C78   2A 0343   LD     HL,(§entry_word)   ;HL -> FCB
0C7B   B6        OR     (HL)               ;Usernummer in den
0C7C   77        LD     (HL),A             ;FCB einsetzen
0C7D   C9        RET
;
;
; **** BDOS-Funktion Nummer 12  'Return Version Number'
;
?return_version_#:
;
0C7E   3E 22     LD     A,version_#        ;Versionsnummer
0C80   C3 0301   JP     return_byte        ;zurückgeben
;
;
; **** BDOS-Funktion Nummer 13  'Reset Disk System'
;
?reset_disk_system:
;
0C83   21 0000   LD     HL,0               ;HL = 0
0C86   22 0DAD   LD     (§ro_vec),HL       ;R/O-Vektor und
0C89   22 0DAF   LD     (§login_vec),HL    ;Login-Vektor löschen
0C8C   AF        XOR    A                  ;Aktuelle Disknummer
0C8D   32 0342   LD     (§bdos_disk),A     ;auf 0 setzen
0C90   21 0080   LD     HL,§buffer         ;DMA-Adresse auf
0C93   22 0DB1   LD     (§dma_address),HL  ;Standardwert setzen
0C96   CD 05DA   CALL   restore_dma        ;Und dies auch dem
                                           ;BIOS mitteilen
0C99   C3 0C21   JP     login_disk         ;Disk A: (für SUBMIT)
                                           ;anwählen
```

```
;
; **** BDOS-Funktion Nummer 15  'Open File'
;
?open_file:
;
0C9C  CD 0572  CALL  zero_eg          ;Extendgruppe auf
                                      ;Null setzen
0C9F  CD 0C51  CALL  auto_select      ;Disk selektieren
0CA2  C3 0851  JP    _open_file       ;und File öffnen
;
;
; **** BDOS-Funktion Nummer 16  'Close File'
;
?close_file:
;
0CA5  CD 0C51  CALL  auto_select      ;Disk selektieren
0CA8  C3 08A2  JP    _close_file      ;und File schliessen
;
;
; **** BDOS-Funktion Nummer 17  'Search for First'
;
?search_first:
;
0CAB  0E 00    LD    C,0              ;C = Anzahl der zu
                                      ;vergleichenden Bytes
0CAD  EB       EX    DE,HL            ;HL -> FCB
0CAE  7E       LD    A,(HL)           ;Diskcode holen
0CAF  FE 3F    CP    '?'              ;Diskcode - '?' ?
0CB1  CA 0CC2  JP    Z,search_now     ;J: Beliebigen Eintrag
                                      ;suchen; Es braucht
                                      ;kein Byte (C ist 0 !)
                                      ;zwischen FCB und Ein-
                                      ;trag übereinzustimmen

0CB4  CD 04A6  CALL  get_extend_pointer;N: Adresse der Ex-
                                      ;tendnummer holen
0CB7  7E       LD    A,(HL)           ;A = Gewünschte Ex-
                                      ;tendnummer
0CB8  FE 3F    CP    '?'              ;Extendnummer = '?' ?
0CBA  C4 0572  CALL  NZ,zero_eg       ;N: Extendgruppe auf
                                      ;Null setzen
0CBD  CD 0C51  CALL  auto_select      ;Disk anwählen
0CC0  0E 0F    LD    C,15             ;Alle Bytes Eintrags
                                      ;vergleichen
;
search_now:
;
0CC2  CD 0718  CALL  search_dir_first ;Ersten passenden Ein-
                                      ;trag suchen
0CC5  C3 05E9  JP    move_dirbuf_to_dma;und Directory-Record
                                      ;des gefundenen Ein-
                                      ;trags zur DMA-Adresse
                                      ;kopieren
```

```
;
; **** BDOS-Funktion Nummer 18  'Search for Next'
;
?search_next:
;
0CC8  2A 0DD9  LD    HL,(?§search_fcb) ;HL -> FCB vom letzten
                                       ;'Search for First'-
                                       ;Aufruf
0CCB  22 0343  LD    (§entry_word),HL  ;Es wird der nächste,
                                       ;zu diesem FCB passen-
                                       ;de Eintrag gesucht
0CCE  CD 0C51  CALL  auto_select       ;Disk anwählen

0CD1  CD 072D  CALL  search_dir_next   ;Nächsten passenden
                                       ;Eintrag suchen
0CD4  C3 05E9  JP    move_dirbuf_to_dma;und Directory-Record
                                       ;des gefundenen Ein-
                                       ;trags zur DMA-Adresse
                                       ;kopieren
;
;
; **** BDOS-Funktion Nummer 19  'Delete File'
;
?delete_file:
;
0CD7  CD 0C51  CALL  auto_select       ;Disk anwählen
0CDA  CD 079C  CALL  delete_files      ;Alle passenden
                                       ;Einträge löschen
0CDD  C3 0701  JP    return_found_flag ;und Fehlercode
                                       ;zurückgeben
;
;
; **** BDOS-Funktion Nummer 20  'Read Sequential'
;
?read_sequ:
;
0CE0  CD 0C51  CALL  auto_select       ;Disk anwählen
0CE3  C3 09BC  JP    _read_sequ        ;und nächsten sequen-
                                       ;tiellen Record lesen
;
;
; **** BDOS-Funktion Nummer 21  'Write Sequential'
;
?write_sequ:
;
0CE6  CD 0C51  CALL  auto_select       ;Disk anwählen
0CE9  C3 09FE  JP    _write_sequ       ;und nächsten sequen-
                                       ;tiellen Record
                                       ;schreiben
```

```
;
;
; **** BDOS-Funktion Nummer 22  'Make File'
;
?make_file:
;
0CEC  CD 0572  CALL  zero_eg           ;Extendgruppe auf
                                       ;Null setzen
0CEF  CD 0C51  CALL  auto_select       ;Disk anwählen
0CF2  C3 0924  JP    _make_file        ;und neuen Eintrag
                                       ;erzeugen
;
;
; **** BDOS-Funktion Nummer 23  'Rename File'
;
?rename_file:
;
0CF5  CD 0C51  CALL  auto_select       ;Disk anwählen
0CF8  CD 0816  CALL  _rename_file      ;Alle passenden Ein-
                                       ;träge umbennenen
0CFB  C3 0701  JP    return_found_flag ;und Fehlercode
                                       ;zurückgeben
;
;
; **** BDOS-Funktion Nummer 24  'Return Login Vector'
;
?return_login_vec:
;
0CFE  2A 0DAF  LD    HL,(§login_vec)   ;HL = Login-Vektor
0D01  C3 0D29  JP    return_word       ;16-Bit Wert zurück-
                                       ;geben
;
;
; **** BDOS-Funktion Nummer 25  'Return Current Disk'
;
?return_disk:
;
0D04  3A 0342  LD    A,(§bdos_disk)    ;A = Disknummer
0D07  C3 0301  JP    return_byte       ;8-Bit Wert zurück-
                                       ;geben
;
;
; **** BDOS-Funktion Nummer 26  'Set DMA Address'
;
?set_dma:
;
0D0A  EB       EX    DE,HL             ;HL = Neue DMA-Adresse
0D0B  22 0DB1  LD    (§dma_address),HL ;DMA-Adresse speichern
0D0E  C3 05DA  JP    restore_dma       ;und dem BIOS mit-
                                       ;teilen
```

```
;
; **** BDOS-Funktion Nummer 27   'Get Addr (Alloc)'
;
?get_allocation_addr:
;
0D11  2A 0DBF  LD    HL,(?§alv)          ;HL -> ALV
0D14  C3 0D29  JP    return_word         ;16-Bit Wert zurück-
                                         ;geben
;
;
; **** BDOS-Funktion Nummer 29   'Get R/O-Vector'
;
?get_ro_vec:
;
0D17  2A 0DAD  LD    HL,(§ro_vec)        ;HL - R/O-Vektor
0D1A  C3 0D29  JP    return_word         ;16-Bit Wert zurück-
                                         ;geben
;
;
; **** BDOS-Funktion Nummer 30   'Set File Attributes'
;
?set_file_attr:
;
0D1D  CD 0C51  CALL  auto_select         ;Disk anwählen
0D20  CD 083B  CALL  _set_file_attr      ;FCB in den Eintrag
                                         ;übernehmen
0D23  C3 0701  JP    return_found_flag   ;und Fehlercode
                                         ;zurückgeben
;
;
; **** BDOS-Funktion Nummer 31   'Get Addr (Disk Parms)'
;
?get_parameter_addr:
;
0D26  2A 0DBB  LD    HL,(?§dpb)          ;HL -> DPB
;
;
; UPRO für verschiedene BDOS-Funktionen
;
; 16-Bit Wert zurückgeben
;
return_word:
;
0D29  22 0345  LD    (§exit_value),HL    ;16-Bit Wert speichern
0D2C  C9       RET                       ;Rücksprung nach
                                         ;?bdos_exit
```

