SoftwareService

CP/M – ein Betriebssystem für jedermann

Franzis-Software-Service
Franzis-Verlag GmbH
Karlstraße 37–41
8000 München 2

Franzis'

CP/M - EIN BETRIEBSSYSTEM FÜR JEDERMANN

Franzis-Software-Service, Karlsstr. 37-41, 8000 München 2

Inhaltsverzeichnis: Seite:

1. Kommandoeingabe

Kommandos werden in Form einer vollständigen Kommandozeile eingegeben. Eine Kommandozeile besteht aus dem Kommandonamen, einem optionalen Parameterteil und einem carriage Return. Der Parameterteil besteht aus den Dateispezifikationen und Kommandooptionen. Zwischen Kommandonamen und Parameterteil steht ein Leerzeichen. Die Dateiangaben bestehen aus einem dreiteiligen Dateinamen:

d:	bezeichnet das Diskettenlaufwerk (A ... P) Wird das angemeldete Laufwerk verwendet, kann diese Angabe entfallen.
Dateiname	Der Dateiname (primärer Dateiname) bezeichnet eine Datei. Er kann 1 - 8 alphanumerische Zeichen lang sein.
Dateityp	Die Angabe des Dateityps ist optimal. Wird ein Dateityp angegeben, der aus bis zu 3 Zeichen bestehen kann, dann muß er durch einen Punkt vom Dateinamen getrennt werden.
Beispiele:	Datei D: Datei Datei. Dateityp D: Datei. Dateityp

Bestimmte CP/M-Kommandos können eine Mehrzahl von Dateien auswerten und verarbeiten. Die Angabe von Dateigruppen erfolgt durch Einstreuen von sog. "Wildcard"-Zeichen in den Dateinamen. Die beiden "Wildcard"-Zeichen von CP/M sind das Fragezeichen, das einen einzelnen Buchstaben ersetzt und der Stern, der einen ganzen Namensteil ersetzt. Um die Kommandozeile korrigieren zu können, bevor die Returntaste gedrückt wurde, gibt es einige Steuerzeichen.

Backspace	bewegt die Schreibmarke einen Schritt zurück und löscht das zuletzt eingegebene Zeichen.
Control/C	wird am Zeilenanfang eingegeben, um einen CP/M-Warmstart auszulösen.
DEL	dasselbe wie Rubout
Control/E	erzeugt auf der Console einen Zeilenvorschub, der aber nicht die Kommandozeile abschließt.
Control/H	dasselbe wie Backspace
Control/J	Zeilenvorschub, beendet die Eingabe genauso wie Return
Control/M	dasselbe wie Return
Control/P	ist eine Umschalttaste, die den Drucker aktiviert. Ein Druck auf Control/P hat zur Folge, daß die gesamte Consolaktivität auf dem Drucker mitprotokolliert wird. Ein weiterer Druck dieser Taste schaltet die Protokollierung wieder aus.
Control/R	wiederholt die soeben eingegebene Kommandozeile. Diese Taste ist nur sinnvoll, wenn vorher mit Rubout oder DEL gearbeitet wurde.
Return	Abschließen der Kommandozeile und Absenden des Kommandos.
Rubout	löscht das zuletzt eingegebene Zeichen. Das Zeichen wird auf dem Bildschirm wieder als Echo ausgegeben. Die Schreibmarke bewegt sich weiter nach rechts.
Control/S	hält die Consolausgabe zeitweise an. Ein weiterer Druck auf diese Taste läßt die Ausgabe weiterlaufen.
Control/O	löscht die eingegebene Zeile, gibt ein Doppelkreuz aus, bewegt die Schreibmarke eine Zeile nach unten und erwartet die Eingabe einer neuen Kommandozeile.
Control/X	löscht alle Zeichen in der Kommandozeile. Die Schreibmarke bewegt sich wieder ganz an den linken Rand.
Control/Z	Separator für Zeichenketten oder -felder in der Kommandozeile (dargestellt als ^Z).

2. ASM

Syntax:

```
ASM DATEI
ASM DATEI. ABC     Dateityp: ASM
```

Zweck:

ASM ist der 8080-Assembler, der auf der angegebenen Quelldatei vom Dateityp ASM zwei neue Dateien produziert. Die eine Datei enthält den Objektcode im Hexadezimalformat und die andere Datei ein Assemblerlisting. Das Zeichen A der Kommandospezifikation indiziert das Quell-Laufwerk, B indiziert das Ziellaufwerk für die HEX-Datei, und C indiziert das Ziellaufwerk für die PRN-Datei. Wenn Sie ein Z in die Position von B oder C setzen, dann wird die entsprechende HEX- oder PRN-Datei unterdrückt. Ein X in der Position des Zeichens C sendet die Druckdatei auf die Konsole.

Fehlermeldungen:

D - Data Error:	Das Datastatement kann die gewünschten Datenelemente nicht in dem vorgesehenen Bereich speichern.
E - Expression Error:	ein Ausdruck konnte nicht ausgewertet werden.
L - Label Error:	eine Marke darf in diesem Kontext nicht auftreten.
N -	nicht implementiert
O - overflow:	ein Ausdruck ist zu kompliziert
P - Phase Error:	der Wert einer Marke wurde während des Assemblerlaufs umdefiniert.
R - Register Error:	der Wert, der für ein Register spezifiziert worden ist, ist nicht mit dem Code verträglich.
S - Syntax Error:	fehlerhafte Anweisung oder Ausdruck
U - Undefined Label:	eine Marke wurde nicht definiert
V - Value Error:	ein Operand ist nicht richtig spezifiert worden

Beispiele:

```
ASM Druck
ASM B: Prog. ABX
ASM B: Prog. BZZ
```

3. DDT

Syntax:

```
DDT Datei
```

Zweck:

DDT ist eine Debughilfe für 8080-Programme. Wenn Sie DDT ohne eine Dateiangabe aufrufen, wird DDT in die TPA geladen und erwartet eine Dateiangabe. Ein DDT-Kommandobuchstabe kann von einem oder mehreren Argumenten gefolgt werden, die entweder hexadezimal, Dateispezifikationen oder anderen Informationen abhängig vom Kommando sein können. Argumente werden durch Komma oder Leerzeichen voneinander getrennt. Zwischen Kommandobuchstabe und dem ersten Argument ist kein Leerzeichen erlaubt. Optionale Teile des DDT-Kommando sind in Klammern eingeschlossen.

As	Eingabe von Anweisungen in Assemblersprache.
D(S(,f))	Speicher in hexadezimal und in ASCI Anzeigen.
Fs,f,bc	Füllen eines Speicherbereichs mit einem bestimmten Bytewert bc.
G(s)(,bl(,b2))	Beginn der Programmausführung mit Angabe von Breakpoints.
Hc1,c2	Bilden der Hexadezimalsumme und -differenz.
Idatei	Setzen eines Dateinamens für das Einlesen einer Datei.
L(s(,f))	Auflisten des Speicherinhalts in disassemblierter Form.
Ms,f,d	Verschieben eines Speicherblocks
R(o)	Lesen einer Datei in den Speicher
Ss	Setzen oder Ändern eines Speicherwertes

T(n)	Verfolgen der Programmausführung (Trace)
U(n)	Überwachen der Ausführung ohne Trace
X(r)	Spezifizieren und Änderung der Prozessorregister

Beispiel:

```
DDT
DDT. PROG.COM
```

4. DIR

Syntax:

```
DIR d:Datei.Typ
```

Zweck:

DIR gibt die Namen der Dateien auf dem spezifizierten Diskettenlaufwerk aus. DIR zeigt keine Dateien, die mit dem SYS-Attribut versehen wurden.

Beispiele:

```
DIR
DIR B:
DIR B: Datei.Tex
DIR A*.ASM
DIR PROG???.PRN
DIR PROG.*
```

5. ED

Syntax:

```
ED Datei
```

Zweck:

ED ist der Zeileneditor des CP/M-Systems. Mit den ED-Kommandos können Sie Dateien erstellen und ändern.

Beispiel:

. ED PROG.DAT

ED Steuerzeichen:

Control/L	logisches Zeilenende in Zeichenketten
Control/X	Zeile löschen
Control/Z	Zeichenketten-Trennzeichen bzw. Abschluß von Zeichenketten
RUBOUT	Löschen eines Zeichens
Control/C	Stoppfunktion

ED Kommandos:

nA	Anhängen von n Zeilen der Originaldatei an den Puffer im Speicher
OA	Anhängen von Zeilen aus der Originaldatei, bis der Puffer halbvoll ist
#A	Anhängen von Zeilen aus der Originaldatei an den Puffer, bis er voll ist (oder die Datei zu Ende)
B, -B	Bewegen des Zeilenzeigers an den Anfang (B) oder das Ende (-B) des Puffers
nC, -nC	Bewegen des Zeigers n Zeichen vorwärts (C) oder rückwärts (-C) im Puffer
nD, -nD	Löschen von n Zeichen vor (-D) oder hinter (D) im Zeiger
E	Retten der neuen Datei und Rückkehr zum CP/M
Fstring (^Z)	Suchen nach Zeichenkette im Text
H	Retten der neuen Datei, dann Beginn des Edieren der neuen Datei von vorne
I	Beginn des Eingabemodus, der durch Tippen von Control/Z beendet wird
Istring (^Z)	Einfügen einer Zeichenkette an der Zeigerposition

JSuchstring^ZEinfügunsstring^ZLöschstring^Z	Juxtaposition von Zeichenketten
nK, -nK	Löschen von n Zeilen ab der Zeigerposition
nL, -nL, OL	Bewegen des Zeigers um n Zeilen
nMkommandofolge	Ausführen der Kommandofolge n mal
n, -n	Bewegen des Zeiger um n Zeilen und Ausgeben der Zeile, auf die der Zeiger nun zeigt
n:	geh zu Zeile n
:nkommando	Ausführen des Kommandos bis Zeile n
Nstring (^Z)	erweiterte Zeichenkettesuche
O	Rückkehr zur Originaldatei
nP, -nP	Bewegen des Zeigers 23 Zeilen nach vorne oder rückwärts. Ausgeben der 23 Zeilen auf der Console
Q	Löschen der neu erstellten Datei, Rückkehr zum CP/M
R	Lesen einer Datei X$$$$$$$.LIB in den Puffer
R Datei	Lesen einer Datei .LIB in den Puffer
Slöschstring^ZEinfügestring^Z	Ersetzen einer Zeichenkette
nT, -nT, OT	Ausgeben von n Zeilen
U, -U	Umsetzen in Großschreibung
V, -V	Ein- bzw. Ausschalten der Zeilennumerierung
OV	Ausgabe des freien Pufferspeichers
nW	Ausgeben n Zeilen auf die neue Datei
nX	Ausgeben von n Zeilen auf eine temporäre LIB Datei
OX	Löschen der Datei X$$$$$$$.LIB
nZ	Warten n Sekunden

6. ERA

Syntax:

ERA Datei

Zweck:

ERA löscht eine angegebene Datei oder Dateigruppe. Dieses Kommando akzeptiert die "Wildcard"-Zeichen.

Beispiel:

```
ERA Programm.BAS
ERA B: Prog.DAT
ERA A: Brief.*
ERA*.BAK
ERA  :*.*
```

7. MOVCPM

Syntax:

```
MOVCPM *
MOVCPM nn
MOVCPM nn *
MOVCPM * *
```

Zweck:

MOVCPM konfiguriert eine nnKbyte-Version von CP/M. Wenn ein Stern an Stelle von nn spezifiziert wird, berechnet MOVCPM die geeignete Speichergröße. Wenn Sie den zweiten Stern spezifizieren, läßt CP/M die neue Version im Speicher für eine SYSGEN- oder SAVE-Operation. SYSGEN schreibt eine Kopie des verschobenen Systems auf die Systemspuren der Zieldiskette. SAVE kreiert eine Datei mit dem verschobenen System und schreibt sie auf die spezifizierte Diskette. Wenn Sie den zweiten Stern nicht spezifizieren, verschiebt MOVCPM die neue Systemversion an den gewünschten Speicherplatz und startet das System dort. Beachten Sie bitte, daß in den meisten Fällen der zweite Stern notwendig ist.

Beispiel:

```
MOVCPM 48 *
MOVCPM * *
```

8. PIP

Syntax:

```
PIP Zieldatei = Quelldatei (Option), ...
PIP Zieldatei=Gerät
PIP Gerät = Quelldatei (Option)
PIP Gerät = Gerät (Option)
```

Zweck:

PIP kopiert Dateien, kombiniert Dateien und transferiert Dateiinhalte von oder zu Peripheriegeräten. Die erste Dateiangabe ist immer die Zieldatei; die zweite Dateiangabe ist immer die Quelldatei. Quelle oder Ziel kann jedes logische CP/M Gerät sein. Wird CP/M ohne Angabe von Parametern gestartet, meldet sich das Programm mit einem Sternchen und erwartet eine Serie von PIP-Kommandos, die Zeile für Zeile eingegeben und bearbeitet werden. Quelldateiangaben mit Optionen können, getrennt durch Komma, angegeben werden, um eine oder mehrere Dateien zu einer neuen Datei zu kombinieren.

