Manfred Walter Thoma

CPC-464/664 Praxis III

Band 3: CP/M

Hüthig

Manfred Walter Thoma · CPC 464/664 - Praxis III

Manfred Walter Thoma

CPC-464/664 – Praxis III

Band 3: CP/M

Dr. Alfred Hüthig Verlag Heidelberg

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Thoma, Manfred Walter:
[CPC-vierhundertvierundsechzig-Praxis]
CPC-464-664-Praxis III / Manfred Walter Thoma. —
Heidelberg : Hüthig

Bd. 3. CP/M – 1986 –
ISBN 3-7785-1261-7

Printed in Germany
Satz und Druck: Druckerei Bitsch GmbH, Birkenau

Inhaltsverzeichnis

Programmverzeichnis

0 Vorwort

Der CPC-464/664 eröffnet durch seine CP/M-Fähigkeit eine vollkommen neue Perspektive im Homecomputerbereich. Vor dem CPC-464/664 waren nur wesentlich teurere Systeme CP/M-fähig oder mußten für viel Geld umgerüstet werden. Erstmals besteht auch für den Hobby- und Freizeitprogrammierer die Möglichkeit, mit einem ‚billigen' System in die Geheimnisse des CP/M-Betriebssystems einzusteigen und/oder eines der vielen unter CP/M lauffähigen Programme zu nutzen. Für das Arbeiten unter CP/M ist mindestens ein Diskettenlaufwerk notwendig (CPC-464 + DDI-1 oder CPC-664). Wie wichtig heute, zumindest für die 8-Bit Computergeneration, das CP/M-Betriebssystems ist, zeigt sich nicht zuletzt an der enormen Verbreitung und dem Angebot von (semi-) professioneller CP/M-Software.

Der CP/M-Benutzer wird mit drei Problemen ständig konfrontiert

1. Die Benutzung und Bedienung des CPC-464/664 unter CP/M.
2. Die richtige und sinnvolle Anwendung der CP/M-Kommandos.
3. Wie kann ich unter CP/M programmieren?

Die Bedienung des Systems und die Anwendung der CP/M-Kommandos (und Dienstprogramme) ist der zentrale Schritt für ein problemloses Arbeiten unter CP/M. Obwohl in diesem Band speziell auf die Möglichkeiten der Schneider-Computer unter CP/M eingegangen wurde, entsprechen sie bis auf wenigen Ausnahmen (spezielle Steuerzeichen, verschiedene Dienstprogramme) dem normalen Umfang des CP/M 2.2. Die hier erworbenen Kenntnisse lassen sich problemlos auf andere CP/M-fähige Computer transportieren.

Wesentlich interessanter sind für den Programmierer die Möglichkeiten der Programmierung unter CP/M. Hier werden die enormen Vorteile eines vereinheitlichten Betriebssystems besonders deutlich. Durch die einheitliche Lösung von vielen, teilweise sehr komplexen Ein-/Ausgabe-Funktionen, stehen dem Programmierer sehr leistungsstarke Werkzeuge für eine rechnerunabhängige Programmierung zur Verfügung. Diese vereinheitlichten Routinen (BDOS-Funktionen) bilden das Rückgrad des CP/M-Betriebssystems.

Der Ihnen hier vorliegende 3. Band der CPC-464/664 Praxis Reihe gliedert sich in zwei, weitgehend getrennte Teile

Der erste Teil beschäftigt sich mit der Bedienung und der Benutzung aller CP/M-Kommandos und aller 22 CP/M-Dienstprogramme, die auf der Systemdiskette zum CPC-464/664 mitgeliefert werden (SETUP, STAT, PIP, etc.).

Im zweiten Teil wird schwerpunktmäßig auf die technischen Aspekte des CP/M-Betriebssystems eingegangen. Hier findet eine detaillierte Beschreibung des Aufbaus und der Arbeitsweise des Systems statt. Weiterhin findet der Leser einen Einstieg in die Programmierung unter CP/M mit dem auf der Systemdiskette vorhandenen Assembler ‚ASM' (8080-Assembler). Einen großen Raum nehmen dabei die Beschreibungen aller BDOS-Funktionen ein, die fast alle mit einem Beispielprogramm erläutert werden. Am Ende dieses Buches finden Sie 4 umfangreichere Programme, die zeigen, wie die beschriebenen BDOS-Funktionen in der Praxis anzuwenden sind.

Durch den sehr übersichtlichen Aufbau läßt sich dieses Buch nicht nur als Arbeitsbuch im eigentlichen Sinne, sondern auch als Nachschlagewerk benutzen.

Hamburg, Mai 1986

1.0 Grundsätzliches über CP/M

CP/M (Control Program/Monitor) ist ein Betriebssystem, das von der Diskette in den Arbeitsspeicher des CPC-464/664 geladen wird. Nach dem Laden besitzt das CP/M-Betriebssystem die Kontrolle über den CPC. Weiterhin sind damit grundsätzliche Ein- und Ausgaberoutinen wie Zeichen von der Tastatur holen, Zeichen auf den Drucker ausgeben, Zugriffe auf die Diskette, etc. einheitlich geregelt. D.h. Routinen sind nicht mehr Rechnerspezifisch, sondern durch einen vereinheitlichten Sprung über das CP/M-Betriebssystem erreichbar. Damit ist es (theoretisch) möglich, ein unter CP/M erstelltes Programm auf jedem anderen CP/M-fähigen Rechner auszuführen. Diese Kompatiblität ist der entscheidende Vorteil des CP/M-Betriebssystems.