```
;
; **** BDOS Funktion Nummer 32  'Set/Get User Code'
;
?set_get_user_code:
;
0D2D  3A 0DD6  LD    A,(§entry_byte)   ;A = Neue Usernummer
0D30  FE FF    CP    0FFH              ;Usernummer holen ?
0D32  C2 0D3B  JP    NZ,set_user_code  ;N: Neue Usernummer
                                       ;speichern

0D35  3A 0341  LD    A,(§user_#)       ;J: Usernummer als
0D38  C3 0301  JP    return_byte       ;8-Bit Wert zurück-
                                       ;geben
;
;
; Neue Usernummer speichern
;
set_user_code:
;
0D3B  E6 1F    AND   01FH              ;Usernummer modulo 32
0D3D  32 0341  LD    (§user_#),A       ;speichern
0D40  C9       RET
;
;
; **** BDOS-Funktion Nummer 33  'Read Random'
;
?read_random:
;
0D41  CD 0C51  CALL  auto_select       ;Disk anwählen
0D44  C3 0B93  JP    _read_random      ;und Record lesen
;
;
; **** BDOS-Funktion Nummer 34  'Write Random'
;
?write_random:
;
0D47  CD 0C51  CALL  auto_select       ;Disk anwählen
0D4A  C3 0B9C  JP    _write_random     ;und Record schreiben
;
;
; **** BDOS-Funktion Nummer 35  'Compute File Size'
;
?compute_file_size:
;
0D4D  CD 0C51  CALL  auto_select       ;Disk anwählen
                                       ;und höchste Record-
                                       ;nummer bestimmen
0D50  C3 0BD2  JP    _compute_file_size
```

```
;
; **** BDOS-Funktion Nummer 37  'Reset Disk'
;
?reset_disk:
;
0D53  2A 0343  LD    HL,(§entry_word)  ;HL = Vektor; Die ge-
                                       ;setzten Bitposi-
                                       ;tionen werden im
                                       ;Lgoin- und R/O-
                                       ;Vektor gelöscht.
0D56  7D       LD    A,L
0D57  2F       CPL                     ;Bits negieren
0D58  5F       LD    E,A               ;E = LSB der Vektor-
                                       ;maske
0D59  7C       LD    A,H
0D5A  2F       CPL                     ;A = MSB der Vektor-
                                       ;maske

0D5B  2A 0DAF  LD    HL,(§login_vec)   ;HL = Login-Vektor
0D5E  A4       AND   H                 ;Entsprechende Bits
                                       ;löschen
0D5F  57       LD    D,A               ;und MSB in D merken
0D60  7D       LD    A,L               ;A = LSB des Login-
                                       ;Vektors
0D61  A3       AND   E                 ;Entsprechende Bits
                                       ;löschen
0D62  5F       LD    E,A               ;und LSB in E merken

0D63  2A 0DAD  LD    HL,(§ro_vec)      ;HL = R/O-Vektor
0D66  EB       EX    DE,HL             ;DE und HL tauschen
0D67  22 0DAF  LD    (§login_vec),HL   ;Neuen Login-Vektor
                                       ;speichern und weiter
                                       ;als Maske benutzen
                                       ;(Es können nur Disks
                                       ;'R/O' sein, die auch
                                       ;im Login-Vektor
                                       ;stehen)
0D6A  7D       LD    A,L               ;A = LSB der Maske
0D6B  A3       AND   E                 ;LSB des R/O-Vektors
                                       ;maskieren und
0D6C  6F       LD    L,A               ;in L speichern
0D6D  7C       LD    A,H               ;A = MSB der Maske
0D6E  A2       AND   D                 ;MSB des R/O-Vektors
                                       ;maskieren und
0D6F  67       LD    H,A               ;in H speichern

0D70  22 0DAD  LD    (§ro_vec),HL      ;Neuen R/O-Vektor
                                       ;speichern
0D73  C9       RET
```

```
;
; BDOS-Funktion abschliessen
;
; Alle BDOS-Funktionen durchlaufen diese Routine
;
?bdos_exit:
;
0D74  3A 0DDE  LD    A,(§file_func?)    ;Wurde eine File-Funk-
0D77  B7       OR    A                  ;tion ausgeführt ?
0D78  CA 0D91  JP    Z,ret_to_user      ;N: Fertig

0D7B  2A 0343  LD    HL,(§entry_word)   ;HL -> FCB
0D7E  36 00    LD    (HL),0             ;Usernummer im FCB
                                        ;löschen
0D80  3A 0DE0  LD    A,(§old_fcb_disk)  ;War der Diskcode im
0D83  B7       OR    A                  ;FCB gleich Null ?
0D84  CA 0D91  JP    Z,ret_to_user      ;J: Fertig, es wurde
                                        ;keine neue Disk
                                        ;selektiert

0D87  77       LD    (HL),A             ;N: Alten Diskcode
                                        ;wieder in den FCB
                                        ;einsetzen
0D88  3A 0DDF  LD    A,(§old_bdos_disk);A = Alte Disknummer
0D8B  32 0DD6  LD    (§entry_byte),A    ;Disknummer speichern
0D8E  CD 0C45  CALL  ?select_disk       ;und alte Disk wieder
                                        ;anwählen
;
ret_to_user:
;
0D91  2A 030F  LD    HL,(§user_stack)   ;HL = Stackpointer
0D94  F9       LD    SP,HL              ;Alten Stackpointer-
                                        ;wert wieder einsetzen
0D95  2A 0345  LD    HL,(§exit_value)   ;HL = Rückgabewert
0D98  7D       LD    A,L                ;A = L
0D99  44       LD    B,H                ;und B = H setzen
0D9A  C9       RET                      ;Fertig
;
;
; **** BDOS-Funktion Nummer 40  'Write Random with Zero Fill'
;
?write_zero_random:
;
0D9B  CD 0C51  CALL  auto_select        ;Disk anwählen
0D9E  3E 02    LD    A,2                ;Zugriffsflag ent-
0DA0  32 0DD5  LD    (§sequential?),A   ;sprechend setzen
0DA3  0E 00    LD    C,0                ;Schreibkennung setzen
0DA5  CD 0B07  CALL  zero_trans_r_to_s  ;RRN in SRN wandeln
                                        ;Fehler ?
0DA8  CC 0A03  CALL  Z,write_rnd_seq    ;N: Record schreiben
0DAB  C9       RET
```

```
;
; Zweiter Parameterbereich
;
;
; 1-Byte FCB für 'Make File'
;
0DAC  E5        §make_fcb:        defb    0e5h
;
;
; R/O-Vektor
;
0DAD  0000      §ro_vec:          defw    0
;
;
; Login-Vektor
;
0DAF  0000      §login_vec:       defw    0
;
;
; DMA-Adresse für Schreib/Leseoperationen
;
0DB1  0080      §dma_address:     defw    §buffer
;
;
; Zeiger auf die höchste belegte Eintragsnummer +1
; (Zeiger nach DPH+2)
;
0DB3  0000      ?§last_entry:     defw    0
;
;
; Zeiger auf die aktuelle (logische) Spurnummer
; (Zeiger nach DPH+4)
;
0DB5  0000      ?§track_#:        defw    0
;
;
; Zeiger auf die (logische) Recordnummer des ersten
; Records im aktuellen (logischen) Track
; (Zeiger nach DPH+6)
;
0DB7  0000      ?§trkrecord:      defw    0
;
;
; Zeiger auf die Adresse des aktuellen Directory-
; Buffers (DIRBUF)  (= Inhalt von DPH+8)
;
0DB9  0000      ?§dirbuf:         defw    0
```

```
;
; Zeiger auf die Adresse des aktuellen Disk-Parameter-
; Blocks (DPB)  (= Inhalt von DPH+10)
;
0DBB  0000      ?§dpb:          defw    0
;
;
; Zeiger auf die Adresse des aktuellen Prüfsummen-
; Vektors (CSV)  (= Inhalt von DPH+12)
;
0DBD  0000      ?§csv:          defw    0
;
;
; Zeiger auf die Adresse des aktuellen Belegungs-
; Vektors (ALV)  (= Inhalt von DPH+14)
;
0DBF  0000      ?§alv:          defw    0
;
;
; Aktueller Disk Parameter Block (DPB)
;
                §dpb:
0DC1  0000      §spt:           defw    0
0DC3  00        §bsh:           defb    0
0DC4  00        §blm:           defb    0
0DC5  00        §exm:           defb    0
0DC6  0000      §dsm:           defw    0
0DC8  0000      §drm:           defw    0
0DCA  0000      §al0_al1:       defw    0
0DCC  0000      §cks:           defw    0
0DCE  0000      §off:           defw    0
;
;
; Zeiger auf die Adresse der aktuellen Sektor-Verschränkungs-
; Tabelle (XLTAB)  (= Inhalt von DPH+0)
;
0DD0  0000      ?§xltab:        defw    0
;
;
; Flag für 'open_next_extend'
;
; = FFH, falls beim Schliessen des alten Extends, auch der
;        komplette Eintrag geschlossen wurde.
;
0DD2  00        §entry_closed?: defb    0
```