PIP-Optionen:

A	Kopiere nur modifizierte Dateien
Dn	Lösche alle Zeichen nach der Spalte n beim Kopieren
E	Die Übertragung zur Konsole erfolgt mit Echo
F	Entferne Seitenvorschubzeichen (form-feed) aus der Quelldatei
Gn	Hole eine Datei von User-Nummer n
H	Teste, ob die Datei korrektes Hexadezimalformat enthält
I	Ignoriere die :00-Records einer Hex-Datei und teste im übrigen auf korrektes Hexadezimalformat
L	Übersetze Großbuchstaben in Kleinbuchstaben
N	Numeriere die Ausgabenzeilen beginnend bei 1 in einer Schrittweite von 1
O	Übertrage Objektcodedatei, ignoriere Control/Z
Pn	Setze Seitenlänge auf n (Voreinstellung: 60)
QText^Z	Beende das Kopieren aus der Quelldatei, wenn die Zeichenkette "Text" angetroffen wurde
R	Lies Dateien, die mit dem Systemattribut versehen wurden
Stext^Z	Beginne das Kopieren ab der Stelle, an der "text" angetroffen wurde
Tn	Expandiere Tabulatoren zu n Leerzeichen
U	Konvertiere Kleinbuchstaben in Großbuchstaben

V	Verifiziere, daß die Daten korrekt übertragen wurden
W	Überschreibe Dateien, die mit Schreibschutz versehen worden sind
Z	Lösche das Paritätsbit

Beispiele:

```
Kopieren von einer Diskette auf die andere:
PIP B: = A:Datei
PIP B:Datei = A:
Kopieren einer Datei und Umbenennen
PIP B:Neu = A:Alt
PIP Neu = Alt
Kopieren mehrerer Dateien:
PIP
B: = *.COM  [AV]
B: = C:Datei.* [RW]
PIP B: = C:*.*

Kombinieren von Dateien:
PIP B: Neu = Alt 1, Alt 2, Alt 3,

Kopieren, Umbenennen und Holen von User 1:
PIP Neu = Alt  [G1]

Kopieren von oder zu Peripheriegeräten:
PIP B: Datei = Con:
PIP LST: = CON:
PIP LST: = B:Text [T8]
PIP PRN: = B:Datei. TEX

Logische Geräte von PIP:
CON Console
LST Listgerät
PUN Lochstreifenstanzer
RDR Lochstreifenleser
INP Spezialeingabegerät
OUT Spezialausgabegerät
PRN wie LST, jedoch Tabulator in jeder 8. Spalte,
    Seitenvorschübe alle 60 Zeilen, Numerieren der
    Zeilen
EOF generiert ein Control /Z (End-of-File)
NUL generiert 40 Nullbytes für das PUN-Gerät
```

PIP physikalische Geräte:
TTY Konsole, Terminal, Leser, Stanzer, Teletype
CRT Konsole, Terminal, Bildschirmgerät
PTR Lochstreifenleser, Kartenleser
PTP Lochstreifenstanzer, Kartenstanzer
LPT Drucker
UC1 Benutzer-definierter Konsole
UR1 Benutzer-definierter Leser
UR2 Benutzer-definierter Leser
UP1 Benutzer-definierter Stanzer
UP2 Benutzer-definierter Stanzer
UL1 Benutzer-definiertes Druckgerät

9. REN

Syntax:

REN Neue_Datei = Alte_Datei

Zweck:

REN ändert den Namen einer existierenden Datei. Der Name "Alte Datei" wird geändert zu "Neue Datei". Sie können keine Dateien auf verschiedenen Laufwerken definieren.

Beispiele:

```
REN Neu = Alt
REN B: Neu = Alt
REN Neu = B: Alt
```

10. SAVE

Syntax:

SAVE n Datei

Zweck:

SAVE liest n Blöcke zu 256 Bytes aus der TPA und speichert sie auf der angegebenen Datei. Benutzen Sie den DDT, um die Größe Ihres Programms festzustellen.

Beispiel:

```
SAVE 3 X.COM
SAVE 40 Q
SAVE 4 X.Y
SAVE 10 B:ZOT.COM
```

11. STAT

Syntax:

```
STAT d:
STAT Datei
STAT d:DSK:
STAT DEV:
STAT VAL:
STAT USR:
STAT d:=R/O
STAT Datei $R/O
STAT Datei $R/W
STAT Datei $SYS
STAT Datei $DIR
STAT Datei $S
```

Zweck:

STAT gibt Ihnen Informationen über Diskettenlaufwerke, Dateien und andere Peripheriegeräte, die mit dem Rechner verbunden sind. STAT ändert die Attribute von Laufwerken und Dateien. STAT ohne Parameterteil gibt Ihnen den freien Speicherplatz des augenblicklich selektierten Laufwerks in Kilobyte zurück uns sagt Ihnen, ob das Laufwerk schreibgeschützt ist (R/O) oder nicht (R/W). Benützen Sie STAT, um ein Laufwerk schreibzuschützen und verwenden Sie Control/C, um das Laufwerk freizugeben. STAT mit einer Dateiangabe liefert Ihnen die Anzahl der von dieser Datei belegten Kilobytes und zeigt Ihnen die Attribute dieser Datei an. Mit STAT kann eine Datei schreibgeschützt werden, der Schreibschutz wieder aufgehoben werden, die Datei mit einem Systemattribut versehen werden oder dieses Systemattribut wieder aufgehoben werden. Dateien mit Systemattribut werden von STAT in Klammern aufgelistet. STAT DSK: zeigt Ihnen auf dem Bildschirm die Charakteristik des spezifizierten Laufwerks. STAT USR: zeigt Ihnen an, welche Benutzernummern auf dem aktuellen Laufwerk belegt sind. STAT VAL: zeigt die möglichen STAT Kommandos an. STAT akzeptiert "Wildcard"-Zeichen im Parameterteil des Kommandos.

Beispiele:

```
STAT
STAT B: R/O
STAT Datei
STAT C: Datei
STAT Datei $R/O
STAT *.com $R/O
STAT *.bak
STAT B:*.*
```

12. SUBMIT

Syntax:

SUBMIT Datei Parameter Dateityp: SUB

Zweck:

SUBMIT startet die Ausführung einer Datei, die Kommandos für CP/M enthält. Die Datei darf je Zeile ein Kommando enthalten. Die SUBMIT-Datei muß vom Dateityp SUB sein. Jeder (optionale) Parameter, der auf die Dateispezifikation folgt, wie zum Beispiel Laufwerks- oder Dateiangaben, ersetzt einen korrespondierenden, formalen Parameter ($1, $2, $3...) in der SUBMIT-Datei.

Beispiel:

```
SUBMIT START
SUBMIT B: START
SUBMIT START C: Brief
```

13. SYSGEN

Syntax:

SYSGEN d: Datei

Zweck:

SYSGEN kopiert das Betriebssystem CP/M von den Systemspuren auf der Quelldiskette zu einer Zieldiskette. Mit einem SYSGEN-Aufruf können mehrere CP/M-Disketten generiert werden. SYSGEN mit einer Dateispezifikation fragt nicht nach einer Quelldatei an, sondern benutzt die spezifizierte Datei als Quelldatei.

Beispiel:

```
A>SYSGEN

SYSGEN VER 2.0
SOURCE DRIVE NAME (OR RETURN TO SKIP)A
SOURCE ON A, THEN TYPE RETURN <cr>
FUNCTION COMPLETE
DESTINATION DRIVE NAME (OR RETURN TO REBOOT)B
DESTINATION ON B, THE TYPE RETURN <cr>
FUNCTION COMPLETE
DESTINATION DRIVE NAME (OR RETURN TO REBOOT) <cr>

A> SYSGEN CPM32.COM

DESTINATION DRIVE NAME (OR RETURN TO REBOOT)A
DESTINATION ON A; THEN TYPE RETURN <cr>
FUNCTION COMPLETE
. . .
```

14. TYPE

Syntax:

```
TYPE Datei
```

Zweck:

TYPE gibt den Inhalt einer ASCII-Datei auf den Bildschirm aus. Um die Ausgabe abzubrechen, drücken Sie irgendeine Taste. TYPE akzeptiert keine "Wildcard"-Dateispezifikation. Wird vor dem Kommando TYPE die Taste Control/P gedrückt, wird die Ausgabe auf der Konsole gleichzeitig auf den Drucker mitprotokolliert, bis die Taste Control/P wieder gedrückt wird.

Beispiel:

```
TYPE Brief
TYPE a: Dokument.3
TYPE Programm.bas
TYPE Prog.asm
```

15. USER

Syntax:

USER n (0 ... 15)

Zweck:

Anzeigen und Wechsel des augenblicklichen Benutzerbereichs. USER ohne Parameterteil gibt die augenblickliche Benutzernummer aus. USER mit einer Nummer zwischen 0 und 15 wechselt den augenblicklichen Benutzerbereich zu dem neuen, angegebenen Benutzerbereich. CP/M nimmt einen voreingestellten Benutzerbereich von 0 an.

Beispiel:

```
USER
USER 2
USER 7
```

16. XSUB

Syntax:

XSUB

Zweck:

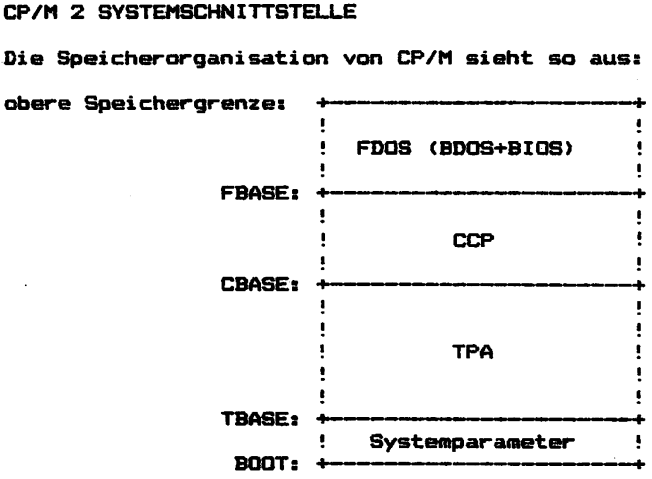
XSUB erlaubt es Programmen, die eine gepufferte Eingabe verwenden, wie zum Beispiel ED, in SUBMIT-Dateien abzulaufen. Das XSUB Kommando erscheint als erstes Kommando in der SUBMIT-Datei.

Beispiel:

```
A>TYPE SAVER.SUB
  XSUB
  DDT
  I$1.HEX
  R
  GO
  SAVE 1 $2.COM
```

CP/M 2 SYSTEMSCHNITTSTELLE

Die Speicherorganisation von CP/M sieht so aus:



Die exakten Adressen der Speicherstellen BOOT, TBASE, CBASE und FBASE variieren von Rechnerversion zu Rechnerversion.
Jedoch wird bei allen Standardversionen von CP/M die Adresse BOOT=0000H verwendet, die die unterste RAM-Adresse ist.Der Maschinencode an der Speicherstelle BOOT führt einen Warmstart durch, bei dem die Programmteile und Variablen des CCP initialisiert werden. Daher müssen Programme, die auf die Kommandoebene des CP/M zurückkehren wollen, nur zur Adresse BOOT springen. Des weiteren setzen die Standardversionen von CP/M für TBASE=BOOT+100H voraus, was normalerweise die Speicherstelle 100H ist. Der Haupteintrittspunkt des FDOS ist BOOT+0005H (normalerweise 0005H), wo sich ein Sprung zum FDOS befindet. Die Adresse in BOOT+0006H (normalerweise 0006H) enthält den Wert von FBASE und er kann verwendet werden um die Obergrenze des freien Speichers festzustellen, wenn man voraussetzt, dass der CCP überschrieben werden kann.
Programme ("transient Programs") werden in die TPA geladen und ausgeführt. Der Bediener kommuniziert mit dem CCP durch Eingabe von Kommandozeilen nach jedem Prompt. Jede Kommandozeile hat eine der folgenden Erscheinungsweisen:

Kommando

Kommando Datei1

Kommando Datei1 Datei2

wobei "Kommando" entweder eine der eingebauten Funktionen wie DIR oder TYPE oder der Name eines Programms ist (DDT, ED, PIP, ...). Wenn das Kommando eine der eingebauten Funktionen von CP/M ist, wird es sofort ausgeführt. Im anderen Fall durchsucht der CCP die augenblicklich angewählte Diskette nach einer Datei namens

Kommando.COM

Ist die Datei vorhanden, nimmt der CCP an, dass es sich um das Speicherabbild eines ausführbaren Programms handelt, das bei Adresse TBASE beginnt. Der CCP läd diese COM-Datei von der Diskette in den Speicher bei TBASE beginnend. Das Programm darf

sich bis CBASE erstrecken.
Falls das Kommando von einer oder zwei Dateispezifikationen gefolgt wird, bereitet der CCP ein oder zwei Dateikontrollblocknamen (file control block - FCB) im Systemparameterbereich vor. Diese optionalen FCBs sind in der Form, die es gestattet, die Dateien über das FDOS anzusprechen und sind im folgenden Abschnitt beschrieben.
Das Programm wird dann vom CCP angesprungen und beginnt mit der Ausführung, wobei es die E/A des FDOS verwendet. as Programm wird vom CCP "aufgerufen". Daher kann nach Ablauf einfach zum CCP zurückspringen oder zu BOOT springen um zum CCP zurückzukehren. Im ersten Fall darf das Programm den Speicherbereich oberhalb von CBASE nicht verändern, wogegen im zweiten Fall der Speicher bis FBASE-1 benutzt werden kann. Das aufgerufene Programm kann die E/A von CP/M einschliesslich des Diskettensystems verwenden um mit Benutzerperipherie oder Hintergrundspeichern Daten auszutauschen. Die E/A-Module werden angesprochen, indem eine Funktionsnummer und eine Adresse für Informationen durch den FDOS Eintrittspunkt bei BOOT+0005H übergeben wird. Soll zum Beispiel von Diskette gelesen werden, sendet das Programm die Nummer für "Diskette lesen" und die FCB-Adresse an das CP/M FDOS. Das FDOS führt seinerseits die Funktion aus und kehrt dann entweder mit der Meldung "Lesen fertig" oder mit einer Fehlernummer, die den erfolglosen Leseversuch anzeigt, zum aufrufenden Programm zurück.