Das CP/M-Betriebssystem setzt allgemein einige Hardwareeigenschaften voraus

- 48/64 kByte RAM
- 8080-kompatibler Prozessor (Z80, 8085)
  (heute sind auch angepaßte Systeme für andere Prozessoren erhältlich 8086, 8088, 68000)

CP/M lauffähige Programme müssen in einem dem Prozessor verständlichen Code (beim CPC-464/664 in 8080- oder Z80-Assembler) geschrieben sein und ab der Adresse 0100H liegen (Im folgenden werden alle hexadezimalen Zahlen mit einem nachgestellten ‚H', wie es beim 8080-Assembler üblich ist, dargestellt. Eine ausführliche Behandlung des hexadezimalen (sedezimalen) Zahlensystems fand bereits im Band 1, Kapitel 20.0 ff statt).

Allgemein setzt CP/M einen 48 kByte großen freien Arbeitsspeicher voraus (auch Kernspeicher genannt). Leider sind beim CPC-464/664 im CP/M-Betrieb nur noch rund 40 kByte frei, so daß die Lauffähigkeit von größeren Programmen nur bedingt möglich ist. Durch eine RAM-Erweiterung, die die Firma Schneider anbieten will (inzwischen von anderen Firmen bereits angeboten wird), läßt sich dieser Mangel beheben.

Auch das sehr ungewöhnliche Diskettenformat mit 3 Zoll erschwert den Zugriff auf kommerzielle CP/M-Programme. Außerdem sind gute (kommerzielle) Programme (im Verhältnis zum CPC) sehr teuer. Es bleibt abzuwarten, ob durch den Einzug von CP/M im Homecomputerbereich die Preise fallen.

CP/M wurde 1973 von dem Softwareberater der Fima Intel, Gary Kildall entwickelt. Seither sind viele verbesserte Versionen entstanden. Die erste fehlerfreie Version war das CP/M 1.4. Die Ihnen vorliegende Version 2.2 ist eine aktuelle und weiter verbesserte Version. Seit Beginn der 80ziger Jahre ist CP/M eines der verbreitesten Betriebssysteme überhaupt. Auf Grund der großen Verbreitung gibt es heute angepaßte Systeme für fast jeden Prozessortyp.

1.1 CP/M laden

Jedesmal wenn Sie mit dem CP/M-Betriebssystem arbeiten wollen, müssen Sie zunächst das CP/M von der Diskette in den Arbeitsspeicher des CPCs laden. Dazu legen Sie entweder die Systemdiskette (Kopie) oder eine Diskette mit den CP/M-Systemspuren in das Laufwerk. Arbeiten Sie mit zwei Laufwerken (Diskettenstationen), muß die Systemdiskette in das Laufwerk eingelegt werden, welches direkt am CPC angeschlossen oder eingebaut ist (CPC-664). Dieses Laufwerk nennt sich Laufwerk A. Das Laden und Starten des Betriebssystems findet dann mit dem Kommando

```
|CPM
```

statt. Der senkrechte Strich „|“ erscheint durch das gleichzeitige Betätigen der „@“- und der SHIFT-Taste. Nach kurzer Ladezeit sollte die Meldung

```
CP/M 2.2 Amstrad Consumer Electronics plc
```

in schwarzer Schrift auf pastellblauem Hintergrund erscheinen. Erscheint statt dieser Meldung die Fehlermeldung

```
Failed to load boot sector
Retry, Ignore or Cancel ?
```

konnte der Rechner die Systemspuren auf der Diskette nicht finden. Daraus

ist zu schließen, daß sich entweder keine Systemspur auf der eingelegten Diskette befindet oder aber diese Spur defekt ist.
Jetzt fragt das System, ob dieser Vorgang

(R)etry : wiederholt,
(I)gnore : ignoriert oder
(C)ancel : abgebrochen

werden soll. Mit der Betätigung der Tasten R, I oder C wird der Ladevorgang entsprechend wiederholt, ignoriert oder abgebrochen. Auf diese (Fehler-) Meldung werden Sie beim Arbeiten unter CP/M immer dann treffen, wenn eine Aufgabe nicht ordnungsgemäß ausgeführt werden konnte. Legen Sie jetzt eine Diskette mit den Systemspuren in das Laufwerk und wiederholen den Ladevorgang (R). Nach ca. 4 Sekunden sollte dann die CP/M-Einschaltmeldung (siehe oben) erscheinen. Dieser Ladevorgang nennt sich auch „booten".

Nach erfolgreichem Booten des CP/M-Betriebssystems erscheint weiterhin ein sogenanntes Prompt-Zeichen auf dem Bildschirm

```
A>_
```

und meldet damit die Bereitschaft, ein Kommando aufzunehmen. Der Buchstabe „A" gibt Auskunft darüber, welches Laufwerk zur Zeit ausgewählt ist. In diesem Fall beziehen sich alle Kommandos auf das Laufwerk A (Hauptlaufwerk). Mit dem Erscheinen des Prompt-Zeichens kann davon ausgegangen werden, daß das Booten erfolgreich war.

1.2 Kommandoeingabe

Nach dem Erscheinen des Prompt-Zeichens wartet das CP/M-System auf die Eingabe eines Kommandos. Im wesentlichen gliedern sich die CP/M-Kommandos in zwei Gruppen

1. Eingebaute Kommandos
2. Flüchtige (transiente) Kommandos

Nach erfolgreichem Booten von CP/M 2.2 stehen 7 Kommandos, die fest in das CP/M integriert sind, zur Verfügung. Diese 7 Kommandos können jederzeit, ohne weiteres Nachladen von der Diskette, ausgeführt werden. Sie bilden das zentrale Werkzeug für die komfortable Bedienung des CP/M-Betriebssystems.