```
;
; Allgemeines Flag, ob zur Zeit eine Schreib- oder Lese-
; funktion bearbeitet wird
;
; = 00H, falls eine Schreibanforderung bearbeitet wird
; = FFH, falls eine Leseanforderung bearbeitet wird
;
0DD3  00        $read_or_write?: defb    0
;
;
; Fehlercode nach einer Directory-Suchfunktion
;
; = 00H, falls ein passender Eintrag gefunden wurde
; = FFH, falls kein passender Eintrag gefunden wurde
;
0DD4  00        $file_found?:    defb    0
;
;
; Zugriffskennung für die aktuelle Filefunktion
;
; = 0, bei direktem (Random) Zugriff
; = 1, bei sequentiellem Zugriff
; = 2, bei direktem Schreibzugriff mit Blockinitialisierung
;           (Write Random With Zero Fill)
;
0DD5  00        $sequential?:    defb    0
;
;
; 1-Byte Eingangsparameter
;
0DD6  00        $entry_byte:     defb    0
;
;
; Position der aktuelle Blocknummer in der Blocktabelle
;
0DD7  00        $block_pos:      defb    0
;
;
; Anzahl der zu vergleichenden Bytes bei den Directory-
; Suchfunktionen
;
0DD8  00        $compare_count:  defb    0
;
;
; Adresse des FCB, bei der Suche nach dem nächsten
; passenden Eintrag
;
0DD9  0000      ?$search_fcb:    defw    0
```

```
;
; Unbenutzt
;
0DDB  0000                    defw    0
;
;
; Flag, ob die Blocknummern als 8- oder 16-Bit Wert in der
; Blocktabelle abgelegt sind
;
; = FFH, für  8-Bit Blocknummern
; = 00H, für 16-Bit Blocknummern
;
0DDD  00      §large_dsm?:    defb    0
;
;
; Flag für '?bdos_exit', ob die abgeschlossene Funktion eine
; Filefunktion war
;
; = FFH, für alle Filefunktionen
; = 00H sonst
;
0DDE  00      §file_func?:    defb    0
;
;
; Alte Disknummer, falls für eine Filefunktion eine
; andere Disk selektiert wurde
;
0DDF  00      §old_bdos_disk: defb    0
;
;
; Im FCB übergebener Diskcode, der nach Beendigung der File-
; funktion wieder eingesetzt wird
;
0DE0  00      §old_fcb_disk:  defb    0
;
;
; Höchste Recordnummer innerhalb des aktuellen Extends
;
0DE1  00      §max_ext_rec:   defb    0
;
;
; Aktuelle Extendnummer
;
0DE2  00      §extend_#:      defb    0
;
;
; Aktuelle Recordnummer innerhalb des aktuellen Extends
;
0DE3  0000    §sequ_rec:      defw    0
```

```
;
; Blocknummer oder Absolute Recordnummer zur Anwahl eines
; neuen Track und Sektors
;
                §block_#:
0DE5  0000      §abs_rec_#:       defw    0
;
;
; Absolute Recordnummer des ersten Records im aktuellen Block
;
0DE7  0000      §blkrecord:       defw    0
;
;
; Position des aktuellen Eintrags innerhalb des DIRBUF
;
0DE9  00        §dir_pos:         defb    0
;
;
; Nummer des aktuellen Eintrags
;
0DEA  0000      §dir_entry_#:     defw    0
;
;
; Absolute Recordnummer des im DIRBUF stehenden Records
;
0DEC  0000      §dir_record_#:    defw    0
;
;
; Bytes zum Auffüllen an die nächste Seitengrenze.
; Damit braucht das BDOS genau 14 Seiten (0E00H Bytes)
;
0DEE  00                          defs    18,0
;
;
; Start des BIOS
;
bios:

                    END
```

Tips & Tricks

Das CP/M Betriebsystem ist in mancher Hinsicht nicht gerade als 'benutzerfreundlich' zu bezeichnen. Durch ein paar Änderungen im CCP und im BDOS, kann das Verhalten des CP/M jedoch in gewissen Grenzen verbessert werden.

Die nachfolgend beschriebenen Änderungen setzen Kenntnisse im Programmieren in Maschinensprache bzw. Assembler voraus und wenden sich vor allem an diejenigen, die ein eigenes BIOS schreiben wollen.

Da sowohl im CCP als auch im BDOS noch Speicherplätze unbenutzt sind (14 Bytes am Ende des CCP, 18 Bytes am Ende des BDOS), können Programmteile auch dort untergebracht werden. Neue BIOS-Routinen werden am besten über eine Erweiterung der BIOS-Sprungleiste erreicht.

Alle Adressangaben sind in der Form CCP+xxxxH bzw. BDOS+xxxxH gemacht und bezeichnen dadurch immer eine relative Position, ausgehend vom Anfang des CCP bzw. des BDOS.

Submit von einem anderen Laufwerk

Submit-Files werden normalerweise vom CCP nur auf dem Laufwerk A: erkannt.
Das Programm 'SUBMIT.COM' erzeugt das Submit-File '$$$.SUB' zwar auf dem jeweils aktuellen Laufwerk, der CCP wählt aber immer das Laufwerk A: explizit an. Auch erkennt der CCP einen SUBMIT-Lauf nur dann, wenn die 'Reset Disk System'-Funktion im BDOS auf dem Laufwerk A:, einen mit '$' beginnenden Filenamen findet.
Durch drei Änderungen im CCP, wird ein Submit-File immer auf dem jeweils aktuellen Laufwerk erkannt:

```
Änderung von:  CCP+013DH  CA 0196   JP    Z,no_submit
in:            CCP+013DH  00        NOP
               CCP+013EH  00        NOP
               CCP+013FH  00        NOP
```

damit wird immer versucht das File '$$$.SUB' zu öffnen.

```
Änderung von:  CCP+0146H  C4 00BD   CALL  NZ,select_disk
in:            CCP+0146H  00        NOP
               CCP+0147H  00        NOP
               CCP+0148H  00        NOP
```

damit wird die Anwahl von Laufwerk A: vor dem Öffnen des Files '$$$.SUB' verhindert.

```
Änderung von:  CCP+0181H  C4 00BD   CALL  NZ,select_disk
in:            CCP+0182H  00        NOP
               CCP+0183H  00        NOP
               CCP+0184H  00        NOP
```

damit wird die Anwahl von Laufwerk A: vor dem Löschen des Files '$$$.SUB' verhindert.

Schliessen eines Files

Der in der CP/M Dokumentation angegebene Algorithmus zur Behandlung von physikalischen Sektoren mit mehr als 128 Bytes (Blocking and Deblocking), arbeitet nicht immer korrekt. Besonders beim Schliessen eines Files kann es passieren, daß der geänderte Directory-Sektor nicht zurückgeschrieben wird. Zur Behebung dieses Fehlers ist eine Erweiterung der BIOS-Sprungleiste um die Funktion 'Flush' (Internen Sektorbuffer schreiben) nötig. Die 'Close File'-Funktion im BDOS ist zu diesem Zweck folgendermassen zu ändern:

```
Änderung von   BDOS+0CA8H  C3 08A2   JP    _close_file
in:            BDOS+0CA8H  C3 xxxx   JP    flush
```

Die Routine 'flush' muß als ersten Befehl einen 'CALL BDOS+08A2H' enthalten, der die 'Close File'-Funktion ausführt. Danach muß der BIOS-interne Sektorbuffer auf die Diskette geschrieben und die Routine mit einem 'RET' beendet werden.

Ausgabe der Usernummer

Beim Arbeiten in verschiedenen Userbereichen ist es oft seht störend, daß das CP/M keine Angaben über die aktuelle Usernummer macht. Durch Änderung der 'Set/Get User Code'-Funktion im BDOS kann eine neue Usernummer erkannt und über eine spezielle BIOS-Funktion (hier 'show_user' genannt) dargestellt werden.

```
Änderung von:  BDOS+0D3DH  32 0341   LD    (&user_#),A
in:            BDOS+0D3DH  CD xxxx   CALL  show_user
```

Die 'show_user'-Funktion muß die im A-Register übergebene Usernummer in BDOS+0341H ablegen. Die Anzeige kann entweder durch einen speziellen Tastencode (z.B. im Zusammenhang mit der BIOS-Funktion 'CONIN') oder über eine zusätzliche Bildschirmzeile (falls vorhanden) erfolgen.

Anzeige der Printfunktion

Der aktuelle Status der über CTRL-P schaltbaren Drucker-Protokollfunktion im BDOS, kann durch folgende Änderung dem

BIOS (hier 'show_print' genannt) mitgeteilt werden:

```
Änderung von:  BDOS+0245H  C3 01EF   JP    read_next1
in:            BDOS+0245H  C3 xxxx   JP    show_print
```

Die BIOS-Funktion darf keines der Register verändern. Das Register A enthält eine 1, falls die Protokollfunktion eingeschaltet wurde, sonst 0.

Andere Fehlermeldungen

Die Fehlerroutinen im BDOS werden alle über die 'err_loc_table' an der Adresse BDOS+0009H erreicht. Durch Ersetzen dieser Tabelle, kann die Fehlerbehandlung auch im BIOS durchgeführt werden. Eine weiter Möglichkeit ist auch, die Verweise auf die Fehlertexte in den BDOS-Fehlerroutinen zu ändern:

```
Änderung von:  BDOS+0099H  21 00CA   LD    HL,txt_bad_sector
in:            BDOS+0099H  21 xxxx   LD    HL,xxxx
```

für den 'Bad Sector'-Fehler.