Konventionen für den Systemaufruf

CP/M- Funktionen, die für den Zugriff durch Benutzerprogramme bereitstehen lassen sich in zwei Gruppen einteilen: einfache Geräte-E/A und Disketten-E/A. Die Gerätetreiber beinhalten:

Lesen Zeichen von Konsole
Schreiben Zeichen nach Konsole
Lesen Zeichen sequentiell von Band
Schreiben Zeichen sequentiell auf Band
Schreiben Zeichen nach List-Device
Holen oder Setzen E/A Status
Ausdrucken Konsolenpuffer
Lesen Konsolenpuffer
Abfragen, ob Konsole bereit

Die FDOS-Operationen, die Disketten-E/A betreiben, sind:

Rücksetzen Diskettensystem
Auswählen Laufwerk
Datei kreieren
Datei eröffnen
Datei schliessen
Durchsuchen Directory (DAteiverzeichnis)
Löschen Datei
Umbenennen Datei
Wahlfei oder sequentiell Lesen
Wahlfrei oder sequentiell Schreiben
Abfragen zugreifbarer Diskettenlaufwerke
Abfragen des selektierten Laufwerks
Setzen der DMA-Adresse
Setzen/Rücksetzen der Dateiattribute

Wie schon erwähnt, wird der Zugriff zu einer FDOS-Operation durch übergabe einer Funktionsnummer und Ansprung des Eintrittspunkts

bei BOOT+0005H erreicht. Allgemein wird die Funktionsnummer in Register C und die Informationsadresse im Registerpaar DE übergeben. Einzelbytes werden in Register A, 16-bit-Werte im Registerpaar HL zurückgeliefert (Null wird zurückgegeben, wenn die Funktionsnummer nicht im erlaubten Bereich liegt). Aus Gründen der Kompatibilität ist bei Rückkehr in allen Fällen das Register A=H und Register B=L. Diese Registerübergabekonventionen entsprechen jenen von Intels Systemimplementierungssprache PL/M. Die CP/M Funktionen und ihre Nummern sind im Folgenden aufgelistet:

0	System Rücksetzen	19	Löschen Datei
1	Konsoleingabe	20	Lesen sequentiell
2	Konsolausgabe	21	Schreiben sequentiell
3	Reader Eingabe	22	Datei erzeugen
4	Punch Ausgabe	23	Datei umbenennen
5	List Ausgabe	24	Login Vektor zurückgeben
6	Direkte Konsol-E/A	25	Laufwerknummer geben
7	Holen I/O-Byte	26	Setzen DMA-Adresse
8	Setzen I/O-Byte	27	Adr. der Belegungstabelle
9	Drucke String	28	Schreibschutz setzen
10	Lies Konsolpuffer	29	Holen R/O Vektor
11	Holen Konsolstatus	30	Setzen DAteiattribute
12	Holen Versionsnummer	31	Holen Adr.(Disk.Param.)
13	Disksystem rücksetzen	32	Holen/Setzen User Code
14	Laufwerk wählen	33	Wahlfrei lesen
15	Datei eröffnen	34	Wahlfrei schreiben
16	Datei schliessen	35	Dateigrösse berechnen
17	Durchsuchen DIR nach 1.	36	Setzen wahlfreien Record
18	Durchsuchen DIR nach nächster	37	Zurücksetzen Laufwerk
		40	Wahlfrei schreiben und mit Nullen auffüllen

(Die Funktionen 28 und 32 sollten in Anwendungsprogrammen vermieden werden um die Aufwärtskompatibilität mit CP/M sicherzustellen.)
Beim Eintritt in ein Programm setzt der CCP den Stackpointer auf einen Stackbereich mit 8 Ebenen und der CCP-Rückkehradresse auf der obersten Position, sodass noch 7 Ebenen zur Verfügung stehen, bis ein Überlauf auftritt. Obwohl dieser Stack normalerweise nicht von Benutzerprogrammen verwendet wird (die meisten Programme benutzen den Sprung über die Adresse 0000H), ist er doch gross genug für CP/M-Systemaufrufe, da das FDOS einen eigenen Stackbereich verwendet. Das folgende Assembler-Programmstück liest beispielsweise fortlaufend einzelne Zeichen, bis ein Sternchen auftaucht und gibt dann die Kontrolle an den CCP zurück (es wird ein Standard-CP/M mit BOOT=0000H angenommen).

```
BDOS    EQU     0005h       ;STANDARD CP/M EINTRITT
CONIN   EQU       1         ;KONSOLEINGABEFUNKTION
;
        ORG     0100h       ;TPA-BEGINN
NEXTC:  MVI     C,CONIN     ;LIES NäCHSTES ZEICHEN
        CALL    BDOS        ;ZEICHEN IN A
        CPI     '*'         ;ENDE DER BEARBEITUNG ?
        JNZ     NEXTC       ;WEITER FALLS NICHT
        RET                 ;ZURüCK ZUM CCP
        END
```

Jede einzelne Datei kann eine beliebige Zahl von Blöcken enthalten, von einer leeren Datei bis zur vollen Diskettenkapazität. Die einzelnen Laufwerke werden logisch voneinander unterschieden, jedes hat sein eigenes Directory (Dateiverzeichnis) und seine eigenen DAteien. Der Name einer Diskettendatei besteht aus drei Teilen: dem Laufwerknamen (ein Buchstabe), dem Dateinamen, der aus bis zu 8 Zeichen ungleich Blank besteht und dem Dateityp (auch Extension genannt) der aus bis zu 3 Zeichen bestehen kann und auch fehlen darf. Der Dateityp sagt etwas über die Herkunft der Datei aus, wogegen der DAteiname zur Unterscheidung der einzelnen Dateien auf der Diskette dient. Unten werden einige Dateikathegorien aufgeführt, die sich eingebürgert haben, die Angaben sind aber nicht bindend.

ASM	Assemblerquelldatei	PLI	PL/1 Quelldatei
PRN	Listing für Drucker	REL	Relozierbares Modul
HEX	Hexadez. Maschinencode	TEX	TEX Formatierer Quelle
BAS	Basicquelle	BAK	ED Sicherheitskopie
INT	Zwischencode	SYM	SID Symboldatei
COM	Kommandodatei	PAS	Pascalquelle
$$$	Temporäre Datei		

Quelldateien werden als Folge von ASCII-Zeichen betrachtet, wobei jede Zeile von Wagenrücklauf (Carriage Return (CR=0DH)) und Zeilenvorschub (Line Feed (LF=0AH)) abgeschlossen wird. So kann ein CP/M Record von 128 Byte mehrere Zeilen Quelltext enthalten. Das Ende einer ASCII-Datei wird entweder durch das ASCII-Zeichen SUB (1AH = Control-Z) oder durch ein echtes end-of-file, das von der CP/M Leseoperation gemeldet wird, markiert. In Dateien, die Maschinencode enthalten wird das Zeichen SUB jedoch ignoriert, hier wird nur das end-of-file der Leseoperation verwendet.
Dateien bei CP/M kann man sich als eine Folge von bis zu 65536 Records zu je 128 Byte vorstellen, die von 0 bis 65535 durchnummeriert sind. So sind Dateien bis zu 8 Megabyte möglich.Der Benutzer sollte jedoch beachten, das die Records physikalisch nicht in der Reihenfolge auf der Diskette gespeichert werden müssen, in der sie logisch aufeinanderfolgen. Intern werden die Dateien in Abschnitte zu 16 KByte unterteilt, die "logische Extensionen" genannt werden. So können die Zähler einfach als 8-bit-Grössen behandelt werden. Die Aufteilung in Extensionen wird in den folgenden Paragraphen behandelt; sie sind jedoch nicht wirklich wichtig für den Programmierer, da auf jede Extension automatisch zugegriffen wird(im sequentiellen wie im wahlfreien Zugriff).
In den Dateioperationen, die mit Funktionsnummer 15 beginnen, wird über DE normalerweise ein Dateikontrollblock (FCB) adressiert. Programme verwenden oft den Speicherbereich bei BOOT+005CH (standardmässig 005CH), der normalerweise von CP/M für die FCBs verwendet wird, für einfache Dateioperationen. Die Grundeinheit der Dateiinformation ist ein Block (Record) zu 128 Byte, der für alle Dateioperationen verwendet wird. So wird ab der Speicherstelle BOOT+0080H (normalerweise 0080H) ein Standard-Speicherbereich für die Disketten-E/A bereitgestell, der auch der voreingestellte DMA-Bereich ist (Funktion 26). Alle Operationen, die das Directory betreffen finden in einem reservierten Bereich statt und beeinflussen nicht wie in Version 1 die Schreib-Puffer. Eine Ausnahme bilden hier nur die Funktionen 17 und 18

(Durchsuchen DIR) aus Gründen der Kompatibilität.
Der FCB-Bereich besteht aus einer Folge von 33 Bytes für sequentiellen Zugriff und 36 Byte für wahlfreien Zugriff. Der FCB auf der voreingestellten Adresse BOOT+005CH kann auch für wahlfreien Zugriff verwendet werden, da beginnend bei BOOT+007DH drei Bytes frei sind. Das Format eines FCB enthält die folgenden Felder:

```
+--+--+--+---+--+--+--+--+--+--+--+---+--+--+--+--+--+---
!dr!f1!f2!...!f8!t1!t2!t3!ex!s1!s2!rc!d0!...!dn!cr!r0!r1!r2!
+--+--+--+---+--+--+--+--+--+--+--+---+--+--+--+--+--+---
 00 01 02 ... 08 09 10 11 12 13 14 15 16 ... 31 32 33 34 35
```

Die Bedeutung der Felder ist folgende:

dr — Laufwerknummer (0-16)
0 => verwende voreingestelltes Laufwerk
1 => Laufwerk A
2 => Laufwerk B
.
.
16 => Laufwerk P

f1...f8 — Dateiname in Grossbuchstaben (bit 8 = 0)

t1...t3 — Dateityp in Grossbuchstaben (bit 8 = 0)
t1',t2' und t3' bezeichnen die höchstwertigen bits dieser Zeichen mit der Bedeutung:
t1' = 1 => Datei schreibgeschützt
t2' = 1 => SYS-Datei, wird nicht bei DIR gelistet

ex — augenblickliche Extensionsnummer, wird vom Benutzer normalerweise auf 00 gesetzt, liegt während einer E/A-Operation zwischen 0 und 31.

s1 — reserviert für System

s2 — reserviert für System (0 für Funktionen 15,17,18,22)

rc — Blockzähler für Extension 'ex', Werte von 0 bis 127

d0...dn — Wird von CP/M gefüllt, reserviert für System

cr — Blocknummer des gerade gelesenen oder geschriebenen Blocks, wird vom Benutzer auf 0 gesetzt.

r0..r2 — optionale Blocknummer im Bereich von 0-65535 für wahlfreie E/A mit überlauf nach r2. r0 und r1 bilden einen 16-bit-Wert mit dem höchstwertigen Byte in r1.