Flüchtige (transiente) Kommandos sind im Gegensatz zu den eingebauten Kommandos Programme, die erst von der Diskette in den Arbeitsspeicher geladen werden müssen. Bei den nachgeladenen Programmen braucht es sich nicht um Dienstprogramme wie FORMAT, STAT, PIP etc. zu handeln, sondern es können auch sonstige unter CP/M lauffähige Programme sein (Textverarbeitung, Kalkulation etc.). Ein solches CP/M-Programm (LOGO) befindet sich auf der B-Seite der Systemdiskette. Nach dem Laden eines Programm, starten sich diese dann selber (Autostart).

Nach dem Prompt-Zeichen ist es möglich, jedes beliebige Kommando einzugeben und mit der ENTER-Taste abzuschließen (Abschluß eines Kommandos ist auch mit <CTRL>-J oder <CTRL>-M möglich).

```
A>DIR<cr>
```

Dabei ist es unwichtig, ob Sie das Kommando groß und/oder klein eingeben (Dir, DIr, dir etc.), da das CP/M alle Zeichen selbsttätig in Großbuchstaben umwandelt.

Geben Sie ein dem CP/M nicht bekanntes Kommando ein, gibt das CP/M das unbekannte Kommando mit einem nachgestellten Fragezeichen (?) erneut als Fehlermeldung aus

```
A>DRI<cr>
DRI?
A>_
```

Grundsätzlich prüft das CP/M zunächst, ob es sich um eines der 7 eingebauten Kommandos handelt. Ist dies nicht der Fall, wird geprüft, ob sich auf der Diskette im aktuellen Laufwerk ein Programm unter diesem Namen befindet. Befindet sich auch auf der Diskette kein Programm unter diesem Namen oder es ist kein unter CP/M ausführbares Programm (kein .COM-File), meldet sich CP/M mit der oben genannten (Fehler-) Meldung.

Liegt ein korrektes CP/M-Kommando vor, wird es entweder sofort (eingebaute Kommandos) oder nach kurzer Ladezeit von der Diskette ausgeführt.

Die 7 eingebauten CP/M-Kommandos (d:, DIR, ERA, REN, SAVE, TYPE und USER) sind sehr einfache Routinen, die im wesentlichen den komfortablen Umgang mit den angeschlossenen Laufwerken (Diskettenstationen) ermöglichen. Alle eingebauten Kommandos beziehen sich immer auf eines der Laufwerke. Damit ein bestimmtes Programm auf der Diskette wiedergefun-

den werden kann, muß es einen eindeutigen Namen tragen. Wie vom «normalen« BASIC-Betrieb der Schneider-Computer gewohnt, legt auch das CP/M seine Programme (Files) auf der Diskette unter einem 8-stelligen Filenamen mit einer durch den Dezimalpunkt getrennten 3-stelligen Typenkennzeichnung (Extention) ab. Grundsätzlich ist bei allen Kommandos immer die vollständige Filebezeichnung anzugeben

A>kommando filename.typ

Beim Laden eines unter CP/M lauffähigen Programms (.COM-Datei) kann die Angabe des Typs entfallen, da das CP/M hierbei grundsätzlich den Typ „.COM" annimmt

A>filename = A>filename.COM

Ebenfalls ist die Benutzung der sogenannten Wildcards (Referenzzeichen) auch bei einigen CP/M-Kommandos und Programmen zulässig. Bei allen Kommandos und Dienstprogrammen wird darauf im folgenden gesondert hingewiesen.

Alle diejenigen unter Ihnen, denen die Aufgabe und Funktion der Filetyp-Kennzeichnung und die Benutzung der Wildcards nicht (mehr) geläufig ist, finden im 2. Band der CPC-Reihe in den Abschnitten 5 und 6 eine umfassende Erläuterung.

1.3 Tastatur unter CP/M

Damit eine eingegebene Kommandozeile auch beliebig korrigiert werden kann, verfügt CP/M über eine Reihe von Steuertasten, die eine komplexe Funktion auslösen (z.B. Kommandozeile löschen). Sie sind alle in Kombination mit der CONTROL-Taste (CTRL) und einer anderen Taste ansprechbar. Im folgenden wird das gleichzeitige Betätigen der Control- und einer anderen Taste (hier mit der Taste ‚C') mit

Control-C = ^C

abgekürzt. Einige Kommandos können jederzeit angewandt werden, andere müssen Bestandteil einer Kommandozeile sein und mit einem Return der ENTER-Taste geschlossen werden. Diese Kommandos sind im folgenden mit <cr> gekennzeichnet.

Der erste Teil dieser Liste faßt alle Standard-Steuerzeichen des CP/M 2.2 zusammen. Sie bieten wesentliche Erleichterungen beim Edieren einer Kommandozeile. Im zweiten Teil folgen dann alle CPC speziellen Steuerzeichen, die unter CP/M nutzbar sind. Mit diesen speziellen Steuerzeichen (Steuertasten) lassen sich z.B. die Farben ändern, der Bildschirm löschen, ja sogar Fenster (WINDOWs) generieren (eine ausführliche Beschreibung aller CPC-Steuerzeichen finden Sie auch im Band 1, Kap. 16.0). Die Benutzung der speziellen Steuerzeichen bietet natürlich einige Vorteile, sollten aber auf Grund der Kompatibilität zu anderen CP/M-fähigen Computern nur sehr sparsam angewandt werden.

1.4 Standard Steuerzeichen unter CP/M

Die folgenden Standard-Steuerzeichen sind die elementaren Möglichkeiten, um eine Kommandozeile beliebig zu edieren. Probieren Sie diese Steuerzeichen in allen Kombinationen aus.

^C Warmstart (warm boot)
Bricht das laufende Kommando/Programm (z.B. DDT) ab und kehrt zur Kommandoebene zurück. Dabei wird das eigentliche CP/M erneut von der Diskette geladen, allerdings ohne die Daten des Arbeitsspeichers (TPA) zu zerstören.

