# CP/M 2.2 Interna

**Klaus Kämpf: CP/M 2.2 Assembler-Listing**
**ISBN 3-925074-II-2**
**1. Auflage, Dezember 1985**
**© 1985 by 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

## Grundlagen

### 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 größer Vorteil dieses Aufbaus war, das 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.

## 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, das, 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 Kompatibilitat 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, das 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 Anwenderprogramme 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:

<ditaa noedgesep>

```
       +----------------+ oberes Speicherende
       |                |
       |      BIOS      |
       |                |
       +----------------+
       |                |
       |      BDOS      |
       |                |
```

FBASE: +———————-+ Startadresse des FDOS

```
       |                |
       |      CCP       |
       |                |
```

CBASE: +———————-+ Startadresse des CCP

```
       |                |
       |      TPA       |
       |                |
```

TBASE: +———————-+ 0100H

```
       |  System-       |
       |     parameter  |
```

BOOT: +———————-+ 0000H </ditaa>

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 das 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, das 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 Belegungstabeile, 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.
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, das 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 der 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 moglichst 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 Sektor 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 CP/M 2.2 maximal ~~16384~~ 8192 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-gleichzeitig 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 Eintragen pro File ist im CP/M 1.4 auf 16 begrenzt, sodas ein File höchstens eine Größe von 256 kbytes erreichen kann.

### Extends

Diese Zahlweise 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 Zahlweise der Recordnummern von 0 bis 127 zu erhalten.

1 Extend = 128 Records = 16 kByte

### 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.

## Datenformate

### 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 Eintragen zu einem File, immer relativ zum Fileanfang 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, das 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 Blocknummern.
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, so daß ein bestimmter Extend schon beim Öffnen erreicht wird.

Die Anwahl einer Extendgruppe 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 RRN ergibt sich die Extendnummer EX aus der Division durch 128. Der Divisionrest ist die Recordnummer CR innerhalb des Extends. Diese Werte müssen aber immer vor dem Öffnen gesetzt sein, damit das BDOS die richtige Blocktabelle in den FCB überträgt.

```
RRN -> SRN:
    RRN = 128 + EX * 128 + CR
SRN -> RRN:
    EX = RRN / 128
    CR = RRN modulo 128
```

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 EG, Extend EX und Recordnummer CR.

Zwischen der Random Record Nummer RRN und der Sequential Record Nummer SRN 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
```

# Diskettenparameter

## 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)
```

## Diskettenparameter

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 Laufwerkswechsel 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, das 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, das 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 (DSM+1) 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 8 Bit    EXM 16 Bit
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.

//**04/2019:** Unter CPM 2.2 sind nachgewiesen aufgrund der 16Bit-Arithmetik nur Laufwerkwerke bis maximal 8 Mbyte nutzbar![1] Mit anderen BDOS-Varianten können größere Laufwerke genutzt werden.[2] //

**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.

# BDOS-Funktionen

## 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 Registerbelegung angegeben.

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

| **Nummer:** | 0 |
|---|---|
| **Bezeichnung:** | System Reset<br>Warmstart |
| **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<br>Consoleneingabe |
|---|---|
| **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<br>Consolenausgabe |
| **I:** | C = 02H<br>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<br>'Reader'-Eingabe |
| **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<br>'Punch'-Ausgabe |
| **I:** | C = 04H<br>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<br>Druckerausgabe |
| **I:** | C = 05H<br>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<br>Direkte Consolenein/ausgabe |
| **I:** | C = 06H<br>E = 0FFH für Consoleneingabe<br>E = 0FEH für Consolen-Status prüfen<br>E = ASCII Zeichen zur Consolenausgabe sonst |
| **O:** | A = Zeichen von der Console (falls E = 0FFH)<br>A = Consolen-Status (falls E = 0FEH)<br>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<br>I/O Byte holen |
| **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/Ausgabekanale vorgenommen werden, dies hat jedoch keinen Einfluss auf das BDOS.

| | |
|---|---|
| **Nummer:** | 8 |
| **Bezeichnung:** | Set I/O Byte<br>I/O Byte setzen |

| | |
|---|---|
| **I:** | C = 08H<br>E = I/O-Byte |
| **O:** | - |

Ausgeführte Funktion:

Der im E-Register ubergebene 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<br>Zeichenkette ausgeben |
| **I:** | C = 09H<br>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<br>Zeile von der Console einlesen |
| **I:** | C = 0AH<br>DE → Zeilenbuffer |
| **O:** | Zeilenbuffer gefüllt mit den eingegebenen Zeichen |

Ausgeführte Funktion:

Alle folgende Zeichen von der Consoleneingabe werden in den Zeilenpuffer ubernommen 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<br>Consolenstatus testen |
| **I:** | C = 0BH |
| **O:** | A = FFH, falls ein Zeichen von der Console ansteht<br>= 00H sonst |

Ausgeführte Funktion:

Es wird gepruft, 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<br>CP/M Versionsnummer zurückmelden |
| **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.

Fix Me!

1)

P.Schorn, 04/2019: Ich habe ein Experiment mit einer 16 mb und einer 32 mb Disk gemacht. In beiden Fällen konnte ich zwar STAT überzeugen, dass die Disk so gross ist, aber ich konnte sie nicht über 8

mb hinaus füllen. Bei überschreiten dieser Grenze sind dann BDOS Fehler aufgetreten.
2)

https://groups.google.com/forum/#!topic/comp.os.cpm/rFagrp7WT9E Reason they only do 16bit math and the limit is the total number of sectors not the total number of allocation blocks. If it were the latter it could go to 1GB. If you want that then look at P2dos, Novados, suprbdos, Zrdos, Dosplus. Before someone else chimes in we also had DRIs improved but those only go to 32mb namely CP/m+ and MPM.

From:
https://hc-ddr.hucki.net/wiki/ - **Homecomputer DDR**

Permanent link:
**https://hc-ddr.hucki.net/wiki/doku.php/cpm/systemdoku?rev=1713167576**

Last update: **2024/04/15 07:52**