Jede Datei,auf die durch CP/M zugegriffen wird muss einen korrespondierenden FCB in Speicher besitzen, der Namen und Zugriffsinformationen für alle Dateioperationen hält. Beim Dateizugriff ist der Benutzer dafür verantwortlich, dass die unteren 16 Bytes gefüllt und das Feld cr gesetzt werden. Normalerweise werden die Bytes 1 bis 11 mit den ASCII-Werten des Dateinamens und des Dateityps gefüllt, während die anderen Felder mit Null vorbesetzt werden.
FCBs werden in einem Directory-Bereich auf der Diskette gespeichert und in den Speicher übertragen bevor der Benutzer Dateioperationen aufruft (siehe Funktionen "Datei erzeugen" und

"Datei eröffnen"). Die Speicherkopie des FCB wird von den Dateioperationen immer auf den aktuellen Stand gebracht und beim schliessen der Datei auf die Diskette zurückgeschrieben.
Der CCP konstruiert die ersten 16 Bytes zweier optionaler FCBs für ein Programm indem er den Rest der Kommandozeile nach dem Kommandonamen nach Dateinamen durchsucht und nicht spezifizierte Angaben durch Leerzeichen ersetzt (Datei1 und Datei2 in der Kommandozeile weiter vorne). Der erste FCB wird bei Adresse BOOT+005CH konstruiert und kann für fortlaufende Dateioperationen verwendet weden. Der zweite FCB belegt die Felder d0..dn des ersten FCB und muss vor Gebrauch in enen anderen Speicherbereich übertragen werden. Wenn beispielsweise der Bediener eingibt:

PROGRAMM B:X.ABC Y.DEF

wird die Datei PROGRAMM.COM in die TPA geladen und der erste FCB bei BOOT+005CH wird für Laufwerk 2, Dateiname X, Dateityp ABC initialisiert. Die zweite Laufwerknummer wird auf 0 gesetzt und bei BOOT+006CH eingetragen, der Dateiname Y bei BOOT+006DH, der Dateityp DEF bei BOOT+0075H, also 8 Bytes später, eingetragen. Alle übrigen Felder werden auf 0 gesetzt. Es liegt also in der Verantwortung des Benutzers die Angaben über die zweite Datei in einen anderen Speicherbereich, gewöhnlich einen anderen FCB zu verschieben, bevor er die Datei eröffnet, deren FCB bei BOOT+005CH beginnt, denn die Eröffnungsoperation überschreibt die zweite Dateiangabe.
Werden beim Kommandoaufruf keine Dateinamen angegeben, so werden die Felder bei BOOT+005CH und BOOT+006CH mit Leerzeichen gefüllt. In allen Fällen übersetzt der CCP Kleinbuchstaben in Grossbuchstaben um den Namenskonventionen von CP/M zu genügen.
Als zusätzliche Annehmlichkeit wird der Standard-Pufferbereich bei BOOT+0080H mit dem Rest der Kommandozeile (nach dem Kommandonamen) gef§llt, wobei die erste Speicherposition die Anzahl der folgenden Zeichen enthält, und danach dei Zeichen der Zeile folgen. Bei der oben angegebenen Kommandozeile wird der Speicher ab BOOT+0080H folgendermassen initialisiert:

BOOT+0080H:

+00	+01	+02	+03	+04	+05	+06	+07	+08	+09	+0A	+0B	+0C	+0D	+0E
E	' '	'B'	':'	'X'	'.'	'A'	'B'	'C'	' '	'Y'	'.'	'D'	'E'	'F'

Dabei sind die Zeichen zu Grossbuchstaben konvertiert worden. Der Speicher hinter dem Letzten Zeichen wird nicht verändert. Es liegt wieder in der Verantwortung des Programmierers, die Information aus diesem Pufferbereich zu holen, bevor die erste Dateioperation stattfindet, wen nicht die Adresse des DMA-Puffers explizit geändert wird.
Auf den folgenden Seiten werden nun die Funktionen im Einzelnen erklärt:

Funktion 0: System zurücksetzen

Eintrittsparameter:
Register C: 00H

Die Funktion "System zurücksetzen" hat die Rückkehr zum CP/M auf Kommandoebene (CCP) zur Folge. Der CCP initialisiert das Diskettensystem und stellt das Laufwerk A ein. Die Funktion hat genau die gleiche Wirkung wie ein Sprung nach BOOT.

```
+-----------------------------------------------+
!                                               !
! Funktion  1: Konsoleingabe                    !
!                                               !
! Eintrittsparameter:                           !
!    Register  C: 01H                           !
!                                               !
! Resultat:                                     !
!    Register  A: ASCII-Zeichen                 !
!                                               !
+-----------------------------------------------+
```

Diese Funktion liest das nächste Zeichen von der Konsole in das Register A ein. Darstellbare Zeichen sowie Wagenrücklauf, Zeilenvorschub, und Rückwärtsschritt (Control/H) werden an die Konsole zurückgeschickt (Echo). Tabulatorzeichen (Control/I) bewegen den Cursor auf die nächste Tabulatorposition.Das Anhalten des Scolling (Control/S) wird abgeprüft ebenso das Ein- und Ausschalten des Protokolldrucks (Control/P). Das FDOS kehrt erst zum aufrufenden Programm zurück, wenn ein Zeichen eingetippt wurde. Der Programmablauf wird also angehalten, bis ein Zeichen zur Verfügung steht.

```
+-----------------------------------------------+
!                                               !
! Funktion  2: Konsolausgabe                    !
!                                               !
! Eintrittsparameter:                           !
!    Register  C: 02H                           !
!    Register  E: ASCII-Zeichen                 !
!                                               !
+-----------------------------------------------+
```

Das ASCII-Zeichen in Register E wird zur Konsole gesandt. Genau wie bei Funktion 1 werden Tabs expandiert sowie Control/P und Control/S abgefragt.

```
+-----------------------------------------------+
!                                               !
! Funktion  3: Reader Eingabe                   !
!                                               !
! Eintrittsparameter:                           !
!    Register  C: 03H                           !
!                                               !
! Resultat:                                     !
!    Register  A: ASCII-Zeichen                 !
!                                               !
+-----------------------------------------------+
```

Diese Funktion liest das nächste Zeichen vom logischen Gerät "Reader" (Lochstreifenleser) in das Register A (siehe auch Definition des IOBYTE in Kapitel 6). Die Kontrolle wird erst an das aufrufende Programm zurückgegeben, wenn ein Zeichen gelesen wurde.

```
+-------------------------------------------------+
!                                                 !
! Funktion  4: Punch Ausgabe                      !
!                                                 !
! Eintrittsparameter:                             !
!    Register  C: 04H                             !
!    Register  E: ASCII-Zeichen                   !
!                                                 !
+-------------------------------------------------+
```

Diese Funktion sendet ein Zeichen in Register E zum logischen Gerät "Punch" (Lochstreifenstanzer).

```
+-----------------------------------------------+
!                                               !
! Funktion  5: List Ausgabe                     !
!                                               !
! Eintrittsparameter:                           !
!    Register  C: 05H                           !
!    Register  E: ASCII-Zeichen                 !
!                                               !
+-----------------------------------------------+
```

Diese Funktion sendet ein Zeichen in Register E zum logischen Protokoll-Ausgabegerät.

```
+-----------------------------------------------+
!                                               !
! Funktion  6: Direkte Konsol-E/A               !
!                                               !
! Eintrittsparameter:                           !
!    Register  C: 06H                           !
!    Register  E: 0FFH (Eingabe) oder           !
!                 ASCII-Zeichen (Ausgabe)       !
!                                               !
! Resultat:                                     !
!    Register  A: Zeichen oder Status           !
!                                               !
+-----------------------------------------------+
```

Direkte Konsol-E/A wird von CP/M bereitgestellt um besondere Anwendungen zu unterstützen, die eine Konsol-E/A auf tieferem Niveau benötigen. Im allgemeinen sollte die Verwendung dieser Funktion vermieden werden, da sie keine der CP/M Kontrollfunktionen wie Control/P oder Control/S berücksichtigt. Jedoch sollten Programme, die unter früheren Versionen von CP/M direkte Konsol-E/A durch das BIOS ausführten so geändert werden, dass sie nun diese BDOS-Funktion verwenden. Auf diese Weise wird eine Kompatibilität zu spateren Versionen von CP/M oder MP/M sichergestellt.
Beim Eintritt in Funktion 6 muss Register E entweder 0FFH , wodurch eine Eingabeanforderung signalisiert wird, oder ein ASCII-Zeichen für die Ausgabe enthalten. Ist der Eingabewert 0FFH,liefert die Funktion A = 00 zurück, wenn kein Zeichen eingegeben wurde. Im anderen Fall enthält Register A das eingegebene Zeichen. Falls das Eingaberegister einen Wert ungleich 0FFH enthält, wird das entsprechende ASCII-Zeichen zur Konsole geschickt.
Die Funktion 6 darf nicht zusammen mit den anderen Konsolfunktionen verwendet werden.

```
+-----------------------------------------------+
!                                               !
! Funktion  7: Holen I/O Byte                   !
!                                               !
! Eintrittsparameter:                           !
!    Register  C: 07H                           !
!                                               !
! Resultat:                                     !
!    Register  A: Wert des I/O Byte             !
!                                               !
+-----------------------------------------------+
```

Die Funktion 7 liefert den Wert des IOBYTE in Register A. Für nähere Erläuterungen dazu siehe Kapitel 6.

```
+-------------------------------------------------+
!                                                 !
! Funktion  8: Setzen I/O Byte                    !
!                                                 !
! Eintrittsparameter:                             !
!    Register  C: 08H                             !
!    Register  E: Wert de I/O Byte                !
!                                                 !
+-------------------------------------------------+
```

Diese Funktion ändert den Wert des IOBYTE auf jenen, der in Register E gegeben wurde.

```
+-------------------------------------------------+
!                                                 !
! Funktion  9:  Drucke Zeichenkette (String)      !
!                                                 !
! Eintrittsparameter:                             !
!    Register  C: 09H                             !
!    Register DE: Adresse eines Strings           !
!                                                 !
+-------------------------------------------------+
```

Diese Funktion sendet eine Zeichenkette, die im Speicher abgelegt ist an die Konsole. Die Adresse des ersten Zeichens steht in DE, das Ende der Zeichenkette wird durch das $-Zeichen signalisiert. Wie in Funktion 2 werden Tabulatorzeichen expandiert und die Kontrollzeichen Control/P und Control/S abgeprüft.

```
+-------------------------------------------------+
!                                                 !
! Funktion 10: Lesen Konsolpuffer                 !
!                                                 !
! Eintrittsparameter:                             !
!    Register  C: 0AH                             !
!    Register DE: Pufferadresse                   !
!                                                 !
! Resultat:                                       !
!    Zeichen von der Konsole im Puffer            !
!                                                 !
+-------------------------------------------------+
```

Diese Funktion liest eine Zeile von der Konsole, die bei der Eingabe korrigiert werden kann, und speichert sie in einem Puffer, der von DE adressiert wird. Die Konsoleingabe wird beendet, wenn entweder der Puffer überläuft oder eines der beiden Zeichen Wagenrücklauf(CR) und Zeilenvorschub(LF) eingegeben wurde. Der Puffer hat die Form:

```
DE: +0 +1 +2 +3 +4 +5 +6 +7 +8 ... +n
   +--+--+--+--+--+--+--+--+--+---+--+----------------
   !mx!nc!c1!c2!c3!c4!c5!c6!c7!...!??!
   +--+--+--+--+--+--+--+--+--+---+--+----------------
```

wobei mx die Maximalzahl der Zeichen ist, die der Puffer aufnehmen soll (1..255) und nc die Zahl der tatsächlich vom FDOS gelesenen Zeichen ist (wird vom FDOS gesetzt). FAlls nc<mx ist, werden die Speicherzellen, die auf das letzte Zeichen folgen (hier durch ?? markiert) unverändert. Werden der Eingabe der Zeile werden die folgenden Steuerzeichen akzeptiert:

Rubout/Del	löscht und echot das letzte Zeichen
Control/C	Warmstart des CP/M (falls am Zeilenanfang)
Control/E	Physikalisches Zeilenende
Control/H	Rückwärtsschritt
Control/J	Zeilenvorschub - Ende der Eingabe
Control/M	Wagenrücklauf - Ende der Eingabe
Control/R	Gibt die Eingabe in einer neuen Zeile noch-einmal aus
Control/U	Löscht die ganze Zeile
Control/X	wie Control/U

Anzumerken ist, dass alle Steuerzeichen, die einen Wagenrücklauf zur Folge haben, nur bis zu der Spalte, die auf den Prompt folgt zurückgehen.

```
+-------------------------------------------------+
!                                                 !
! Funktion 11: Holen Konsolstatus                 !
!                                                 !
! Eintrittsparameter:                             !
!    Register  C: 0BH                             !
!                                                 !
! Resultat:                                       !
!    Register  A: Konsolstatus                    !
!                                                 !
+-------------------------------------------------+
```

Der Konsolstatus prüft, ob ein Zeichen an der Konsole eingegeben wurde. Falls ein Zeichen ansteht, enthält Register A einen Wert ungleich 0, sonst 00H.

```
+-------------------------------------------------+
!                                                 !
! Funktion 12: Versionsnummer                     !
!                                                 !
! Eintrittsparameter:                             !
!    Register  C: 0CH                             !
!                                                 !
! Resultat:                                       !
!    Register HL: Versionsnummer                  !
!                                                 !
+-------------------------------------------------+
```

Diese Funktion erlaubt versionsunabhängige Programme. Es wird ein 2-Byte-Wert zurückgegeben. H enthält für CP/M den Wert 00H, für MP/M den Wert 01H. Für alle Versionen vor 2.0 ist L=00. Ab CP/M 2.0 enthält L den Wert 20. Alle weiteren Revisionen wie 2.1, 2.2, 2.3, ... , 2.16 enthält das Register L die Hexadezimalen Werte 21, 22, 23, ... ,2F. Mit dieser Funktion kann man Programme schreiben die sowohl sequentiellen als auch wahlfreien Dateizugriff unterstützen.

```
+-----------------------------------------------+
!                                               !
! Funktion 13: Disksystem zurücksetzen          !
!                                               !
! Eintrittsparameter:                           !
!    Register  C: 0DH                           !
!                                               !
+-----------------------------------------------+
```

Diese Funktion ermöglicht das programmgesteuerte Rücksetzen des Disksystems. Alle Disketten werden auf read/write (siehe Funkt. 28 und 29) gesetzt, Laufwerk A wird selektiert und die DMA-Pufferadresse auf BOOT+0080H gesetzt. Diese Funktion kann beispielsweise für ein Benutzeprogramm verwendet werden, das einen Diskettenwechsel ohne Warmstart des Systems benötigt.

```
+-----------------------------------------------+
!                                               !
! Funktion 14: Laufwerk wählen                  !
!                                               !
! Eintrittsparameter:                           !
!    Register  C: 0EH                           !
!    Register  E: Laufwerknummer                !
!                                               !
+-----------------------------------------------+
```