```
Änderung von:  BDOS+00A5H  21 00D5   LD    HL,txt_select
in:            BDOS+00A5H  21 xxxx   LD    HL,xxxx
```

für den 'Select'-Fehler.

```
Änderung von:  BDOS+00ABH  21 00E1   LD    HL,txt_disk_ro
in:            BDOS+00ABH  21 xxxx   LD    HL,xxxx
```

für den 'Disk R/O'-Fehler.

```
Änderung von:  BDOS+00B1H  21 00DC   LD    HL,txt_file_ro
in:            BDOS+00B1H  21 xxxx   LD    HL,xxxx
```

für den 'R/O'-Fehler.

Die neuen Fehlertext müssen entsprechend dem CP/M-Standard, mit '$' abgeschlossen sein.

Warmstart nach einem Schreib/Lesefehler

Nach einem 'Bad Sector'-Fehler kann die fehlerhafte Funktion entweder wiederholt oder durch einen Warmstart abgebrochen werden. Im letzteren Fall greift der CCP wieder auf das zuletzt angewählte Laufwerk zu.
Lag nun der gemeldete Fehler im Directory-Bereich der Diskette, so tritt er beim 'Einloggen' der Diskette wieder auf und ein Abbruch ist nur noch durch Ausschalten des Rechners möglich.
Durch die folgende Änderung kann das Verhalten nach einem

'Bad Sector'-Fehler geändert werden:

```
Änderung von:  BDOS+00A1H  CA 0000   JP    Z,?boot
in:            BDOS+00A1H  CA xxxx   JP    Z,err_wboot
```

Die Routine err_wboot kann z.B. den Wert der Speicherstelle 0004H ändern und dadurch die Anwahl eines anderen Laufwerkes erreichen.

Warmstart nach einem 'Select'-Fehler

Der CCP ändert vor der Anwahl eines neuen Laufwerkes immer erst die Speicherzelle 0004H. Prüft das BIOS vor dem Start des CCP nicht den Inhalt dieser Speicherzelle, sondern übergibt ihn einfach im C-Register, so führt dies zu einem erneuten 'Select'-Fehler.
Die weiter oben genannte 'err_wboot'-Routine kann auch in diesem Fall benutzt werden:

```
Änderung von:  BDOS+00B7H  C3 0000   JP    ?boot
in:            BDOS+00B7H  C3 xxxx   JP    err_wboot
```

Durch diese Änderung wird allerdings auch nach einem 'Disk R/O'- oder 'R/O'-Fehler ein neues Laufwerk angewählt.

Alphabetische Reihenfolge der CCP-Label

?BDOS	0005H
?BUFFER_ADRESS	0088H
?DEFAULT_FCB	005CH
?DISK_BUFFER	0080H
?FIRST_CHAR	008AH
?TPA_START	0100H
§CURRENT_DEFAULT_DRIVE	0004H
ADD_HL_A	0259H
BACK	0008H
BAD_LOAD	0771H
BDOS	0800H
BDOS_CALL	00C3H
BDOS_DISKIO	00F4H
BELL	0007H
BUFFER	0006H
B_CLOSE_FILE	0010H
B_CONSOLE_INPUT	0001H
B_CONSOLE_OUTPUT	0002H
B_CONSOLE_STATUS	000BH
B_DELETE_FILE	0013H
B_MAKE_FILE	0016H
B_OPEN_FILE	000FH
B_READ_BUFFER	000AH
B_READ_SEQU	0014H
B_RENAME_FILE	0017H
B_RESET_DRIVE	0025H
B_RESET_SYSTEM	000DH
B_RETURN_CURRENT	0019H
B_RETURN_DISK	0018H
B_SEARCH_FIRST	0011H
B_SEARCH_NEXT	0012H
B_SELECT_DISK	000EH
B_SET_DMA	001AH
B_SET_GET_USER	0020H
B_WRITE_SEQU	0015H
CCP	0000H
CCP_PROMPT	0382H
CHECK_CONSOLE	01C2H
CHECK_WILDCARD	0301H
CLEAR_REC_NO	02F0H
CLEAR_REC_NO_LOOP	02F2H
CLOSE_FILE	00DAH
CLOSE_SAVE_FILE	05F1H
CMD_ADRESSES	03C1H
CMD_TABLE	0310H
COMMAND_ERROR	0209H
COMPARE_LOOP	01FDH
COMPARE_NEXT_CHAR	033CH
COMPARE_NEXT_CMD	0333H
COMPARE_SERIAL	01F5H
CONOUT	008CH
CONOUT_BC	0092H

Labels

CONVERT_LOOP	01ABH
CR	000DH
CTRLP	0010H
CURRENT_DISK	07EFH
CURRENT_SUBMIT_REC	07CCH
DEFAULT_EXT	0783H
DEL	007FH
DELETE_FILE	00EFH
DELETE_SUBMIT	01DDH
DIR	0477H
DIR_EXIT	051BH
DIR_FILE	048FH
DIR_LOOP	0498H
DIR_NEXT_CHAR	04D9H
DIR_NEXT_IN_LINE	04CCH
DIR_PRINT_CHAR	04F9H
DIR_PRINT_SPACE	04F7H
ENTRY_NOT_USED	050FH
EOF	001AH
ERA	051FH
ERA_SINGLE_FILE	0542H
EXECUTE_CMD	0398H
EXECUTE_PROGRAM	06A5H
FILE_DRIVE_NO	07F0H
FILE_EXISTS	0679H
FILL_EXT_SPACE	02E9H
FILL_FCB_SPACE	02B9H
GET_BYTE_FROM_BUFFER	044BH
GET_CHECK_CHAR	0230H
GET_COMMAND_LINE	0139H
GET_DRIVE_CODE	01D0H
GET_NEXT_CHAR	024FH
GET_NEXT_DIR_ENTRY	050EH
GET_NUMBER	03F8H
GET_NUMBER_EXIT	0433H
GET_NUMBER_LOOP	0408H
GET_USER	0113H
INC_CMD_COUNTER	0354H
INC_TO_NEXT_CMD	034FH
INTERNAL_EXTENSION	07D6H
INTERNAL_FCB	07CDH
INTERNAL_FILENAME	07CEH
INTERNAL_RECORD_#	07EDH
LF	000AH
LOAD_AND_EXECUTE	06C4H
LOAD_NEXT_RECORD	06E1H
MAKE_FILE	0109H
MAKE_UPPER	0130H
MOVE_3_BYTES	0440H
MOVE_B_BYTES	0442H
NEXT_EXTENSION_CHAR	02DBH
NEXT_FILENAME_CHAR	02ABH
NO_FILE	03EAH
NO_NEW_DRIVE	0289H

NO_SPACE	05FBH
NO_SUBMIT	0196H
NO_WILDCARD	0309H
OPEN_FCB	00D0H
OPEN_FILE	00CBH
PRINT_CRLF	0098H
PRINT_LOOP	00ACH
PRINT_NEXT_CHAR	020FH
PRINT_QUESTION_MARK	0222H
PRINT_SPACE	00A2H
PRINT_TEXT	00A7H
PROMPT	003EH
READ_ERROR	03D9H
READ_FCB	00FEH
READ_SEQU	00F9H
REENTER_CCP	0786H
REENTER_NEW_DISK	0789H
REN	0610H
RENAME_FCB	07DDH
RENAME_FILE	010EH
REN_DELIMITER_OK	063FH
REN_DISK_OK	0659H
REN_FILE_NOT_FOUND	0673H
REN_NO_FILE	066DH
RESET_BUFFER	01BAH
RESET_DISK	00B8H
RETURN_CODE	07EEH
RUN_TRANSIENT	074FH
SAVE	05ADH
SAVE_EXIT	0601H
SAVE_LOOP	05D4H
SEARCH_CCP_CMD	032EH
SEARCH_END_OF_EXTENSION	02DFH
SEARCH_END_OF_FILENAME	02AFH
SEARCH_FCB	00E9H
SEARCH_FIRST	00DFH
SEARCH_NEXT	00E4H
SEARCH_PARAMETERS	0730H
SELECT_DISK	00BDH
SELECT_NEW_DISK	0454H
SELECT_OLD_DISK	0466H
SERIAL_NUMBER	0328H
SETUP_CMD_BUFFER	073EH
SETUP_CMD_BUFFER_LOOP	0743H
SETUP_EXTENSION	02C0H
SETUP_EXTENSION_LOOP	02C8H
SETUP_FCB	025EH
SETUP_FCB_OFFSET	0260H
SETUP_FILENAME	0296H
SETUP_FILENAME_LOOP	0298H
SETUP_PARAMETERS	0701H
SET_CHAR_TO_EXT	02D9H
SET_CHAR_TO_FCB	02A9H
SET_COMMAND	01A7H

Labels

SET_DEFAULT_DMA	01D5H
SET_DMA	01D8H
SET_DRIVE	0129H
SET_DRV_USR	011AH
SET_NEW_DRIVE	0290H
SET_USER	0115H
SET_WILDCARD_LOOP	0488H
SHOW_FILE	04D4H
SPACE	0020H
STACK_AREA	07ABH
START_DRIVE_C	035CH
START_NO_COMMAND	0358H
SUBMIT_FCB	07ACH
SUBMIT_FLAG	07ABH
SUBMIT_REC_COUNT	07BBH
SUBMIT_REC_NO	07BAH
TAB	0009H
TRANSIENT_NOT_FOUND	076BH
TXT_ALL_YN	0552H
TXT_BAD_LOAD	077AH
TXT_FILE_EXISTS	0682H
TXT_NO_FILE	03F0H
TXT_NO_SPACE	0607H
TXT_READ_ERROR	03DFH
TYPE	055DH
TYPE_CHAR	0587H
TYPE_CHAR_COUNTER	07F1H
TYPE_ERROR	05A7H
TYPE_EXIT	05A0H
TYPE_LOOP	0574H
USER	068EH
WRITE_SEQU	0104H
WRONG_SERIAL	03CFH

Numerische Reihenfolge der CCP-Label

0000H	CCP
0001H	B_CONSOLE_INPUT
0002H	B_CONSOLE_OUTPUT
0004H	§CURRENT_DEFAULT_DRIVE
0005H	?BDOS
0006H	BUFFER
0007H	BELL
0008H	BACK
0009H	TAB
000AH	B_READ_BUFFER
000AH	LF
000BH	B_CONSOLE_STATUS
000DH	B_RESET_SYSTEM
000DH	CR
000EH	B_SELECT_DISK
000FH	B_OPEN_FILE
0010H	B_CLOSE_FILE
0010H	CTRLP
0011H	B_SEARCH_FIRST
0012H	B_SEARCH_NEXT
0013H	B_DELETE_FILE
0014H	B_READ_SEQU
0015H	B_WRITE_SEQU
0016H	B_MAKE_FILE
0017H	B_RENAME_FILE
0018H	B_RETURN_DISK
0019H	B_RETURN_CURRENT
001AH	B_SET_DMA
001AH	EOF
0020H	B_SET_GET_USER
0020H	SPACE
0025H	B_RESET_DRIVE
003EH	PROMPT
005CH	?DEFAULT_FCB
007FH	DEL
0080H	?DISK_BUFFER
0088H	?BUFFER_ADRESS
008AH	?FIRST_CHAR
008CH	CONOUT
0092H	CONOUT_BC
0098H	PRINT_CRLF
00A2H	PRINT_SPACE
00A7H	PRINT_TEXT
00ACH	PRINT_LOOP
00B8H	RESET_DISK
00BDH	SELECT_DISK
00C3H	BDOS_CALL
00CBH	OPEN_FILE
00D0H	OPEN_FCB
00DAH	CLOSE_FILE
00DFH	SEARCH_FIRST
00E4H	SEARCH_NEXT

Labels

00E9H	SEARCH_FCB
00EFH	DELETE_FILE
00F4H	BDOS_DISKIO
00F9H	READ_SEQU
00FEH	READ_FCB
0100H	?TPA_START
0104H	WRITE_SEQU
0109H	MAKE_FILE
010EH	RENAME_FILE
0113H	GET_USER
0115H	SET_USER
011AH	SET_DRV_USR
0129H	SET_DRIVE
0130H	MAKE_UPPER
0139H	GET_COMMAND_LINE
0196H	NO_SUBMIT
01A7H	SET_COMMAND
01ABH	CONVERT_LOOP
01BAH	RESET_BUFFER
01C2H	CHECK_CONSOLE
01D0H	GET_DRIVE_CODE
01D5H	SET_DEFAULT_DMA
01D8H	SET_DMA
01DDH	DELETE_SUBMIT
01F5H	COMPARE_SERIAL
01FDH	COMPARE_LOOP
0209H	COMMAND_ERROR
020FH	PRINT_NEXT_CHAR
0222H	PRINT_QUESTION_MARK
0230H	GET_CHECK_CHAR
024FH	GET_NEXT_CHAR
0259H	ADD_HL_A
025EH	SETUP_FCB
0260H	SETUP_FCB_OFFSET
0289H	NO_NEW_DRIVE
0290H	SET_NEW_DRIVE
0296H	SETUP_FILENAME
0298H	SETUP_FILENAME_LOOP
02A9H	SET_CHAR_TO_FCB
02ABH	NEXT_FILENAME_CHAR
02AFH	SEARCH_END_OF_FILENAME
02B9H	FILL_FCB_SPACE
02C0H	SETUP_EXTENSION
02C8H	SETUP_EXTENSION_LOOP
02D9H	SET_CHAR_TO_EXT
02DBH	NEXT_EXTENSION_CHAR
02DFH	SEARCH_END_OF_EXTENSION
02E9H	FILL_EXT_SPACE
02F0H	CLEAR_REC_NO
02F2H	CLEAR_REC_NO_LOOP
0301H	CHECK_WILDCARD
0309H	NO_WILDCARD
0310H	CMD_TABLE
0328H	SERIAL_NUMBER

032EH	SEARCH_CCP_CMD
0333H	COMPARE_NEXT_CMD
033CH	COMPARE_NEXT_CHAR
034FH	INC_TO_NEXT_CMD
0354H	INC_CMD_COUNTER
0358H	START_NO_COMMAND
035CH	START_DRIVE_C
0382H	CCP_PROMPT
0398H	EXECUTE_CMD
03C1H	CMD_ADRESSES
03CFH	WRONG_SERIAL
03D9H	READ_ERROR
03DFH	TXT_READ_ERROR
03EAH	NO_FILE
03F0H	TXT_NO_FILE
03F8H	GET_NUMBER
0408H	GET_NUMBER_LOOP
0433H	GET_NUMBER_EXIT
0440H	MOVE_3_BYTES
0442H	MOVE_B_BYTES
044BH	GET_BYTE_FROM_BUFFER
0454H	SELECT_NEW_DISK
0466H	SELECT_OLD_DISK
0477H	DIR
0488H	SET_WILDCARD_LOOP
048FH	DIR_FILE
0498H	DIR_LOOP
04CCH	DIR_NEXT_IN_LINE
04D4H	SHOW_FILE
04D9H	DIR_NEXT_CHAR
04F7H	DIR_PRINT_SPACE
04F9H	DIR_PRINT_CHAR
050EH	GET_NEXT_DIR_ENTRY
050FH	ENTRY_NOT_USED
051BH	DIR_EXIT
051FH	ERA
0542H	ERA_SINGLE_FILE
0552H	TXT_ALL_YN
055DH	TYPE
0574H	TYPE_LOOP
0587H	TYPE_CHAR
05A0H	TYPE_EXIT
05A7H	TYPE_ERROR
05ADH	SAVE
05D4H	SAVE_LOOP
05F1H	CLOSE_SAVE_FILE
05FBH	NO_SPACE
0601H	SAVE_EXIT
0607H	TXT_NO_SPACE
0610H	REN
063FH	REN_DELIMITER_OK
0659H	REN_DISK_OK
066DH	REN_NO_FILE
0673H	REN_FILE_NOT_FOUND

Labels

0679H	FILE_EXISTS
0682H	TXT_FILE_EXISTS
068EH	USER
06A5H	EXECUTE_PROGRAM
06C4H	LOAD_AND_EXECUTE
06E1H	LOAD_NEXT_RECORD
0701H	SETUP_PARAMETERS
0730H	SEARCH_PARAMETERS
073EH	SETUP_CMD_BUFFER
0743H	SETUP_CMD_BUFFER_LOOP
074FH	RUN_TRANSIENT
076BH	TRANSIENT_NOT_FOUND
0771H	BAD_LOAD
077AH	TXT_BAD_LOAD
0783H	DEFAULT_EXT
0786H	REENTER_CCP
0789H	REENTER_NEW_DISK
07ABH	STACK_AREA
07ABH	SUBMIT_FLAG
07ACH	SUBMIT_FCB
07BAH	SUBMIT_REC_NO
07BBH	SUBMIT_REC_COUNT
07CCH	CURRENT_SUBMIT_REC
07CDH	INTERNAL_FCB
07CEH	INTERNAL_FILENAME
07D6H	INTERNAL_EXTENSION
07DDH	RENAME_FCB
07EDH	INTERNAL_RECORD_#
07EEH	RETURN_CODE
07EFH	CURRENT_DISK
07F0H	FILE_DRIVE_NO
07F1H	TYPE_CHAR_COUNTER
0800H	BDOS

Alphabetische Reihenfolge der BDOS-Label

_CLOSE_FILE	08A2H
_COMPUTE_FILE_SIZE	0BD2H
_MAKE_FILE	0924H
_OPEN_FILE	0851H
_READ_RANDOM	0B93H
_READ_SEQU	09BCH
_RENAME_FILE	0816H
_SET_FILE_ATTR	083BH
_WRITE_RANDOM	0B9CH
_WRITE_SEQU	09FEH
?§ALV	0DBFH
?§CSV	0DBDH
?§DIRBUF	0DB9H
?§DPB	0DBBH
?§END_OF_ENTRIES	0DB3H
?§SEARCH_FCB	0DD9H
?§TRACK_#	0DB5H
?§TRACK_RECORD_#	0DB7H
?§XLTAB	0DD0H
?BDOS_EXIT	0D74H
?BOOT	0000H
?CLOSE_FILE	0CA5H
?COMPUTE_FILE_SIZE	0D4DH
?CONSOLE_INPUT	02C8H
?CONSOLE_OUTPUT	0190H
?DELETE_FILE	0CD7H
?DIRECT_IO	02D4H
?DUMMY	0304H
?ERR_BAD_SECTOR	0099H
?ERR_DISK_RO	00ABH
?ERR_FILE_RO	00B1H
?ERR_SELECT	00A5H
?GET_ALLOCATION_ADDR	0D11H
?GET_CONSOLE_STATUS	02FEH
?GET_IO_BYTE	02EDH
?GET_PARAMETER_ADDR	0D26H
?GET_RO_VEC	0D17H
?MAKE_FILE	0CECH
?OPEN_FILE	0C9CH
?PRINT_STRING	02F8H
?READER_INPUT	02CEH
?READ_CONSOLE_BUFFER	01E1H
?READ_RANDOM	0D41H
?READ_SEQU	0CE0H
?RENAME_FILE	0CF5H
?RESET_DISK	0D53H
?RESET_DISK_SYSTEM	0C83H
?RETURN_CURRENT_DISK	0D04H
?RETURN_LOGIN_VEC	0CFEH
?RETURN_VERSION_#	0C7EH
?SEARCH_FIRST	0CABH
?SEARCH_NEXT	0CC8H

Labels

?SELECT_DISK	0C45H
?SET_DMA	0D0AH
?SET_FILE_ATTR	0D1DH
?SET_GET_USER_CODE	0D2DH
?SET_IO_BYTE	02F3H
?SET_RANDOM_RECORD	0C0EH
?WRITE_PROTECT_DISK	052CH
?WRITE_RANDOM	0D47H
?WRITE_SEQU	0CE6H
?WRITE_ZERO_RANDOM	0D9BH
§ABSOLUTE_RECORD_#	0DE5H
§AL0_AL1	0DCAH
§AUTO_SELECTED?	0DDEH
§BDOS_DMA_ADDRESS	0DB1H
§BLM	0DC4H
§BLOCK_#	0DE5H
§BLOCK_OFFSET	0DD7H
§BLOCK_START_RECORD	0DE7H
§BSH	0DC3H
§CHARS_IN_LINE	030AH
§CKS	0DCCH
§COMPARE_COUNT	0DD8H
§CURRENT_DISK	0342H
§CURRENT_EXTEND_#	0DE2H
§CURRENT_SEQU_REC	0DE3H
§DEFAULT_DISK	0DDFH
§DIR_ENTRY_#	0DEAH
§DIR_OFFSET	0DE9H
§DIR_RECORD_#	0DECH
§DISK_BYTE	0004H
§DISK_NAME	00C6H
§DRM	0DC8H
§DSM	0DC6H
§ENTRY_BYTE	0DD6H
§ENTRY_CLOSED?	0DD2H
§ENTRY_WORD	0343H
§EXIT_VALUE	0345H
§EXM	0DC5H
§EXT_RECORD_COUNT	0DE1H
§FCB_DISK_#	0DE0H
§FILE_FOUND?	0DD4H
§INPUT_START	030BH
§IO_BYTE	0003H
§LARGE_DSM?	0DDDH
§LOGIN_VEC	0DAFH
§MAKE_FCB	0DACH
§OFF	0DCEH
§OUTPUT_COUNTER	030CH
§PRINT_FLAG	030DH
§READ_OR_WRITE?	0DD3H
§RO_VEC	0DADH
§SEQUENTIAL?	0DD5H
§SPT	0DC1H
§STATUS_CHAR	030EH

Labels

CHECK_CTRLR	026BH
CHECK_CTRLU	025FH
CHECK_CTRLX	0248H
CHECK_DEL	0216H
CHECK_DIR_OFFSET	0619H
CHECK_DIR_RECORD	0605H
CHECK_DISK_RO	051EH
CHECK_EXTEND_#	0707H
CLEAR_EXIT_VALUE	0AFEH
CLEAR_LINE	01B1H
CLEAR_LOOP	01B9H
CLOSE_BLOCK_LOOP	08CDH
CLOSE_BYTE_BLOCK	08DBH
CLOSE_NEXT_BLOCK	08FDH
CLOSE_WORD_BLOCK	0894H
CLR_ALV_LOOP	06B1H
COMPARE_EXTEND	0773H
COMPARE_NEXT_CHAR	077CH
COMPARE_NEXT_LOOP	0753H
COMPUTE_SIZE_LOOP	0BE4H
CORRECT_RND_ACCESS	0B7FH
CR	000DH
CTRLC	0003H
CTRLE	0005H
CTRLP	0010H
CTRLR	0012H
CTRLU	0015H
CTRLX	0018H
CTRLX_LOOP	024EH
DEFAULT_DMA	0080H
DEL	007FH
DELETE_TABS	028AH
DIFFERENT_EXTEND	0B47H
DIRECT_CONIN	02E0H
DIR_OFFSET_LOOP	0620H
DIR_READ	05D4H
DIR_WRITE	05C6H
DISK_FULL	0A43H
DISK_RO?	0554H
DISK_RO_ERROR	0558H
END_OF_FILE	09FBH
END_OF_INPUT	02C1H
ENTRY_#_FREE?	057FH
ENTRY_FOUND	09ACH
ENTRY_NOT_FOUND	0794H
ENTRY_NOT_RIGHT	06F6H
ENTRY_RO?	0544H
ERASE_FILES	079CH
ERASE_LOOP	07A4H
ERROR_ENTRY	034AH
ERROR_TEXT	00BAH
ERR_BOOT	00B4H
ERR_LOC_TABLE	0009H
FCB_RO?	0547H

FILE_RO_ERROR	054EH
GET_ALV_BIT	0635H
GET_ALV_LOOP	0656H
GET_BLOCK_#	045EH
GET_BLOCK_OFFSET	043EH
GET_BLOCK_WORD	0471H
GET_CHAR	00FBH
GET_ENTRY_POINTER	055EH
GET_EXTEND_POINTER	04A6H
GET_RC_CR_POINTER	04AEH
GET_RECORD_#S	04BBH
GET_S2	0569H
HOME_DISK	03A1H
INPUT_CHAR	0106H
JP_UPDATE_RECORD_#S	0B00H
LF	000AH
LOGICAL_SELECT	06A3H
LOGIN_CURRENT_DISK	0C21H
LOGIN_NEXT_ENTRY	06D2H
MAKE_CLEAR_LOOP	0946H
MAX_FUNC_#	0029H
MOVE_DE_HL	034FH
MOVE_DIRBUF_TO_DMA	05E9H
MOVE_LOOP	0350H
NEW_BLOCK_WORD	0A64H
NEW_END_OF_ENTRIES?	058CH
NEW_EXTEND_OPENED	09AFH
NEW_EXT_TOO_BIG	0983H
NOT_BACK	0179H
NOT_CTRLC	02BDH
NOT_GREATER	0C06H
NO_AUTO_SLCT	0C75H
NO_BLOCK_ALLOC	069DH
POS_CR	0020H
POS_DX	0010H
POS_EX	000CH
POS_RO	0009H
POS_R0	0021H
POS_RC	000FH
DIFF_RC_CR	0011H
POS_S1	000DH
POS_EG	000EH
OPEN_ENTRY	085AH
OPEN_NEXT_EXTEND	095AH
PHYSICAL_SELECT	0359H
POP_HL	0C0CH
PRINT_BACK	01ACH
PRINT_BACKSPACE	01A4H
PRINT_CHAR	0148H
PRINT_CR_LF	01C9H
PRINT_TEXT	01D3H
READ_NEXT1	01EFH
READ_NEXT2	01F1H
READ_NEXT_RECORD	09E6H

Labels

READ_RECORD	03B2H
READ_RND_SEQ	09C1H
READ_WRITE_ERROR	0305H
REC_OFFSET_LOOP	0490H
RENAME_LOOP	0827H
RESET_ENTRY_#	05FEH
RESTORE_DMA	05DAH
RETURN_BYTE	0301H
RETURN_DIR_CODE	0783H
RETURN_FOUND_FLAG	0701H
RETURN_WORD	0D29H
RETYPE_LINE	0270H
RETYPE_LOOP	0278H
RET_TO_USER	0D91H
RND_SEEK_ERROR	0B8BH
SAME_BYTE_BLOCK?	08E1H
SAVE_CHAR	02A6H
SAVE_STATUS	0142H
SEARCH_DIR_FIRST	0718H
SEARCH_DIR_NEXT	072DH
SEARCH_MAKE	074AH
SEARCH_NEXT_EXTEND	098EH
SEARCH_NOW	0CC2H
SELECT_ERROR	0347H
SELECT_NEW_OK	039DH
SERIAL_#	0000H
SET_§ABSOLUTE_RECORD#	048AH
SET_§BLOCK_#	0477H
SET_ALV_LOOP	0664H
SET_ATTR_LOOP	0840H
SET_DIR_TRK_SEC	03C3H
SET_DISK_VEC	050BH
SET_DMA_TO_DIRBUF	05E0H
SET_DMA_TO_HL	05E3H
SET_DSK_TRK_SEC	03D1H
SET_RECORD_COUNT	088BH
SET_S2	0578H
SET_USER_CODE	0D3BH
SHIFT_LEFT_HL	0504H
SHIFT_LEFT_LOOP	0505H
SHIFT_RIGHT_HL	04EAH
SHIFT_RIGHT_LOOP	04EBH
SHOW_CHAR	017FH
SHOW_ERROR	00E5H
SHOW_INPUT	02A9H
SHOW_TAB	0196H
SPACE	0020H
STACK	0341H
STILL_IN_CURRENT_EXT	0AD2H
SUB_DE_HL	0595H
TAB	0009H
TEST_BLOCK_#	068EH
TEST_CSV	059EH
TEST_RW_STATUS	03BBH

TRACK_DEC	03E4H
TRACK_FOUND	040FH
TRACK_INC	03FAH
TRANS_SEQ_TO_RND	0BA5H
TRNS_RND_TO_SEQ	0B03H
TRUE_STATUS	0145H
TXT_BAD_SECTOR	00CAH
TXT_DISK_RO	00E1H
TXT_FILE_RO	00DCH
TXT_SELECT	00D5H
UPDATE_ALV	066BH
UPDATE_ALV_LOOP	0675H
UPDATE_CSV	059CH
UPDATE_CSV_BYTE	05C4H
UPDATE_RECORD_#S	04D2H
UPDATE_SEQU_#	04DEH
VERSION_#	0022H
WORD_BLOCK_CLOSE	08E8H
WORD_VALUE	0688H
WRITE_ACTUAL_DATA	0ABBH
WRITE_CLOSED_ENTRY	0917H
WRITE_COMPLETE_ENTRY	07FDH
WRITE_DIR_RECORD	0810H
WRITE_NEW_BLOCK	0A48H
WRITE_NEW_ENTRY	0A3BH
WRITE_PARTIAL_ENTRY	0801H
WRITE_RECORD	03B8H
WRITE_RND_SEQ	0A03H
XOFF	0013H
XON	0011H
ZERO_S2	0572H
ZERO_TRANS_R_TO_S	0B07H
ZERO_WRITE_BUFFER	0A8CH

Labels

Numerische Reihenfolge der BDOS-Label

0000H	?BOOT
0000H	SERIAL_#
0003H	§IO_BYTE
0003H	CTRLC
0004H	§DISK_BYTE
0005H	CTRLE
0006H	BDOS
0007H	BELL
0008H	BACK
0009H	ERR_LOC_TABLE
0009H	POS_RO
0009H	TAB
000AH	LF
000CH	POS_EX
000DH	CR
000DH	POS_S1
000EH	POS_EG
000FH	POS_RC
0010H	CTRLP
0010H	POS_DX
0011H	BDOS1
0011H	DIFF_RC_CR
0011H	XON
0012H	CTRLR
0013H	XOFF
0015H	CTRLU
0018H	CTRLX
0020H	POS_CR
0020H	SPACE
0021H	POS_R0
0022H	VERSION_#
0029H	MAX_FUNC_#
0047H	BDOS_LOC_TABLE
007FH	DEL
0080H	DEFAULT_DMA
0099H	?ERR_BAD_SECTOR
00A5H	?ERR_SELECT
00ABH	?ERR_DISK_RO
00B1H	?ERR_FILE_RO
00B4H	ERR_BOOT
00BAH	ERROR_TEXT
00C6H	§DISK_NAME
00CAH	TXT_BAD_SECTOR
00D5H	TXT_SELECT
00DCH	TXT_FILE_RO
00E1H	TXT_DISK_RO
00E5H	SHOW_ERROR
00FBH	GET_CHAR
0106H	INPUT_CHAR
0114H	CHECK_CHAR
0123H	CHAR_XOFF?
0142H	SAVE_STATUS