^E Setzt die aktuelle Kommandozeile in der nächsten Bildschirmzeile fort. Dabei entsteht aus den Teileingaben ein Kommando:

```
A>D^E
I^E
R<cr>     (entspricht A>DIR<cr>)
```

^H Löscht das Zeichen links vom Cursor.

^J Line Feed oder

^M Carriage Return
Beide Steuerzeichen bewirken, daß eine eingegebene Kommandozeile abgeschlossen und dem CP/M übergeben wird (Enter). Beide Steuerzeichen können mit der ENTER-Taste ersetzt werden <cr>.

^P Leitet die Ausgabe auf den Drucker um.
Dabei arbeitet ^P als ein Umschalter zwischen Drucker und

Bildschirm bzw. umgekehrt. Bei der Druckerausgabe werden die Informationen zusätzlich noch auf dem Bildschirm ausgegeben (Echo).

^R Zeigt die aktuelle Kommandozeile erneut komplett an und faßt dabei alle durch ^E entstandenen Teilzeilen zu einer Zeile zusammen. Es erscheint ein Nummernzeichen (#) am Ende der (Teil-) Zeile.

```
A>D^E
I^E
R^R#
DIR
```

^S Unterbricht die laufende Anzeige und wartet mit der Fortführung der Anzeige bis zur Betätigung irgendeiner Taste.

```
A>DIR<cr>
A : MOVCPM   COM : TESTPROG COM : (^S)_
```

^U Bricht die aktuelle Kommandozeile ab, geht in eine neue Zeile und kennzeichnet die gelöschte Zeile mit einem Nummernzeichen (#)

```
A>stst *.*^U#
```

^V<cr> Bricht die aktuelle Kommandozeile ab, gibt das Kommando noch einmal aus und wartet auf eine Neueingabe.

```
A>hallo^V
HALLO
```

^X Löscht die eingegebene Kommandozeile und der Cursor erscheint wieder am Anfang derselben Zeile.

1.5 Spezielle Steuerzeichen des CPCs

Alle speziellen Steuerzeichen führen eine besondere, nicht im CP/M vorgesehene Funktion aus. Generell wird nach der Ausführung ein Fragezeichen in der nächsten Bildschirmzeile als Fehlermeldung des CP/M ausgegeben (kein zulässiges CP/M-Kommando). Einige spezielle Steuerzeichen benötigen mehrere Parameter, um ihre Funktion korrekt auszuführen. Dazu folgen nach dem Steuerzeichen weitere Zeichen, die jeweils den Wert ihres ASCII-Wertes darstellen.

Alle hier beschriebenen Steuerzeichen sind Bestandteil der Kommandozeile und müssen mit der ENTER-Taste abgeschlossen werden. Der ASCII-Wert jedes Zeichens steht in Klammern vor der allgemeinen Beschreibung. Beachten Sie bitte, daß diese Steuerzeichen nur auf dem CPC diese Wirkungen erzielen. Bei anderen Computern können dies andere Funktionen sein!! Ebenfalls sollten Sie darauf achten, daß sie nicht in Konflikt mit den Standard-Steuerzeichen kommen.

^@ (0) Hat keine Funktion und wird ignoriert.

^A<cr> (1) Alle nachfolgenden Steuerzeichen führen nicht ihre Funktion aus, sondern es wird nur ihr Grafiksymbol ausgegeben (ASCII 0 bis 31).

^B<cr> (2) Schaltet den Cursor aus. Er bleibt bis zu einem ^F inaktiv

```
A>^B<cr>
A>
```

^C (3) Siehe Standard-Steuerzeichen.

^D (4) Setzt den Bildschirmmodus (80, 40, 20 Zeichen) auf den folgenden Parameter

`A>^D@<cr>`	MODE 0 (20 Zeichen)
`A>^Da<cr>`	MODE 1 (40 Zeichen)
`A>^Db<cr>`	MODE 2 (80 Zeichen)

Die Parameter „@", „a" und „b" können benutzt werden, da nur die niederwertigen 2 Bits geprüft werden (010000 00 = 0).

^E (5) Siehe Standard-Steuerzeichen.

^F<cr> (6) Schaltet den Textcursor wieder an (siehe ^B).

^G<cr> (7) Gibt einen kurzen Ton aus (BELL).

^H (8) Siehe Standard-Steuerzeichen.

^I (9) Siehe Standard-Steuerzeichen.

^J (10) Siehe Standard-Steuerzeichen.

^K<cr> (11) Setzt den Cursor auf Zeilenanfang. In der Praxis nicht anwendbar.

^L<cr> (12) Löscht den Bildschirm (entspricht CLS)

```
A>^L<cr>
```

^M (13) Siehe Standard-Steuerzeichen.

^N<cr> (14) Setzt die PAPER-Farbe auf den nachfolgenden Parameter (entspricht PAPER)

A>^N@<cr>	PAPER 0
A>^Na<cr>	PAPER 1
A>^Nb<cr>	PAPER 2
.....	.....
A>^No<cr>	PAPER 15

^O<cr> (15) Setzt die PEN-Farbe auf den nachfolgenden Parameter (entspricht PEN)

A>O@<cr>	PEN 0
A>Oa<cr>	PEN 1
A>Ob<cr>	PEN 2
.....	....
A>Oo<cr>	PAPER 15

^P (16) Siehe Standard-Steuerzeichen.

^Q<cr> (17) Löscht die Kommandozeile von Cursorposition bis Zeilenanfang (Kommando ignorieren)

```
A>stat^Q<cr>
   ?
```

^R (18) Siehe Standard-Steuerzeichen.

^S<cr> (19) Löscht den Bildschirm von der ersten bis zur aktuellen Cursorzeile (siehe auch Standard-Steuerzeichen).