Die Laufwerknummer in Register E stellt das für die weiteren Operationen verwendete Laufwerk ein. E=0 für Laufwerk A, E=1 für Laufwerk B und so weiter bis E=15 für Laufwerk P. Das entsprechende Laufwerk wird angewählt und dadurch das Directory dieses Laufwerks bis zum nächsten Kalt- oder Warmstart beziehungsweise zum nächsten Rücksetzen des Disksystems aktiviert. Beim Wechsel der Diskette bei einem angewählten Laufwerk wird dieses normalerweise schreibgeschützt (siehe Funktion 28). Bei allen Dateiangaben ohne explizit angegebenen Laufwerknamen (dr=00 im FCB) wird das voreingestellte Laufwerk verwendet, bei Angabe des Laufwerknamens (A-P; dr = 1...16 im FCB) wird das entsprechende Laufwerk direkt angewählt.

```
+-----------------------------------------------+
!                                               !
! Funktion 15: Datei eröffnen                   !
!                                               !
! Eintrittsparameter:                           !
!    Register  C: 0FH                           !
!    Register DE: FCB-Adresse                   !
!                                               !
! Resultat:                                     !
!    Register  A: Directory-Code                !
!                                               !
+-----------------------------------------------+
```

Diese Funktion wird verwendet um eine Datei zu aktivieren, die bereits auf der Diskette existiert. Das FDOS durchsucht das Directory des angewählten Laufwerks nach dem Dateinamen, der in den Positionen 1 bis 14 des FCBs steht, welcher durch die Register DE adressiert wird (das Byte s1 wird automatisch null-gesetzt). Das ASCII-Zeichen '?' (3FH) steht für jedes beliebige Zeichen beim Directory-Vergleich. Normalerweise werden keine Fragezeichen im Namen angegeben und die Bytes ex und s2 des FCB sind Null.
Wird der Name im Directory gefunden , kopiert das FDOS die relevante Information aus dem Directory in den FCB um so den lesenden oder schreibenden Zugriff auf die Datei zu erlauben. Auf eine Datei darf erst zugegriffen werden, wenn die Funktion erfolgreich bendet ist. Sie gibt in Register A einen Code mit einem Wert zwischen 0 und 3 zurück,wenn die Operation erfolgreich war. Ist der Wert des Registers A = 0FFH, dann konnte die Datei nicht gefunden worden. Falls Fragezeichen im FCB auftreten, wird die erste passende Datei eröffnet. Das Feld cr muss im FCB auf 0 gesetzt werden , falls die Datei sequentiell gelesen werden soll.

```
+-----------------------------------------------+
!                                               !
! Funktion 16: Datei schliessen                 !
!                                               !
! Eintrittsparameter:                           !
!    Register  C: 10H                           !
!    Register DE: FCB Adresse                   !
!                                               !
! Resultat:                                     !
!    Register  A: Directory Code                !
!                                               !
+-----------------------------------------------+
```

Diese Funktion ist invers zur vorhergehenden. Angenommen, der FCB, der durch DE adressiert wird, wurde vorher durch eine der Funktionen 15 oder 22 aktiviert. Dann überträgt die Funktion 16 den FCB in das Directory der angewählten Diskette. Der Dateisuchprozess ist derselbe wie beim Eröffnen. Der Ergebniswert in Register A liegt im Erfolgsfall wiedr zwischen 0 und 3, und ist 0FFH, falls die Datei nicht gefunden wurde. Eine Datei braucht nicht geschlossen werden, wenn ausschliesslich lesend zugegriffen wurde. Falls jedoch auf die Datei geschrieben wurde, ist die Abschlussoperation wichtig, da sie das Directory auf den neuen Stand bringt.

```
+-----------------------------------------------+
!                                               !
! Funktion 17: Durchsuchen DIR nach erstem      !
!              Eintrag                          !
!                                               !
! Eintrittsparameter:                           !
!    Register  C: 11H                           !
!    Register DE: FCB Adresse                   !
!                                               !
! Resultat:                                     !
!    Register  A: Directory Code                !
!                                               !
+-----------------------------------------------+
```

Diese Funktion durchsucht das Directory nach einem Dateinamen, der im FCB steht, der durch DE adressiert wird. Wird die DAtei nicht gefunden, enthält A den Wert 0FFH. Im anderen Fall zeigt der Inhalt von A (0,1,2 oder 3) an, dass die Datei vorhanden ist.Dann wird der DMA-Puffer mit dem Record gefüllt, der die Directory-Information enthält. Die relative Startadresse ist dabei A*32 (rotiere A um 5 Bits oder addiere A fünfmal). Obwohl sie normalerweise nicht gebraucht wird, kann sie so aus dem Puffer extrahiert werden.
Taucht auf einer Position des FCBs (Felder f1 bis ex) ein Fragezeichen auf ('?', ASCII 3FH /63 dez.), deckt es sich mit jedem Zeichen des Directory-Eintrags auf dem ausgewählten Laufwerk. Enthält das Feld dr ein Fragezeichen, wird das voreingestellte Laufwerk selektiert und die Funktion liefert jeden passenden Eintrag unabhängig davon, ob die Datei belegt ist und unabhängig vom User. Diese Funktion wird normalerweise nicht von Anwendungsprogrammen verwendet, sie erlaubt aber grosse Flexibilität beim Durchsuchen der Dateiverzeichnisse. Steht im Feld dr kein Fragezeichen, wird s2 auf 0 gesetzt.

```
+-----------------------------------------------+
!                                               !
! Funktion 18: Durchsuchen DIR nach nächstem    !
!              Eintrag                          !
!                                               !
! Eintrittsparameter:                           !
!    Register  C: 12H                           !
!                                               !
! Resultat:                                     !
!    Register  A: Directory Code                !
!                                               !
+-----------------------------------------------+
```

Diese Funktion ist der vorhergehenden ähnlich, es wird jedoch ab dem letzten passenden Eintrag weitergesucht. Der Resultatwert in A entspricht den Angaben bei den Funktionen 17 und 18.

```
+-----------------------------------------------+
!                                               !
! Funktion 19:  Datei löschen                   !
!                                               !
! Eintrittsparameter:                           !
!    Register  C: 13H                           !
!    Register DE: FCB Adresse                   !
!                                               !
! Resultat:                                     !
!    Register  A: Directory Code                !
!                                               !
+-----------------------------------------------+
```

Diese Funktion entfernt die Dateien, deren Namen sich mit dem FCB-Eintrag decken. Der FCB wird durch DE adressiert. Dateinamen und Dateityp dürfen Fragezeichen als Referenzen für alle Zeichen enthalten, nicht jedoch die Laufwerksangabe. Der Resultatwert in A hat denselben Aufbau wie bei den Funktionen 17,18,und 19.

```
+-----------------------------------------------+
!                                               !
! Funktion 20: sequentiell lesen                !
!                                               !
! Eintrittsparameter:                           !
!    Register  C: 14H                           !
!    Register DE: FCB Adresse                   !
!                                               !
! Resultat:                                     !
!    Register  A: Directory Code                !
!                                               !
+-----------------------------------------------+
```

Angenommen der durch DE adressierte FCB wurde durch die Funktionen 15 oder 22 aktiviert. Dann liest die Funktion 20 den nächsten Block zu 128 Byte aus der Datei in den DMA-Puffer. Der Block wird aus Position cr der Extension ex gelesen und cr auf die nächste Blocknummer erhöht. Tritt bei cr ein Überlauf auf, wird automatisch die nächste Extension eröffnet und cr auf 0 gesetzt. In A wird 00H zurückgegeben, wenn die Operation erfolgreich beendet wurde, währen ein Wert ungleich Null geliefert wird, wenn keine Daten verfügbar waren (z.B. end-of-file).

```
+-----------------------------------------------+
!                                               !
! Funktion 21: sequentiell schreiben            !
!                                               !
! Eintrittsparameter:                           !
!    Register  C: 15H                           !
!    Register DE: FCB Adresse                   !
!                                               !
! Resultat:                                     !
!    Register  A: Directory Code                !
!                                               !
+-----------------------------------------------+
```

Angenommen der durch DE adressierte FCB wurde durch die Funktionen 15 und 22 aktiviert. Dann schreibt diese Funktion einen Block zu 128 Byte, der im DMA-Puffer steht auf die Diskettendatei, die vom FCB benannt wurde. Der Block wird auf

Position cr der Datei geschrieben und danach cr erhöht. Wenn cr überläuft, wird automatisch die nächste logische Extension eröffnet und cr für den nächsten Schreibvorgang auf 0 gesetzt.Es kann in eine existierende Datei geschrieben werden, wobei neue Blöcke die früheren ersetzen. War die Schreiboperation erfolgreich, ist Register A = 00H, wogegen ein Wert ungleich Null anzeigt, dass die Diskette voll ist.

```
+-----------------------------------------------+
!                                               !
! Funktion 22: Datei kreieren                   !
!                                               !
! Eintrittsparameter:                           !
!    Register  C: 16H                           !
!    Register DE: FCB Adresse                   !
!                                               !
! Resultat:                                     !
!    Register  A:  Directory Code               !
!                                               !
+-----------------------------------------------+
```

Diese Funktion ähnelt der Funktion 15 (Datei eröffnen), jedoch muss hier im FCB der Name einer Datei angegeben werden, die noch nicht im Directory der angewählten Diskette existiert. Das FDOS kreiert eine neue Datei und macht einen neuen Directory-Eintrag sowohl im FCB als auch auf der Diskette. Der Programmierer muss sicherstellen, dass kein doppelter Dateiname auftritt. Falls die Gefahr besteht, dass die Datei schon existiert, sollte vorhere ein Löschfunktion aufgerufen werden. Enthält nach Ausführung der Funktion das Registe A einen Wert zwischen 0 und 3, war die Operation erfolgreich, ist A = 0FFH, dann ist die Diskette voll. Als Zusatzeffekt wird der FCB gleich richtig belegt, sodass ein Eröffenen der Datei nicht notwendig ist.

```
+-----------------------------------------------+
!                                               !
! Funktion 23: Datei umbenennen                 !
!                                               !
! Eintrittsparameter:                           !
!    Register  C: 17H                           !
!    Register DE: FCB Adresse                   !
!                                               !
! Resultat:                                     !
!    Register  A: Directory Code                !
!                                               !
+-----------------------------------------------+
```

Diese Funktion verwendet den durch DE adressierten FCB um alle Dateien mit dem Namen, der in den ersten 16 Bytes steht umzubenennen. Der neue Name steht in den folgenden 16 Bytes des FCB. Die Laufwerksnummer auf Position 0 wird verwendet um das gewünschte Laufwerk auszuwälen, wogegen für die Laufwerknummer auf Position 16 des FCB der Wert Null angenommen wird. Iat A = 0FFH, dann konnte keine Datei gefunden werden; im Erfolgsfall hat A einen Wert zwischen 0 und 3.

```
+-----------------------------------------------+
!                                               !
! Funktion 24: Login Vektor liefern             !
!                                               !
! Eintrittsparameter:                           !
!    Register  C: 18H                           !
!                                               !
! Resultat:                                     !
!    Register HL: Login Vektor                  !
!                                               !
+-----------------------------------------------+
```

Der Login Vektor ist ein 16-bit-Wert in den Registern HL, wobei das niederwertigste bit von L dem Laufwerk A und das höchstwertige bit von H dem Laufwerk H entspricht. Ist das dem Laufwerk entsprechende bit = 0, kann dies Laufwerk nicht angesprochen werden (nicht bereit oder nicht vorhanden), Ein 1-bit zeigt an, dass das entsprechende Laufwerk bereit und aktivierbar ist (als Ergebnis einer expliziten oder impliziten Selektion des Laufwerks durch eine Dateioperation).Die Kompatibilität mit früheren Versionen von CP/M ist gegeben, da die Register A und L denselben Wert enthalten.

```
+-----------------------------------------------+
!                                               !
! Funktion 25: Laufwerknummer angeben           !
!                                               !
! Eintrittsparameter:                           !
!    Register  C: 19H                           !
!                                               !
! Resultat:                                     !
!    Register  A: Laufwerknummer                !
!                                               !
+-----------------------------------------------+
```

Diese Funktion liefert die Nummer des Bezugslaufwerks, das gerade voreingestellt ist. Der Wert rangiert von 0 bis 15 entsprechend den Laufwerken A bis P.

```
+-----------------------------------------------+
!                                               !
! Funktion 26: Setzen DMA-Adresse               !
!                                               !
! Eintrittsparameter:                           !
!    Register  C: 1AH                           !
!    Register DE: DMA Adresse                   !
!                                               !
+-----------------------------------------------+
```

DMA ist eine Abkürzung für "Direct Memory Access" (= direkter Speicherzugriff). Diese Form des Zugriffs auf den Speicher tritt oft in Zusammenhang mit Hintergrundspeichern bei grösseren Rechnern auf, wo die Laufwerksteuereinheiten die Daten von der Platte direkt in den Speicher schreiben (also ohne Verwendung der CPU). Obwohl viele Computer kein DMA verwenden, sondern die Daten über E/A-Befehle des Prozessors in den Speicher transportieren, hat die DMA-Adresse ihre Bedeutung als diejenige Adresse, bei der ein Block zu 128 Byte vor einer Schreiboperation gespeichert oder nach einer Leseoperation abgelegt wird. Nach einem Kaltstart,