```
0145H                   TRUE_STATUS
0148H                   PRINT_CHAR
0162H                   ADJUST_TAB
0179H                   NOT_BACK
017FH                   SHOW_CHAR
0190H                   ?CONSOLE_OUTPUT
0196H                   SHOW_TAB
01A4H                   PRINT_BACKSPACE
01ACH                   PRINT_BACK
01B1H                   CLEAR_LINE
01B9H                   CLEAR_LOOP
01C9H                   PRINT_CR_LF
01D3H                   PRINT_TEXT
01E1H                   ?READ_CONSOLE_BUFFER
01EFH                   READ_NEXT1
01F1H                   READ_NEXT2
0216H                   CHECK_DEL
0226H                   CHECK_CTRLE
0237H                   CHECK_CTRLP
0248H                   CHECK_CTRLX
024EH                   CTRLX_LOOP
025FH                   CHECK_CTRLU
026BH                   CHECK_CTRLR
0270H                   RETYPE_LINE
0278H                   RETYPE_LOOP
028AH                   DELETE_TABS
0299H                   BACK_TAB
02A6H                   SAVE_CHAR
02A9H                   SHOW_INPUT
02BDH                   NOT_CTRLC
02C1H                   END_OF_INPUT
02C8H                   ?CONSOLE_INPUT
02CEH                   ?READER_INPUT
02D4H                   ?DIRECT_IO
02E0H                   DIRECT_CONIN
02EDH                   ?GET_IO_BYTE
02F3H                   ?SET_IO_BYTE
02F8H                   ?PRINT_STRING
02FEH                   ?GET_CONSOLE_STATUS
0301H                   RETURN_BYTE
0304H                   ?DUMMY
0305H                   READ_WRITE_ERROR
030AH                   $CHARS_IN_LINE
030BH                   $INPUT_START
030CH                   $OUTPUT_COUNTER
030DH                   $PRINT_FLAG
030EH                   $STATUS_CHAR
030FH                   $USER_STACK
0341H                   $USER_#
0341H                   STACK
0342H                   $CURRENT_DISK
0343H                   $ENTRY_WORD
0345H                   $EXIT_VALUE
0347H                   SELECT_ERROR
```

Labels

034AH	ERROR_ENTRY
034FH	MOVE_DE_HL
0350H	MOVE_LOOP
0359H	PHYSICAL_SELECT
039DH	SELECT_NEW_OK
03A1H	HOME_DISK
03B2H	READ_RECORD
03B8H	WRITE_RECORD
03BBH	TEST_RW_STATUS
03BDH	BAD_SECTOR_ERROR
03C3H	SET_DIR_TRK_SEC
03D1H	SET_DSK_TRK_SEC
03E4H	TRACK_DEC
03FAH	TRACK_INC
040FH	TRACK_FOUND
043EH	GET_BLOCK_OFFSET
0445H	BLOCK_1_LOOP
0453H	BLOCK_2_LOOP
045CH	ADD_A_B
045EH	GET_BLOCK_#
0471H	GET_BLOCK_WORD
0477H	SET_§BLOCK_#
0484H	CHECK_§BLOCK_#
048AH	SET_§ABSOLUTE_RECORD#
0490H	REC_OFFSET_LOOP
04A6H	GET_EXTEND_POINTER
04AEH	GET_RC_CR_POINTER
04BBH	GET_RECORD_#S
04D2H	UPDATE_RECORD_#S
04DEH	UPDATE_SEQU_#
04EAH	SHIFT_RIGHT_HL
04EBH	SHIFT_RIGHT_LOOP
04F7H	CHECKSUM_DIRBUF
04FDH	CHECKSUM_LOOP
0504H	SHIFT_LEFT_HL
0505H	SHIFT_LEFT_LOOP
050BH	SET_DISK_VEC
051EH	CHECK_DISK_RO
052CH	?WRITE_PROTECT_DISK
0544H	ENTRY_RO?
0547H	FCB_RO?
054EH	FILE_RO_ERROR
0554H	DISK_RO?
0558H	DISK_RO_ERROR
055EH	GET_ENTRY_POINTER
0564H	ADD_HL_A
0569H	GET_S2
0572H	ZERO_S2
0578H	SET_S2
057FH	ENTRY_#_FREE?
058CH	NEW_END_OF_ENTRIES?
0595H	SUB_DE_HL
059CH	UPDATE_CSV
059EH	TEST_CSV