^T<cr> (20) Löscht den Bildschirm von der aktuellen Cursorzeile bis zur letzten Bildschirmzeile

```
A>^T<cr>
```

^U (21) Siehe Standard-Steuerzeichen.

^V<cr> (22) Schaltet den Transparentmodus ein oder aus (siehe auch Standard-Steuerzeichen)

```
A>^V@<cr>        Transparentmodus aus.
A>^Va<cr>        Transparentmodus an.
```

^W<cr> (23) Ignoriert die Kommandozeile.

^X (24) Siehe Standard-Steuerzeichen.

^Y (25) Löscht Kommandozeile und erwartet Neueingabe.

^Z<cr> (26) Definiert die Größe des Fensters 0 (WINDOW #0). Es folgen 4 Parameter, die den Spaltenanfang, Spaltenende, Zeilenanfang und Zeilenende bestimmen. Diese Werte dürfen die Größe des Bildschirms nicht überschreiten. Die folgenden 4 ASCII-Zeichen repräsentieren ihren ASCII-Code, die der Funktion als Wert übergeben wird. D.h., sie führen (bis auf die Standard-Steuerzeichen !!!) nicht ihre Funktion aus

```
A>^Z^A(^T^Y <cr>
```

(entspricht: WINDOW #0,1,40,20,25)

^[<cr> (27) Bricht aktuelle Kommandozeile ab.

^\<cr> (28) Füllt Farbfaß (1.Parameter) mit den Werten der folgenden Parameter (entspricht INK)

```
A>^\@aa<cr>      INK 0,1,1
A>^\azz<cr>      INK 1,26,26
(@=0, a=1, b=2   .... z=26)
```

^]<cr> (29) Setzt die BORDER-Farbe auf die folgenden zwei Parameter (entspricht BORDER)

```
A>^]@z<cr>       BORDER 0,26
```

(Border blinkt in Schwarz (0) und Weiß (26))

^^<cr> (30) Setzt den Cursor in die obere linke Ecke des Bildschirms (Fensters).

^0<cr> (31) Setzt den Cursor auf den angegebenen Spalten- und Zeilenwert (entspricht LOCATE)

```
A>^0^A^L<cr>     LOCATE (#0,) 1, 12
```

1.6 CP/M-Kommandos

Das CP/M bietet zunächst nur 7 (eingebaute) Kommandos, die jederzeit ohne weiteres Nachladen von der Diskette ausführt werden können. Es sind, wie schon erwähnt, nur sehr einfache Kommandos, die im wesentlichen den Umgang mit den angeschlossenen Laufwerken ermöglichen. Es sind die Kommandos

d:	Wechseln des aktuellen Laufwerks	(Kap. 2.1)
DIR	Anzeigen des Inhaltsverzeichnis der Diskette	(Kap. 2.2)
ERA	Löschen von Dateien (Files) auf der Diskette	(Kap. 2.3)
REN	Umbenennen einer Datei	(Kap. 2.4)
TYPE	Listen einer Datei mit ASCII-Informationen	(Kap. 2.5)
USER	Wechseln der Benutzerebene	(Kap. 2.6)
SAVE	Speicherraum auf Diskette sichern	(Kap. 2.7)
filename	Laden und Starten einer COM-Datei	

Mehr kann das CP/M 2.2 zunächst einmal nicht. Diese Kommandos bilden das zentrale Werkzeug für die Arbeit unter CP/M. Alle weiterführenden Aufgaben (z.B. das Formatieren) können nur CP/M-Programme übernehmen. Es sind Dienstprogramme, die wie eingebaute Kommandos benutzt werden können. Diese Dienstprogramme liegen als COM-Dateien (ein unter CP/M lauffähiges Programm) auf der Diskette und werden von dieser in den Arbeitsspeicher des CPCs geladen und dann gestartet. Schneider bietet auf der CP/M-Systemdiskette insgesamt 22 Dienstprogramme

FORMAT	Formatieren einer Diskette	(Kap. 3.1)
CLOAD	Datei von Kassette auf Diskette kopieren	(Kap. 3.2)
CSAVE	Datei von Diskette auf Kassette kopieren	(Kap. 3.3)
FILECOPY	Dateien im Laufwerk A kopieren	(Kap. 3.4)
DISCCOPY	Gesamte Diskette im Laufwerk A kopieren	(Kap. 3.5)
COPYDISC	Diskette von Laufwerk A nach B kopieren	(Kap. 3.6)
PIP	Kopier- und Übertragungsprogramm	(Kap. 3.7)
DISCCHK	Vergleich von 2 Disketten im Laufwerk A	(Kap. 3.8)
CHKDISC	Vergleich der Diskette im Laufwerk A mit B	(Kap. 3.9)
STAT	Umfangreiche Disketten-/Systemstatistik	(Kap. 3.10)
SUBMIT	Automatische Kommandoausführung	(Kap. 3.11)
XSUB	Automatische Benutzereingabe	(Kap. 3.12)
ED	Einfacher Texteditor	(Kap. 3.13)
ASM	8080-Assembler	(Kap. 3.14)
LOAD	CP/M-Programme generieren (HEXnachCOM)	(Kap. 3.15)

DUMP	Hexadezimale Ausgabe eines Files	(Kap. 3.16)
DDT	CP/M-Monitor (Assembler und Disassembler)	(Kap. 3.17)
SETUP	Ändern des Konfigurationssektors	(Kap. 3.18)
MOVCPM	CP/M an Speichergröße anpassen	(Kap. 2.19)
SYSGEN	Systemspuren kopieren	(Kap. 3.20)
BOOTGEN	Konfigurationssektor kopieren	(Kap. 3.21)
AMSDOS	Rückkehr ins BASIC	(Kap. 3.22)

Auf den nachfolgenden Seiten finden Sie eine Beschreibung aller CP/M-Kommandos und Dienstprogramme. Versuchen Sie die Kommandos und Dienstprogramme in allen ihren Möglichkeiten auszuprobieren. In den einzelnen Beschreibungen finden Sie dazu einige der wesentlichen Fehlermeldungen der Kommandos oder Dienstprogramme, die bei speziellen Bedingungen auftreten können.