Warmstart oder Zurücksetzen des Disksystems wird diese Adresse automatisch auf BOOT+0080H gesetzt. Diese Funktion erlaubt es, die DMA-Adresse umzudefinieren und den DMA-Puffer in einen anderen Speicherbereich zu legen. Der Wert der Adresse bleibt gültig, bis er umdefiniert wird oder ein Kaltstart, Warmstart oder Rücksetzen des Disksystems erfolgt.

```
+-----------------------------------------------+
!                                               !
! Funktion 27: Holen Belegungsverzeichnis       !
!                                               !
! Eintrittsparameter:                           !
!    Register  C: 1BH                           !
!                                               !
! Resultat:                                     !
!    Register HL: Adresse Belegungsverzeichn.   !
!                                               !
+-----------------------------------------------+
```

Im Speicher wird ein Belegungsverzeichnis für jedes aktive Laufwerk geführt. Viele Systemprogramme verwenden die dort gespeicherte Information uber den noch zur Verfügung stehenden Speicher auf Diskette (z.B.: STAT). Funktion 27 liefert in HL die Basisadresse dieses Verzeichnisses für das augenblicklich angewählte Laufwerk. Die Information kann falsch sein, wenn das Laufwerk schreibgeschützt wurde. Obwohl diese Funktion normalerweise nicht verwendet wird, finden Sie weitergehende Informationen über das Belegungsverzeichnis in Kapitel 6.

```
+-----------------------------------------------+
!                                               !
! Funktion 28: Schreibschutz setzen             !
!                                               !
! Eintrittsparameter:                           !
!    Register  C: 1CH                           !
!                                               !
+-----------------------------------------------+
```

Das gerade selektierte Laufwerk wird schreibgeschützt. Jeder Versuch vor dem nächsten Kalt- oder Warmstart auf diese Diskette zu schreiben, liefert die Fehlermeldung:

```
BDOS ERR on d: R/O
```

```
+-----------------------------------------------+
!                                               !
! Funktion 29: Holen R/O Vektor                 !
!                                               !
! Eintrittsparameter:                           !
!    Register  C: 1DH                           !
!                                               !
! Resultat:                                     !
!    Register HL: Wert des Vektors              !
!                                               !
+-----------------------------------------------+
```

In den Registern HL steht, welche Laufwerke schreibgeschützt sind. Wie in Funktion 24 ist das niederwerigst bit von l dem Laufwerk A, das höchstwertige bit von H dem Laufwerk P zugeordnet. Das entsprechende Screibschutzbit wird entweder durch Funktion 28 gesetzt oder wenn CP/M feststellt, dass eine Diskette gewechselt wurde.

```
+-----------------------------------------------+
!                                               !
! Funktion 30: Setzen Dateiattribute            !
!                                               !
! Eintrittsparameter:                           !
!    Register  C: 1EH                           !
!    Register DE: FCB Adresse                   !
!                                               !
! Resultat:                                     !
!    Register  A: Directory Code                !
!                                               !
+-----------------------------------------------+
```

Diese Funktion erlaubt es, vom Programm aus die DAteiattribute zu ändern. Insbesondere können Schreibschutz und das Systemattribut (t1' und t2') gesetzt oder gelöscht werden. Die Register DE adressieren einen FCB mit eindeutigem Dateinamen, in dem die entsprechenden Attribute gesetzt oder gelöscht sind. Funktion 30 durchsucht das Directory nach der Datei und ändert den Eintrag entsprechend. Die Indikatoren f1' bis f4' werden im Augenblick noch nicht verwendet, können aber nützlich fur Anwendungsprogramme sein, da sie bei der Dateieröffnung (beim Schliessen) nicht mitverglichen werden . Die Indikatoren f5' bis f8' und t3' sind für Systemerweiterungen reserviert.

```
+-----------------------------------------------+
!                                               !
! Funktion 31: Adresse Diskparameter            !
!                                               !
! Eintrittsparameter:                           !
!    Register  C: 1FH                           !
!                                               !
! Resultat:                                     !
!    Register HL: DPP-Adresse                   !
!                                               !
+-----------------------------------------------+
```

In HL wird die Adresse des Disk Parameter Puffers zurückgegeben. Diese Adresse kann für zwei Zwecke verwendet werden. erstens lassen sich die Parameterwerte für die Anzeige und das Berechnen des Dateiplatzes verwenden, zweitens können Programme die Parameter ändern, wenn sich die Hardwareumgebung ändert. Normale Programmen brauchen diese Funktion nicht.

```
+-----------------------------------------------+
!                                               !
! Funktion 32: Holen/Setzen User                !
!                                               !
! Eintrittsparameter:                           !
!    Register  C: 20H                           !
!    Register  E: 0FFH (Holen) oder             !
!                 User Code (Setzen)            !
!                                               !
! Resultat:                                     !
!    Register  A: User Code oder kein Wert      !
!                                               !
+-----------------------------------------------+
```

Ein Benutzerprogramm kann mit dieser Funktion die gerade aktive Benutzernummer (User Code) abfragen oder ändern. Falls Register E den Wert 0FFH besitzt, wird die aktuelle Benutzernummer (0 ... 15) in Register A geliefert. Ist Register E ungleich 0FFH wird die Benutzenummer auf den Wert in E (modulo 16) gesetzt.

```
+-----------------------------------------------+
!                                               !
! Funktion 33: Wahlfrei Lesen                   !
!                                               !
! Eintrittsparameter:                           !
!    Register  C: 21H                           !
!    Register DE: FCB Adresse                   !
!                                               !
! Resultat:                                     !
!    Register  A: Fehlercode                    !
!                                               !
+-----------------------------------------------+
```

Diese Funktion ähnelt der Operation "sequentielles Lesen" früherer Versionen von CP/M, jedoch kann hier die Leseoperation für jede einzele Blocknummer ausgefüht werden. Der gewünschte Block wird durch die Felder r0, r1 und r2 (Positionen 33-35 im FCB) ausgewählt (24-bit-Wert). Zu beachten ist, dass der 24-bit-Wert mit seinem niederwertigsten Byte zuerst (r0) gespeichert wird. Danach folgen das mittelwertige (r1) und höchstwertige Byte(r2) der Blocknummer. CP/M verwendet r2 nur um die Grösse einer Datei zu berechnen (Funktion 35). Byte r2 muss jedoch auf Null gesetzt werden, da ein Wert ungleich Null einen überlauf über das Dateiende hinaus anzeigt.
Daher werden r0 und r1 als Doppelbyte - oder Wort - betrachtet, das die Blocknummer enthält. Die Nummer kann sich also im Bereich von 0 ... 65535 bewegen und so jeden Block in einer 8-Megabyte-Datei adressieren. Um eine Datei mit wahlfreiem Zugriff (Random-Datei) zu bearbeiten, muss sie mit der Extension 0 eröffnet werden. Obwohl die Basisextension keine Datenenthalten muss, stellt sie sicher, dass die Datei im Directory richtig aufgeführt wird und beim DIR-Kommando aufgelistet wird. Nach dem Eröffnen wird die gewünschte Blocknummer in den Feldern r0 und r1 des FCB eingetragen und das BDOS aufgerufen um den Block zu lesen. Nach der Rückkehr enthält Register A den Wert 00, wenn die Operation erfolgreich war oder einen Fehlercode (siehe unten). Im ersten Fall enthält der DMA-Puffer den wahlfrei gelesenen Block. Im Gegensatz zum sequentiellen Lesen wird hier die Blocknummer nicht verändert. Daher lesen aufeinanderfolgende Random-Zugriffe immer denselben Block, wenn die Werte r0 und r1 nicht gändert

werden.
Bei jedem Random-Zugriff werden die Felder ex und cr automatisch gesetzt. daher kann die Datei auch sequentiell gelesen werden, nachdem durch Random-Zugriff auf einen Block positioniert wurde. Dabei wird beim Umschalten auf sequentielles Lesen der mit Random-Zugriff gelesene Block als erster nocheinmal gelesen. Der Programmierer kann selbstverständlich auch einfach die Blocknummer erhöhen um die Random-Datei sequentiell zu lesen.
Fehlernummern in Register A beim wahlfreien Lesen:

01	Lesen eines nicht beschriebenen (leeren) Blocks
02	-
03	Augenblickliche Extension kann nicht geschlossen werden
04	Suche nach nicht beschriebenener Extension
05	-
06	Suche hinter dem physikalischen Dateiende

Die Fehler 01 und 04 treten auf, wenn der Versuch erfolgt einen Block zu lesen, der nie vorher beschrieben wurde, oder eine Extension zu eröffnen, die nicht kreiert wurde. Fehler 03 tritt normalerweise nicht auf. Falls doch, kann er beseitigt werden, indem Extension 0 nocheinmal gelesen oder eröffnet wird (vorausgesetzt die Diskette ist nicht Physikalisch schreibgeschützt). Fehler 06 tritt immer dann auf, wenn r2 ungleich Null ist (Version CP/M 2.0). Normalerweise können die Fehlercodes als Anzeige für fehlende Daten aufgefasst werden, während A = 00H anzeigt, dass die Leseoperation erfolgreich war.

```
+-----------------------------------------------+
!                                               !
! Funktion 34: Wahlfrei Schreiben               !
!                                               !
! Eintrittsparameter:                           !
!    Register  C: 22H                           !
!    Register DE: FCB Adresse                   !
!                                               !
! Resultat:                                     !
!    Register  A: Fehlercode                    !
!                                               !
+-----------------------------------------------+
```

Diese Operation wird ähnlich wie die vorhergehende initiiert, nur dass hier Daten nicht gelesen, sondern auf die Datei aus dem DMA-Puffer geschrieben werden. Weiterhin wird, falls der entsprechende Block oder die entsprechende Extension noch nicht eingerichtet wurden, dies vor dem Schreiben erledigt. Genau wie beim wahlfreien Lesen wird auch hier die Blocknummer nicht verändert. Die Felder ex und cr werden entsprechend der Blocknummer im FCB gesetzt. Es kann nach einer wahlfreien Schreiboperation sequentiell gelesen oder geschrieben werden, wobei der wahlfrei zugegriffene Block bei der ersten sequentiellen Operation gelesen oder überschrieben wird. Der Benutzer kann auch hier die Blocknummer erhöhen um einen sequentiellen Zugriff zu erreichen.
Die Fehlercodes sind die gleichen wie beim wahlfreien Lesen,

wobei noch ein Fehler hinzukommt:

05 Neue Extension lässt sich nicht kreieren, da das Directory überläuft.

```
+-----------------------------------------------+
!                                               !
! Funktion 35: Dateigrösse berechnen            !
!                                               !
! Eintrittsparameter:                           !
!    Register  C: 23H                           !
!    Register DE: FCB Adresse                   !
!                                               !
! Resultat:                                     !
!    Blockzahl im FCB gesetzt                   !
!                                               !
+-----------------------------------------------+
```

Zum Berechnen der Dateigrösse muss durch DE ein FCB adressiert werden, der die Felder r0 bis r2 enthält sowie einen eindeutigen Dateinamen. Nach Rückkehr vom FDOS enthält der FCB die virtuelle Dateilänge, die die Blocknummer des auf das Dateiende folgenden Blocks ist. Wenn nach einem Aufruf der Funktion 35 das Feld r1 des FCB = 01 ist, hat die Datei die maximale Blockzahl von 65535 erreicht. Im anderen Fall (r2 = 0), bilden r0 und r1 einen 16-bit-Wert (r0 ist wieder das niederwertige Byte), der die Dateigrösse angibt.
An eine vorhandene Datei lassen sich Daten anfügen, indem durch Aufruf der Funktion 35 auf das Dateiende positioniert wird und nun durch eine Folge von wahlfreien Schreiboperationen an der ermittelten Blocknummer begonnen wird.
Die virtuelle DAteigrösse entspricht der physikalischen Dateigrösse, wenn die Datei sequentiell beschrieben wurde. Wenn die Datei wahlfrei kreiert wurde und "Löcher" in der Belegung existieren, kann die Datei weniger Blöcke enthalten, als die Dateigrösse angibt. Wenn zum Beisiel nur der letzte Block einer 8-Megabyte-Datei geschrieben wurde, so ist die virtuelle Dateigrösse 65536 Blöcke, obwohl nur ein Block tatsächlich belegt wurde.

```
+-----------------------------------------------+
!                                               !
! Funktion 36: Setzen wahlfreien Block          !
!                                               !
! Eintrittsparameter:                           !
!    Register  C: 24H                           !
!    Register DE: FCB Adresse                   !
!                                               !
! Resultat:                                     !
!    Random Feld im FCB gesetzt                 !
!                                               !
+-----------------------------------------------+
```