```
05C4H   UPDATE_CSV_BYTE
05C6H   DIR_WRITE
05D4H   DIR_READ
05DAH   RESTORE_DMA
05E0H   SET_DMA_TO_DIRBUF
05E3H   SET_DMA_TO_HL
05E9H   MOVE_DIRBUF_TO_DMA
05F5H   ALL_ENTRIES_COMPARED?
05FEH   RESET_ENTRY_#
0605H   CHECK_DIR_RECORD
0619H   CHECK_DIR_OFFSET
0620H   DIR_OFFSET_LOOP
0635H   GET_ALV_BIT
0656H   GET_ALV_LOOP
065CH   CHANGE_ALV_BIT
0664H   SET_ALV_LOOP
066BH   UPDATE_ALV
0675H   UPDATE_ALV_LOOP
0688H   WORD_VALUE
068EH   TEST_BLOCK_#
069DH   NO_BLOCK_ALLOC
06A3H   LOGICAL_SELECT
06B1H   CLR_ALV_LOOP
06D2H   LOGIN_NEXT_ENTRY
06F6H   ENTRY_NOT_RIGHT
0701H   RETURN_FOUND_FLAG
0707H   CHECK_EXTEND_#
0718H   SEARCH_DIR_FIRST
072DH   SEARCH_DIR_NEXT
074AH   SEARCH_MAKE
0753H   COMPARE_NEXT_LOOP
0773H   COMPARE_EXTEND
077CH   COMPARE_NEXT_CHAR
0783H   RETURN_DIR_CODE
0794H   ENTRY_NOT_FOUND
079CH   ERASE_FILES
07A4H   ERASE_LOOP
07BEH   ALLOCATE_NEXT_BLOCK
07C0H   ALLOCATE_NEXT_LOOP
07D1H   ALLOCATE_HIGHER
07ECH   ALLOCATE_BLOCK
07F4H   ALLOCATE_LOWER
07FDH   WRITE_COMPLETE_ENTRY
0801H   WRITE_PARTIAL_ENTRY
0810H   WRITE_DIR_RECORD
0816H   _RENAME_FILE
0827H   RENAME_LOOP
083BH   _SET_FILE_ATTR
0840H   SET_ATTR_LOOP
0851H   _OPEN_FILE
085AH   OPEN_ENTRY
088BH   SET_RECORD_COUNT
0894H   CLOSE_WORD_BLOCK
08A2H   _CLOSE_FILE
```

Labels

08CDH	CLOSE_BLOCK_LOOP
08D4H	BYTE_BLOCK_CLOSE
08DBH	CLOSE_BYTE_BLOCK
08E1H	SAME_BYTE_BLOCK?
08E8H	WORD_BLOCK_CLOSE
08FDH	CLOSE_NEXT_BLOCK
0917H	WRITE_CLOSED_ENTRY
091FH	CANT_CLOSE
0924H	_MAKE_FILE
0946H	MAKE_CLEAR_LOOP
095AH	OPEN_NEXT_EXTEND
0983H	NEW_EXT_TOO_BIG
098EH	SEARCH_NEXT_EXTEND
09ACH	ENTRY_FOUND
09AFH	NEW_EXTEND_OPENED
09B6H	CANT_EXTEND_FILE
09BCH	_READ_SEQU
09C1H	READ_RND_SEQ
09E6H	READ_NEXT_RECORD
09FBH	END_OF_FILE
09FEH	_WRITE_SEQU
0A03H	WRITE_RND_SEQ
0A3BH	WRITE_NEW_ENTRY
0A43H	DISK_FULL
0A48H	WRITE_NEW_BLOCK
0A64H	NEW_BLOCK_WORD
0A6CH	BLOCK_WRITE_NEW
0A6EH	BLOCK_WRITE_NORMAL
0A8CH	ZERO_WRITE_BUFFER
0A9AH	BLOCK_WRITE_LOOP
0ABBH	WRITE_ACTUAL_DATA
0AD2H	STILL_IN_CURRENT_EXT
0AFEH	CLEAR_EXIT_VALUE
0B00H	JP_UPDATE_RECORD_#S
0B03H	TRNS_RND_TO_SEQ
0B07H	ZERO_TRANS_R_TO_S
0B47H	DIFFERENT_EXTEND
0B7FH	CORRECT_RND_ACCESS
0B84H	BAD_RND_ACCESS
0B8BH	RND_SEEK_ERROR
0B93H	_READ_RANDOM
0B9CH	_WRITE_RANDOM
0BA5H	TRANS_SEQ_TO_RND
0BD2H	_COMPUTE_FILE_SIZE
0BE4H	COMPUTE_SIZE_LOOP
0C06H	NOT_GREATER
0C0CH	POP_HL
0C0EH	?SET_RANDOM_RECORD
0C21H	LOGIN_CURRENT_DISK
0C45H	?SELECT_DISK
0C51H	AUTO_SELECT
0C75H	NO_AUTO_SLCT
0C7EH	?RETURN_VERSION_#
0C83H	?RESET_DISK_SYSTEM