Neben diesen programmbedingten Meldungen und Fehlermeldungen verfügt das CP/M-System über eine Reihe von allgemeinen Fehlermeldungen. Diese Fehlermeldungen können jederzeit auftreten und zeigen einen fehlerhaften Zugriff auf die Diskette an. Bevor Sie die einzelnen Kommandos und Dienstprogramme ausprobieren, sollten Sie sich über die Bedeutung dieser allgemeinen Fehlermeldungen im klaren sein.

1.7 Allgemeine Fehlermeldungen

kommando?

Eine unkorrekte oder unverständliche Kommandozeile wird vom CP/M erneut mit einem nachgestellten Fragezeichen ausgegeben. Allgemein tritt dieser Fehler auf, wenn es sich bei dem Kommando weder um ein eingebautes Kommando noch um eine auf dem angesprochenen Laufwerk vorhandene COM-Datei handelt.

DRIVE d:DISC MISSING

Das Laufwerk „d" ist entweder nicht (korrekt) angeschlossen oder es befindet sich keine Diskette im Laufwerk. Weiterhin wird die Meldung

```
Retry, Ignore or Cancel?
```

ausgegeben. Durch Betätigung des jeweiligen Anfangsbuchstaben können Sie den Vorgang

R = wiederholen
I = ignorieren oder
C = abbrechen.

Nach Betätigung der Taste „C" (Cancel) erscheint zusätzlich eine der folgenden Meldungen.

BDOS ERR ON d:SELECT

Das angesprochene Laufwerk „d" ist entweder nicht vorhanden oder es befand sich keine Diskette im Laufwerk. Nach Betätigung der ENTER-Taste kehrt das System wieder in die Kommandoebene zurück.

BDOS ERR ON d:BAD SECTOR

Bei dem Zugriff auf das Laufwerk „d" ist das System auf einen nicht lesbaren Sektor gestoßen.

BDOS ERR ON d:R/O

Es wurde versucht, eine schreibgeschützte Diskette im Laufwerk „d" zu beschreiben. Dieser Fehler tritt auch dann auf, wenn die Diskette gewechselt wurde!!

2.0 Eingebaute CP/M-Kommandos

Die nachfolgenden 7 Kommandos bilden das zentrale Werkzeug zur Handhabung und Benutzung des CP/M-Betriebssystems. Sie ermöglichen im wesentlichen den komfortablen Zugriff auf die angeschlossenen Laufwerke (Diskettenstationen). Sie werden immer wieder mit diesen Kommandos konfrontiert werden, so daß eine detaillierte Kenntnis (zumindest) dieser Kommandos umbedingt für ein zügiges und selbstverständliches Arbeiten unter CP/M notwendig ist. Nehmen Sie sich bitte die Zeit und erarbeiten Sie diese Kommandos vollständig.

Alle folgenden Kommandos werden sofort, ohne ein Zuladen von der Diskette ausgeführt. Die Beschreibungen dieser Kommandos und der in Kapitel 3 folgenden Dienst- und Hilfsprogramme haben folgenden Aufbau

1) Allgemeiner Aufruf.
2) Liste der möglichen Aufrufformate.
3) Allgemeine Beschreibung der Funktion und der Arbeitsweise des Kommandos.
4) Beschreibung der unter 2) aufgeführten Formate.
5) Fehlermeldungen. Auflistung und Beschreibung der wichtigsten Fehlermeldungen, sowie deren Ursachen.
6) Beispiel(e). Alle Beispiele wurden anhand der Systemdiskette durchgeführt. Arbeiten Sie immer mit einer Kopie (!!!) der CP/M-Systemdiskette, damit Sie sich Ihr Original nicht versehentlich zerstören!

2.1 Laufwerk wechseln (d:)

A<d:

Format: 1) d:

Ändert das aktuelle Laufwerk auf das durch ‚d:' bezeichnete Laufwerk. Obwohl das CP/M 2.2 es grundsätzlich ermöglicht, bis zu 16 verschiedene Diskettenstationen (Festplatten) anzusprechen, können beim CPC-464/664 nur zwei verschiedene Laufwerke angewählt werden. ‚d' kann nur die Laufwerke ‚A' und ‚B' ansprechen. Nach dem Starten von CP/M beziehen sich alle Operationen immer auf das Laufwerk ‚A'.

1. Wechselt das Laufwerk auf das durch ‚d:' angegebene.

Fehlermeldungen:

Bdos Err On d: Select
: Laufwerkangabe größer als ‚B'

DRIVE B:Disc Missing
: Laufwerk ‚B' nicht angeschlossen

Beispiel:

```
A>B:
B>
```

2.2 Inhaltsverzeichnis anzeigen (DIR)

```
A>DIR <laufwerk:> <filename.typ>
```

Format:
1) DIR
2) DIR d:
3) DIR *.typ
4) DIR filename.*
5) DIR test????.typ
6) DIR d:start?.*
7) DIR *.*

Der DIR-Befehl (directory=Verzeichnis) zeigt die Filenamen (Programme, Daten) an, die sich auf der Diskette befinden. Im Gegensatz zum CAT-Befehl des AMSDOS (siehe Band 2, Kap. 2.4) wird hier nicht die Größe des Files mit angezeigt.