Diese Funktion veranlasst das BDOS, die Nummer eines wahlfrei zugreifbaren Blocks aus der augenblicklich beim sequentiellen Bearbeiten der Datei erreichten Position zu berechnen.Diese Funktion kann in zwei Fällen nützlich sein.
Erstens ist es oft notwendig, eine Datei erst sequentiell zu

lesen um die Position einiger Schlüsselfelder aufzufinden. Wenn jeder Schlüssel gefunden ist, kann die Funktion 36 verwendet werden, um die Blocknummer des Schlüssels für einen direkten Zugriff zu berechnen. Ist die Datenrecordgrösse 128 Byte, kann die resultierende Blocknummer des Schlüssel für spätere Abfragen in einer Tabelle gespeichert werden. Nachdem die Datei vollständig durchsucht wurde und alle Schlüssel in einer Tabelle gespeichert wurden kann sofort mit einem wahlfreien Lesezugriff über den Schlüssel auf die Daten zugegriffen werden.
Eine zweite Anwendung für die Funktion 36 bietet sich beim Wechsel vom sequentiellen zum wahlfreien Zugriff. Eine Datei wird sequentiell bis zu einem bestimmten Punkt bearbeitet, dann Funktion 36 aufgerufen (die die Blocknummer berechnet) und danach weitere Zugriffe wahlfrei von dem berechneten Punkt ausgehend durchgeführt.

```
+-----------------------------------------------+
!                                               !
! Funktion 37: Zurücksetzen Laufwerk            !
!                                               !
! Eintrittsparameter:                           !
!    Register  C: 25H                           !
!    Register DE: Laufwerkvektor                !
!                                               !
! Resultat:                                     !
!    Register  A: 00H                           !
!                                               !
+-----------------------------------------------+
```

Diese Funktion erlaubt das rücksetzen der spezifizierten Laufwerke. Der in DE übergebene Parameter ist ein 16-bit-Vektor der Laufwerke, die zurückgesetzt werden sollen. Das niederwertigste bit entspricht dem Laufwerk A, das höchstwertige dem Laufwerk P.
Um Kompatibilität mit MP/M sicherzustellen, liefert CP/M den Wert 0 in A zurück.

```
+-----------------------------------------------+
!                                               !
! Funktion 40: Wahlfrei schreiben, mit          !
!              Nullen füllen                    !
!                                               !
! Eintrittsparameter:                           !
!    Register  C: 28H                           !
!    Register DE: FCB Adresse                   !
!                                               !
! Resultat:                                     !
!    Register  A: Fehlercode                    !
!                                               !
+-----------------------------------------------+
```

Diese Funktion arbeitet wie Funktion 34, es werden jedoch vorher nicht zugegriffene Blöcke mit Nullbytes aufgefüllt, bevor die Daten geschrieben werden.

CP/M ASSEMBLER

Der CP/M Assembler liest Dateien mit Assembler-Quellprogrammen ein und erzeugt 8080-Maschinencode im Intel Hexformat. Er wird durch das Kommando

```
ASM dateiname
```

oder

```
ASM dateiname.parameter
```

aufgerufen. In beiden Fällen wird angenommen, daß sich auf der Diskette eine Datei mit Namen

```
dateiname.ASM
```

befindet. Diese Datei enthält eine Quelle in 8080-Assemblersprache. In der zweiten Form des Kommandos werden dem ASM Parameter mitgegeben, die den Zugriff auf die Quelldatei und die erzeugten Dateien mit dem Code und dem Listing steuern. Mit dem Kommando ASM wird der Assembler geladen und gestartet. Er meldet sich auf der Konsole mit

```
CP/M ASSEMBLER VER n.m
```

wobei n.m die Versionsnummer ist. Beim parameterlosen ASM-Kommando erzeugt der ASM aus der Quelldatei "name.ASM" die beiden Dateien

```
name.HEX
```

und

```
name.PRN.
```

In der HEX-Datei steht der erzeugte Maschinencode im Intel Hexformat. Die PRN-Datei enthält ein Listing, in dem die Fehlermeldungen des Assemblers, der erzeugte Code und ein Protokoll des Quellprogramms steht. Der Parameterteil des Kommandos besteht aus drei Buchstaben, wovon der erste das Laufwerk der Quelldatei, der zweite das Ziel der HEX-Datei und der dritte das Ziel der PRN-Datei festlegen. Das Kommandoformat ist also

```
ASM dateiname.p1p2p3
```

wobei die einzelnen Buchstaben p1, p2 und p3 die folgende Bedeutung besitzen:

p1:	A, B, ... , P	Laufwerksname der Quelldatei
p2:	A, B, ... , P	Laufwerksname der HEX-Datei
	Z	Die HEX-Datei wird nicht erzeugt
p3:	A, B, ... , P	Laufwerksname der PRN-Datei
	X	Das Protokoll wird auf die Konsole geleitet.
	Z	Die PRN-Datei wird nicht erzeugt.

Ein Assemblerprogramm besteht aus einer Folge von Programmzeilen, die das folgende Format haben:

```
Zeilennummer Marke Operationscode Operand ; Kommentar
```

Eine Zeile muß nicht alle oben angegebene Elemente enthalten. Jede Assemblerzeile wird mit Wagenrücklauf (cr) und Zeilenvorschub (lf) abgeschlossen oder mit dem Zeichen "!" beendet. Das Ausrufezeichen wird vom Assembler als Zeilenende interpretiert.
Die Zeilennummer wird vom Assembler ignoriert.

Die Marke hat die Form

```
Bezeichner
```

oder

```
Bezeichner:
```

Ein Bezeichner ist eine Folge von alphanumerischen Zeichen, wobei das erste Zeichen ein Buchstabe sein muß. Sie können maximal 16 Zeichen lang sein. In einem Bezeichner sind alle Zeichen bis auf das Dollarzeichen ($) signifikant. Das Dollarzeichen kann eingefügt werden, um die Lesbarkeit der Bezeichner zu verbessern. Alle Kleinbuchstaben werden

genau wie Großbuchstaben betrachtet. Gültige Marken sind:

```
x          xy          langer$name
x:         xyl:        sehr$langer$name:
X1Y1       X1Y2        X234$5678$9012$3456:
```

Der Operationscode ist entweder ein 8080-Opcode, eine Assemblerdirektive oder eine Pseudo-Operation sein.
Die Operanden enthalten Ausdrücke mit Konstanten und Marken, die auch mit arithmetischen und logischen Operationen verknüpft werden können.

Zwischen dem ";" und dem Zeilenende können Kommentare angegeben werden. Diese Kommentare erläutern die Befehle und können beliebige Zeichenfolgen enthalten. Sie werden von Assembler gelistet, aber im übrigen ignoriert. Steht in der ersten Spalte einer Zeile ein "*", wird die ganze Zeile als Kommentar betrachtet.

Die Ausdrücke im Operanden bestehen aus Marken, Konstanten und reservierten Worten, die mit arithmetischen und logischen Operationen verknüpft werden. Der Assembler erzeugt für jeden Ausdruck einen 16-bit-Wert. Weiterhin darf die Zahl der gültigen Stellen des Resultats den vorgeschriebenen Maximalwert nicht überschreiten. Wird beispielsweise ein Ausdruck in einem ein-Byte-Sprung verwendet, müssen die 8 höherwertigen Bits des Wertes Null sein.

Der Wert einer Marke hängt vom nachfolgenden Befehl ab. Wird durch den auf die Marke folgenden Befehl Code generiert (z.B.: MOV) oder Speicherplatz reserviert (z.B.:DS) erhält die Marke als Wert die markierte Programmadresse. Steht eine Marke vor einer der Pseudo-Operationen EQU oder SET, wird ihr als Wert das Ergebnis des auf die Pseudo-Operation folgenden Operanden-Ausdrucks zugewiesen. Mit Ausnahme der SET-Operation kann eine Marke nur eine einzige Anweisung markieren. Im Operanden vorkommende Bezeichner von Marken werden durch ihren Wert substituiert und können so in Operanden-Ausdrücken mit anderen Marken und Konstanten verknüpft werden.

Eine numerische Konstante ist eine 16-bit-Zahl zu den verschiedensten Basen. Die Basis oder Radix wird durch Buchstaben gekennzeichnet, die der Zahl nachgestellt sind:

B	binäre Konstante	Basis 2
O	oktale Konstante	Basis 8
Q	oktale Konstante	Basis 8
D	dezimale Konstante	Basis 10
H	hexadezimale Konst.	Basis 16

Da der Buchstabe "O" leicht mit der Zahl "0" verwechselt werden kann, ist der Indikator "Q" bei der Bezeichnung von Oktalzahlen zu bevorzugen. Numerische Konstante ohne einen Basis-Indikator werden als Dezimalzahlen betrachtet.

Es gibt einige Bezeichner, die eine vordefinierte Bedeutung im Operanden haben. Die Namen der 8080-Register unten ergeben im Operanden den danebenstehenden Wert:

A	7
B	0
C	1
D	2
E	3
H	4

```
L    5
M    6
SP   6
PSW  6
```

Maschinenbefehle können auch als Operanden auftreten; sie liefern den Wert ihres intenen Codes. Im Fall, daß der Operand das Bitmuster des Befehl beeinflußt (z.B. MOV A,B), wird der Wert des Operationscodes mit Nullen im Oprandenteil genommen (im Falle MOV also 40H).
Tritt das Dollarzeichen($) alleine stehend im Operanden auf (also nicht innerhalb eines Bezeichners oder einer Konstanten) liefert er den Stand des Befehlszählers, also die Adresse der folgenden Instruktion.

Eine Zeichenketten- oder String-Konstante besteht aus einer Folge von ASCII-Zeichen, die in Apostrophe (') eingeschlossen ist. Um des Zeichen "!" in Strings zu erlauben, müssen sie vollständig in einer (physikalischen) Zeile stehen. Strings dürfen maximal 64 Zeichen lang sein. Um einen Apostroph (') in einer Zeichenkette darzustellen, wird er verdoppelt (''), aber nur als ein Zeichen übersetzt. Der Wert eines Zeichen wird durch seine ASCII-Zuordnung festgelegt. Innerhalb von Zeichenketten gibt es keine Konvertierung von Klein- in Großschreibung. In Strings dürfen nur druckbare ASCII-Zeichen vorkommen.

Die oben beschriebenen Operanden können mit normaler algebraischer Notation in vollständig geklammerten Ausdrücken verknüpft werden. Erlaubte Operationszeichen (Operatoren) sind:

a + b	Addition von a und b
a - b	Subtraktion von a und b
+ b	positives Vorzeichen (entspricht b)
- b	negatives Vorzeichen (entsp. 0 - b)
a * b	Vorzeichenlose Multiplikation von a und b
a / b	Vorzeichenlose, ganzzahlige Division von a durch b
a MOD b	Rest der Division von a durch b
NOT b	bitweise Negation der 16-bit-Zahl b (alle 0-bits werden 1 und umgekehrt).
a AND b	bitweise Und-Verknüpfung von a und b
a OR b	bitweise Oder-Verknüpfung von a und b
a XOR b	bitweise Exklusiv-Oder-Verknüpfung von a und b
a SHL b	Schiebe a um b Bitpositionen nach links und fülle mit Nullen auf (shift left).
a SHR b	Schiebe a um b Bitpositionen nach rechts und fülle mit Nullen auf (shift right).

In jedem der Fälle stellen a und b einfache Operanden (Marken, Konstante, reservierte Bezeichner oder Strings von 1 oder 2 Zeichen Länge) oder geklammerte Unterausdrücke dar.
Alle Berechnungen bei der Assemblierung werden als vorzeichenlose 16-bit-Operationen ausgeführt. -1 wird also als 0 - 1 berechnet und das Resultat ist 0FFFFH (lauter Einsen). Das Ergebnis einer Operandenauswertung muß in den Operationscode passen, der es benützt. So darf beispielsweise der Ausdruck beim Operationscode ADI (add immediate) höchstens den Wert 0FFH (d.h. die 8 höchstwertigen Bits = 0) besitzen. So würde der Befehl "ADI -1" eine Fehlermeldung erzeugen, da -1 gleich 0FFFFH zu groß ist. Dagegen würde die Operation "ADI (-1) and 0ffh" vom Assembler akzeptiert, da durch die AND-Operation die vorderen Bits zu 0 werden.
ASM kennt Vorrangregeln dr einzelnen Operatoren, sodaß der Benutzer meist ohne Klammern auskommt. Den höchsten Vorrang besitzen dieOpratoren in der obersten Zeile, die niedrigste Stufe besitzen jene der untersten Zeile. Ausdrücke werden nach Klammerung und Vorrangstufen und im übrigen

vom links nach rechts ausgewertet.

```
*  /  MOD  SHL  SHL
          -  +
          NOT
          AND
        OR   XOR
```

Assemblerdirektiven oder Pseudo-Operationen werden verwendet um Marken bestimmte Werte zuzuweisen, Bedingte Assemblierung durchzuführen, Speicherplatz zu definieren und Programmanfangsadressen festzulegen.

ORG
Der ORG-Befehl hat die Form

```
marke  ORG  ausdruck
```

Wobei die Marke optional ist. Der "ausdruck" liefert einen 16-bit-Wert, der aus Werten bestimmt wird, die vor der ORG-Anweisung definiert wurden. Der Assembler beginnt mit der Code-Generierung bei der so definierten Adresse.
Wird bei ORG eine Marke angegeben, erhält sie der Wert des Ausdrucks (sie kann dann in Operanden anderer Befehle vorkommen).

END
Der END-Befehl ist nicht unbedingt notwendig. Ist er vorhanden, muß er der letzte Befehl des Programms sein (alle folgenden Zeilen werden beim Assemblieren ignoriert). Es gibt zwei Formen des Befehls:

```
marke END
marke END ausdruck
```

Die Marke ist optional. Bei der ersten Form wird die Assemblierung bbendet und als Startadresse des Programms 0000H genommen. Bei der zweiten Form legt der Ausdruck die Startadresse fest (der letzte Satz der Intel-HEX-Datei enthält die Startadresse).