```
0C9CH   ?OPEN_FILE
0CA5H   ?CLOSE_FILE
0CABH   ?SEARCH_FIRST
0CC2H   SEARCH_NOW
0CC8H   ?SEARCH_NEXT
0CD7H   ?DELETE_FILE
0CE0H   ?READ_SEQU
0CE6H   ?WRITE_SEQU
0CECH   ?MAKE_FILE
0CF5H   ?RENAME_FILE
0CFEH   ?RETURN_LOGIN_VEC
0D04H   ?RETURN_CURRENT_DISK
0D0AH   ?SET_DMA
0D11H   ?GET_ALLOCATION_ADDR
0D17H   ?GET_RO_VEC
0D1DH   ?SET_FILE_ATTR
0D26H   ?GET_PARAMETER_ADDR
0D29H   RETURN_WORD
0D2DH   ?SET_GET_USER_CODE
0D3BH   SET_USER_CODE
0D41H   ?READ_RANDOM
0D47H   ?WRITE_RANDOM
0D4DH   ?COMPUTE_FILE_SIZE
0D53H   ?RESET_DISK
0D74H   ?BDOS_EXIT
0D91H   RET_TO_USER
0D9BH   ?WRITE_ZERO_RANDOM
0DACH   §MAKE_FCB
0DADH   §RO_VEC
0DAFH   §LOGIN_VEC
0DB1H   §BDOS_DMA_ADDRESS
0DB3H   ?§END_OF_ENTRIES
0DB5H   ?§TRACK_#
0DB7H   ?§TRACK_RECORD_#
0DB9H   ?§DIRBUF
0DBBH   ?§DPB
0DBDH   ?§CSV
0DBFH   ?§ALV
0DC1H   §SPT
0DC3H   §BSH
0DC4H   §BLM
0DC5H   §EXM
0DC6H   §DSM
0DC8H   §DRM
0DCAH   §AL0_AL1
0DCCH   §CKS
0DCEH   §OFF
0DD0H   ?§XLTAB
0DD2H   §ENTRY_CLOSED?
0DD3H   §READ_OR_WRITE?
0DD4H   §FILE_FOUND?
0DD5H   §SEQUENTIAL?
0DD6H   §ENTRY_BYTE
0DD7H   §BLOCK_OFFSET
```

Labels

0DD8H	§COMPARE_COUNT
0DD9H	?§SEARCH_FCB
0DDDH	§LARGE_DSM?
0DDEH	§AUTO_SELECTED?
0DDFH	§DEFAULT_DISK
0DE0H	§FCB_DISK_#
0DE1H	§EXT_RECORD_COUNT
0DE2H	§CURRENT_EXTEND_#
0DE3H	§CURRENT_SEQU_REC
0DE5H	§ABSOLUTE_RECORD_#
0DE5H	§BLOCK_#
0DE7H	§BLOCK_START_RECORD
0DE9H	§DIR_OFFSET
0DEAH	§DIR_ENTRY_#
0DECH	§DIR_RECORD_#
0E00H	BIOS
0E00H	B_CBOOT
0E03H	B_WBOOT
0E06H	B_CONST
0E09H	B_CONIN
0E0CH	B_CONOUT
0E0FH	B_LIST
0E12H	B_PUNCH
0E15H	B_READER
0E18H	B_HOME
0E1BH	B_SELDSK
0E1EH	B_SETTRK
0E21H	B_SETSEC
0E24H	B_SETDMA
0E27H	B_READ
0E2AH	B_WRITE
0E2DH	B_LISTST
0E30H	B_SECTRN

ISBN 3-925074-11-2