1. Listet das gesamte Inhaltsverzeichnis des aktuellen Laufswerks.
2. Listet das Inhaltsverzeichnis des mit ‚d:' bezeichneten Laufwerks. ‚d' kann eines der angeschlossenen Laufwerke A oder B sein. Kombinationen wie

   DIR d:*.typ oder
   DIR d:filename.* etc.

   sind zulässig.
3. Listet alle auf der Diskette vorhandenen Filenamen vom Dateityp ‚typ' auf.
4. Listet alle Dateien mit Namen ‚filename' auf, unabhängig vom Typ.
5. Listet alle 8-stelligen mit ‚test' beginnenden Filenamen vom Dateityp ‚typ' auf.
6. Listet alle 6-stelligen mit ‚start' beginnenden Filenamen unabhängig vom Typ auf, die sich auf dem mit ‚d:' bezeichneten Laufwerk befinden.
7. Entspricht dem Kommando DIR (gesamtes Inhaltsverzeichnis zeigen, entspricht auch ‚DIR ???????.???').

Fehlermeldungen:

File not found

Diskette enthält den angegebenen File nicht.

No File

Die gewünschten Files befinden sich nicht auf der Diskette.

Beispiel:

```
A>DIR *.BAS
A: EX1      BAS : EX2      BAS
A>
```

2.3 File(s) löschen (ERA)

```
A>ERA <laufwerk:>filename.typ
```

Format:
1) ERA filename.typ
2) ERA d:filename.typ
3) ERA filename.*
4) ERA file????.*
5) ERA *.typ
6) ERA *.*

Mit dem ERA-Befehl (erase=löschen) können Files auf der Diskette gelöscht werden (Er kennzeichnet das 1. Byte des entsprechenden Directory-Eintrags mit dem Wert 0E5H, siehe 7.3). Ein versehentlich gelöschtes Programm kann mit dem DISCMON (siehe 7.4) gerettet werden, indem das 1. Byte des gelöschten Eintrags mit dem Wert 00H gefüllt wird (USER-Ebene 0).

1. Löscht den mit ‚filename.typ' bezeichneten File auf dem aktuellen Laufwerk.
2. Löscht den mit ‚filename.typ' bezeichneten File auf dem mit ‚d' bezeichneten angeschlossenen Laufwerk. Kombinationen wie

   ERA d:filename.* oder
   ERA d:file????.* etc.

   sind zulässig.
3. Löscht alle Files mit dem Namen ‚filename' unabhängig vom Typ.
4. Löscht alle 8-stelligen Files, deren Namen mit ‚file' beginnt, unabhängig vom Typ.
5. Löscht alle Files vom Dateityp ‚typ'.
6. Die gesamte Diskette wird gelöscht. Nach der Nachricht

   ALL FILES (Y/N)?

   können Sie das Kommando mit ‚Y' bestätigen (alle Files werden gelöscht) oder mit ‚N' das Löschen verhindern.

Fehlermeldungen:

File not Found

Diskette enthält die angegebenen Files nicht.

Beispiel:

```
A>ERA *.*
ALL (Y/N)? N
A>
```

2.4 File umbenennen (REN)

A>REN <laufwerk:>neuname.typ=altname.typ

Format: 1) REN neuname.typ=altname.typ
2) REN d:neuname.typ=altname.typ

Das Umbenennen eines Filenamens ist mit Hilfe des REN-Kommandos (rename=umbenennen) möglich. Die Namen der Files müssen immer eindeutig sein, d.h. es dürfen keine Wildcards benutzt werden.

1. Umbenennen des Files ‚altname.typ' in ‚neuname.typ' auf dem aktuellen Laufwerk.
2. Umbenennen des Files ‚altname.typ' in ‚neuname.typ' auf dem Laufwerk ‚d'.

Fehlermeldungen:

No File
File (‚altname.typ') befindet sich nicht auf der angesprochenen Diskette.

File exists
Filename (‚neuname.typ') ist bereits auf der Diskette vorhanden.

Beispiel:

```
A>REN NEU.BAS=MONITOR.BAS
A>
```

2.5 Listen einer ASCII-Datei (TYPE)

A>TYPE <laufwerk:>filename.typ

Format: 1) TYPE filename.typ
2) TYPE d:filename.typ

TYPE (type=Ausdruck) ermöglicht den Ausdruck einer ASCII-Datei auf dem Bildschirm bzw. Drucker. Handelt es sich bei dem ausgewählten File um keine ASCII-Datei (Texte, Daten, etc.), können unerwünschte Nebenwirkungen auftreten (Bildschirmaufbau, Farben, Programmende, etc.).

1. Listet die ASCII-Datei ‚filename.typ', die sich auf dem aktuellen Laufwerk befindet, auf dem Bildschirm (Drucker) aus.
2. Listet die ASCII-Datei ‚filename.typ' vom Laufwerk ‚d:' auf dem Bildschirm (Drucker) aus.

Fehlermeldungen:

filename.typ?
Gewünschte ASCII-Datei befindet sich nicht auf der aktuellen Diskette (auch bei der Benutzung von Wildcards).

Beispiel:

```
A>TYPE dump.asm
;     FILE DUMP PROGRAM, READ AND INPUT FILES AND PRINTS
      IN HEX
;     COPYRIGHT (C) 1975, 1976, 1977, 1978
;
;     DIGITAL RESEARCH
;     BOX ........
```

2.6 Wechseln der Benutzerebene (USER)

```
A>USER x
```

Format: 1) USER x

Bis zu 16 verschiedene Benutzer (user=Benutzer) können sich eine Diskette miteinander teilen. ‚x' kennzeichnet die gewünschte Benutzerebene (0-15). Nach dem Start des CP/M-Betriebssystems befindet man sich in der Benutzerebene 0. Alle Befehle beziehen sich immer auf die aktuelle Benutzerebene (DIR, REN, ERA, etc.).