EQU
Der EQU-Befehl definiert Prögramakonstante, d.h. er weist einem Bezeichner einen festen numerischen Wert zu. Die Form ist

```
marke EQU ausdruck
```

Hier muß die Marke unbedingt angegeben werden und sie muß sich von allen anderen Marken unterscheiden. Der Wert des Ausdrucks wird der Marke zugewiesen.

SET
Die SET-Anweisung ähnelt der EQU-Anweisung. Sie hat die Form

```
marke SET ausdruck
```

Im Gegensatz zu EQU kann der Wert der Marke in weiteren SET-Befehlen neu definiert werden. Es wird der Ausdruck berechnet und der Marke jeweils der aktuelle Wert zugewiesen. SET wird wie EQU verwendet, jedoch wird die bedingte Assemblierung mit SET gesteuert.
SET kann eine Marke also mehrfach mit einem Wert belegen, EQU nur ein einziges mal.

IF und ENDIF
Diese Direktiven definieren einen Bereich von Assemblerzeilen, die nur unter bestimmten Bedingungen bearbeitet werden. Die Form ist:

```
IF Ausdruck
Befehl 1
Befehl 2
...
Befehl n
```

ENDIF

Sößt der Assembler auf einen IF-Befehl, wird der Ausdruck nach IF ausgewertet (dabei müssen alle Operanden in diesem Ausdruck vorher definiert sein). Ist dr Wert des AUsdrucks ungleich Null, werden die Zeilen zwischen IF und ENDIF assembliert; ist der Wert gleich Null werden die Zeilen nur gelistet (wie Kommentare). Bedingte Assemblierung wird verwendet, um allgemein verwendbare Programme zu schreiben, die dann an verschidene Laufzeitumgebungen durch die Definition eier Konstanten angepasst werden können.

DB

Mit dieser Direktive kann Speicherplatz byteweise belegt und initialisiert werden. Die Form des Befehls ist:

marke DB a1,a2,a3, ... , an

wobei a1 bis an Ausdrücke sind, die einen 8-bit-Wert besitzen (das höherwertige Byte gleich 00). Ebenso sind ASCII-Strings möglich, die nicht mehr als 64 Zeichen enthalten. Es gibt also keine praktische Einschränkung für die Zahl der vorkommenden Ausdrücke. Die Ausdrücke werden von links nach rechts ausgewertet und die Ergebnisbytes fortlaufen ab der aktuellen Adresse im Speicher abgelegt. In Ausdrücken können nur Strings zu ein oder zwei Zeichen verwendet werden. Wichtig ist, da8 ASCII-Zeichen immer ohne Paritätsbit abgelegt werden (Bit 8 = 0). Die optionale Marke kann im Programm verwendet werden, um die Datenbereiche anzusprechen.

DW

Die Direktive entspricht der DB-Anweisung, nur das hier der Speicher mit Doppelbytes (Worten) initialisiert wird. Seine Form ist:

marke DW a1,a2,a3, ... ,an

wobei a1 bis an diesmal 16-bit-Ausdrücke sind. Es sind hier nur Strings mit einem oder zwei Zeichen erlaubt. In allen Fällen ist die Speicherung der Daten mit dem Intel 8080 konsistent. Das niederwertige Byte des Ausdrucks wird zuerst im Speicher abgelegt; danach folgt das höherwertige Byte.

DS

Die DS-Direktive reserviert einen nichtinitialisierten Speicherbereich. Sie hat die Form:

marke DS ausdruck

Die Marke ist optional. Der Ausdruck gibt an, wieviele Bytes reserviert werden sollen. Die Codegenerierung wird nach dem reservierten Speicherbereich fortgesetzt.

Operationscodes

Generell akzeptiert ASM alle mnemotechnischen Abkürzungen für Befehle (Mnemonics) für den Intel 8080 Prozessor, wie sie im Handbuch von Intel "8080 Assembly Language Programming Manual" beschrieben sind. In den folgenden Abschnitten stellen die Operanden dar:

e3 einen 3-bit-Wert zwischen 0 und 7. Es kann hier eines der vordefinierten Register A, B, C, D, E, F, H, L, M, SP oder PSW stehen.

e8 einen 8-bit-Wert (ein Byte) zwischen 0 und 255.

e16 einen 16-bit-Wert zwischen 0 und 65535.

Die Ausdrücke e3, e8 oder e16 können durch beliebige Operanden-Ausdrücke dargestellt werden. In manchen Fällen (z.B.: PUSH) sind die Operanden auf bestimmte feste Werte beschränkt.

Sprünge, Unterprogrammaufruf und -rücksprung.
Die Sprung- (Jump), Unterprogrammaufruf- (Call) und UP-Rücksprung- (Return) Instruktionen haben die verschiedensten Formen, je nachdem, welche Bedingungen (Bedingungsbits = condition flags) den Sprung auslösen. Die Bedingungen werden wie folgt abgekürzt:

NZ	(non zero)	ungleich Null
Z	(zero)	gleich Null
NC	(no carry)	Übertragsbit gleich Null
C	(carry)	Übertragsbit ungleich Null
PO	(parity odd)	Parität ungerade
PE	(parity even)	Parität gerade
P	(plus)	Vorzeichen positiv
M	(minus)	Vorzeichen negativ

Diese Bedingungen können bei Sprüngen, UP-Aufrufen und UP-Returns auftreten. "Jbed" in der Tabelle unten steht dann für JNZ, JZ, JNC, JC,

JMP	e16	JMP L1	Springe nach Marke L1
Jbed	e16	JNZ L2	Springe nach L2, falls ungleich 0
		JZ $+100H	Springe 256 Bytes vorwärts,falls 0
		...	
CALL	e16	CALL TTY	Rufe Unterprogramm TTY auf
Cbed	e16	CNZ SUM	Rufe SUM auf,falls letztes Erg.#0
		CM L2	Rufe L2 auf, falls letztes Erg.#0
RET		RET	Rückkehr vom UP
Rbed		RNZ	Rücksprung vom UP, falls ERg. # 0
RST	e3	RST 7	Restart mit der entsprechenden Nummer. Entspricht "CALL e3*8".

Immediate-Befehle

MVI e3,e8	MVI B,255	(move immediate) Bringe den Wert e8 in Register e3 (A, B, C, D, E, H, L oder M (memory = Speicher).
ADI e8	ADI 1	(add immediate) Addiert den Wert e8 zum Akkumulator ohne Carry.
ACI e8	ACI 0ffh	(add with carry immediate) Wie ADI, jedoch mit Carry (übertrag).
SUI e8	SUI L+3	(subtract immediate) Subtrahiere ohne Übertrag(borrow)
SBI e8	SBI L	(subtract with borrow) Subtrahiere mit übertrag
ANI e8	ANI 0fh	(and immediate) Und-Verknüpfung zwischen A und Immediate-Wert
XRI e8	XRI 111B	(exclusive or immediate) Exclusiv-Oder mit A.
ORI e8	ORI 10H	(or immediate) Oder-Verknüpfung mit A.

CPI e8	CPI 'z'	(compare immediate) Vergleiche A mit dem Imm.-Wert. Wie SUI, jedoch wird A nicht verändert.
LXI e3,e16	LXI B,100H	(load extended immediate) Das Registerpaar e3 wird mit dem Imm.-Wert geladen (e3 gleich B, D, H oder SP).

Increment und Decrement

INR e3	INR E	(increment register) Erhöhe den Wert des Registers e3 um 1.
DCR e3	DCR A	(decrement register) Vermindere den Wert des Registers e3 um 1.
INX e3	INX SP	(increment extended) Erhöhe den Wert des 16-bit-Registers e3 um 1. e3 gleich B, D, H oder SP.
DCX e3	DCX H	(decrement extended) Vermindere den Wert des 16-bit-Registers e3 um 1. e3 gleich B, D, H oder SP.

Datentransport

MOV e3,e3	MOV a,b	(move) Transportiert Daten vom rechten Register in das linke (im Beispiel von b nach a). e3 gleich A, B, C, D, E, H, L, M (memory).
LDAX e3	LDAX B	(load accu indirect) Lade A von der berechneten Adresse (e3 gleich B oder D).
STAX e3	STAX D	(store accu indirekt) Speichere A auf der berechneten Adresse (e3 gleich B oder D).
LHLD e16	LHLD M1	(load HL direct) Läd HL aus der Adesse e16.
SHLD e16	SHLD M1	(store HL direct) Speichert HL auf die Adresse e16.
LDA e16	LDA GAMMA	(load accu) Läd Register A auf der adresse e16.
STA e16	STA X3-5	(store accu) Speichere A auf Adresse e16.
POP e3	POP PSW	(pop stack) Hole Register e3 vom Stack (e3 gleich B, D, H oder PSW).
PUSH e3	PUSH PSW	(push onto stack) Rette Register e3 auf dem Stack (e3 gleich B, D, H oder PSW).

IN e8	IN 20h	(input from port) lade Register A vom Port e8.
OUT e8	OUT TTY	(output to port) Sende Daten aus Register A zum Port e8.
XTHL	XTHL	(exchange TOS with HL) Tausche Daten auf oberster Stackposition mit HL.
PCHL	PCHL	(load PC from HL) Lade Programmzähler mit dem Inhalt von HL.
SPHL	SPHL	(load SP from HL) Lade Stackpointer mit dem Inhalt von HL.
XCHG	XCHG	(exchange) Tausche den Inhalt von DE mit HL.

Arithmetik und Logik

ADD e3	ADD B	(add) Addiere A und das Register e3.
ADC e3	ADC C	(add with carry) Addiere Register e3 und A mit Übertrag.
SUB e3	SUB A	(subtract) Subtrahiere e3 von A ohne Übertrag.
SBB e3	SBB H	(subtract with borrow) Subtrahiere e3 von A mit Übertrag.
ANA e3	ANA 1+1	(and with accu) Logische Und-Verknüpfung von 3 und A.
XRA e3	XRA A	(exclusive or with accu) Logische Exklusiv-Oder-Verknüpfung von e3 und A.
ORA e3	ORA B	(or with accu) Logische Oder-Verknüpfung von e3 mit A.
CMP e3	CMP H	(compare with accu) Vergleiche Register e3 mit A.
DAA		(decimal adjust accu) Dezimal-korrektur des Akkumulators A.
CMA		(complement accu) Komplementiere A
STC		(set carry) Setze übertagsbit.
CMC		(complement carry) Invertiere übertragsbit.
RLC		(rotate left) Rotiere A nach links. Carrybit wird gesetzt vom

höchstwertigen bit von A.

RRC — (rotate right) Rotiere A nach rechts. Das Carrybit wird gesetzt vom niederwertigsten bit von A.

RAL — (rotate left with carry) Rotiere A nach links über das Carrybit.

RAR — (rotate right with carry) Rotiere A nach rechts über das Carrybit.

DAD e3 DAD B — (double precision add) Addiere das Registerpaar e3 zu HL. e3 gleich B, D, H oder SP.

Steuerbefehle

HLT — (halt) Hält den 8080 an.

DI — (disable interrupt) Sperre Interrupt.

EI — (enable interrupt) Ermögliche Interrupt.

NOP — (no operation) Leerbefehl.

Fehlermeldungen
Treten Fehler in einem Assemblerprogramm auf, werden sie durch einen Buchstaben in der ersten Spalte markiert. Außerdem wird die Fehlerzeile auf der Konsole ausgegeben. Es muß also beim Auftreten von Fehlern nicht das ganze Listing ausgegeben werden.

D Data error: Data-Anweisung kann nicht richtig ausgeführt werden.
E Expression error: Ausdruck kann nicht ausgewertet werden.
L Label error: Marke kann nicht in diesem Kontext auftreten (eventuell doppelt definiert).
N Not implemented: Nicht implementierte Eigenschaft wie z. B. Makros.
O Overflow: Ausdruck zu kompliziert
P Phase error: Wert einer Marke wird umdefiniert
R Register error: Registerspezifikation unvoll ständig.
S Syntax error: Falsche Anweisung
U Undefined Label: nicht definierte Marke
V Value error: Fehler im Operanden

Es gibt auch einige Fehler, die den ASM terminieren:

NO SOURCE FILE PRESENT
Die spezifizierte Quelldatei ist nicht auf der Diskette vorhanden.

NO DIRECTORY SPACE
Das Directory der Diskette ist voll. Löschen Sie einige Daten und versuchen Sie es nocheinmal.

SOURCE FILE NAME ERROR
Der Name der Quelldatei ist falsch angegeben worden.

SOURCE FILE READ ERROR
Die Quelldatei kann vom ASM nicht richtig gelesen werden. Schauen Sie die Datei mit TYPE an.

OUTPUT FILE WRITE ERROR
Es kann nicht ordentlich auf die Ausgabedateien geschrieben werden. Diskette voll oder defekt.

CANNOT CLOSE FILE
Eine Ausgabedatei kann nicht geschlossen werden. Diskette wahrscheinlich schreibgeschützt.

1984

Druck: Franzis-Druck GmbH, Karlstraße 35, 8000 München 2
Printed in Germany. Imprimé en Allemagne.