1. Umschalten auf die Benutzerebene ‚x'.

Fehlermeldungen:

x?
: ‚x?' gibt eine unzulässige USER-Nummer wieder.

Beispiel:

```
A>USER 7
A>DIR
NO FILE
A>
```

2.7 Speicherraum auf Diskette sichern (SAVE)

A>SAVE page <laufwerk:>filename.typ

Format: 1) SAVE page filename.typ
2) SAVE page d:filename.typ

Ein bestimmter Speicherraum, beginnend mit 0100H (TPA, Arbeitsspeicher) kann auf die Diskette unter dem Namen ‚filename.typ' abgelegt werden (save=sichern). ‚page' gibt dabei die Länge in 256 Byte-Blöcken an (256 Bytes = 1 Page). Um z.B. das erste kByte der TPA zu sichern (0100H bis 0500H), müssen die ersten 4 Seiten (4*256=1024= 1 kByte) angegeben werden. Da ein unter CP/M aufrufbares Maschinenprogramm gesichert wird, bietet sich der Typ ‚.COM' an.

1. Sichert den Speicherraum ab 0100H mit einer Länge von ‚page'*256 Bytes unter dem Namen ‚filename.typ' auf dem aktuellen Laufwerk.
2. Sichert den Speicherraum ab 0100H mit einer Länge von ‚page'*256 Bytes unter dem Namen ‚filename.typ' auf dem Laufwerk ‚d:'.

Fehlermeldungen:

filename?
Unzulässige Benutzung von Wildcards oder die Seitenangabe (‚page') fehlt.

Disc Full
Auf der benutzten Diskette befindet sich weniger Platz, als für die Sicherung notwendig ist.

Beispiel:

```
A>SAVE 4 TEST.COM
A>DIR TEST.COM
A : TEST    COM
A>
```

3.0 Dienstprogramme (transiente Kommandos)

Neben den bisher beschriebenen 7 CP/M-Kommandos befinden sich auf der Systemdiskette weitere 22 Dienst- bzw. Hilfsprogramme, die ähnlich gehandhabt werden. Ein Großteil der Programme sind Kopier- und Verifyprogramme (DISCCOPY, PIP, CHKDISC, etc.), Programme zum Anpassen des CP/M-Betriebsystems (SETUP, MOVCPM), sowie einige Programme zum Arbeiten und Programmieren unter CP/M (ED, DDT, DUMP, ASM, etc.). Einige Dienst- bzw. Hilfsprogramme sind sehr umfangreich und können daher an dieser Stelle in ihrer Komplexität nicht immer vollständig erläutert werden. Der Aufbau der im folgenden beschriebenen Dienst- und Hilfsprogramme ist identisch mit dem im Kapitel 2.0 beschriebenen Aufbau.

Bei den Fehlermeldungen wurden nur die in der alltäglichen Praxis häufig auftretenden Meldungen berücksichtigt (PIP hat z.B. weit über 40 Fehlermeldungen).

Die Beispiele beziehen sich durchweg auf die Systemdiskette. D.h. die Beispiele zeigen die Auswirkungen und die Ergebnisse bei eingelegter CP/M-Systemdiskette. Arbeiten Sie deshalb zur Vorsicht immer nur mit einer Kopie der Systemdiskette (siehe DISCCOPY, 3.5).

An dieser Stelle möchte ich Ihnen empfehlen, alle Dienst- bzw. Hilfsprogramme einmal in den verschiedenen Variationen auszuprobieren. Gerade so komplexe Programme wie z.B. PIP bieten eine Unzahl von interessanten Möglichkeiten. Ein Experiment sagt oftmals mehr als viele Worte.

3.1 Formatieren einer Diskette (FORMAT)

A>FORMAT <formtyp>

Format: 1) FORMAT
2) FORMAT x

Mit dem FORMAT-Programm (format=formatieren) wird eine Diskette formatiert (siehe 1.0). Beim Formatieren einer beschriebenen Diskette werden alle Daten zerstört!! Gleichzeitig unterstützt die Schneider Diskettenstation DDI-1 auch andere Diskettenformate (siehe weiter unten und Band 2, 10.5).

1. Formatiert die im aktuellen Laufwerk befindliche Diskette im Standard-(AMS)-Format (40 Spuren/9 Sektoren). Gleichzeitig wird das CP/M-Betriebssystem in die Spuren 0 und 1 kopiert.
2. Formatiert die im aktuellen Laufwerk befindliche Diskette in dem durch ein Leerzeichen getrennten Format ‚x'. Dabei darf ‚x' folgende Kennungen tragen

D	=Daten-Format	(40 Spuren/9 Sektoren, ohne CP/M)
V	=Vendor-Format	(40 Spuren/9 Sektoren, ohne CP/M)
I	=IBM-Format	(40 Spuren/8 Sektoren)

Fehlermeldungen:

You must insert the disc to be formated into drive A
Keine Diskette im Laufwerk A

The disk to be formated in drive A must be write-enabled
Zu formatierende Diskette ist schreibgeschützt

Beispiel:

```
A>FORMAT

FORMAT V2.0

Please insert disc to be formatted into drive A then
press key:
```

```
Formating started
Formating track (0-39) complete
Do you want to format another disk (Y/N):N
Please insert a CP/M system disc into drive A then
press any key

FORMAT V2.0 finished
A>
```

3.2 File kopieren <Kassette - Diskette> (CLOAD)

```
A>CLOAD <''filename''>
```

Format: 1) CLOAD
2) CLOAD ''filename''

Ein File wird von der Kassette in den Arbeitsspeicher des Rechners geladen und anschließend auf die Diskette kopiert. Dabei ist zu beachten, daß der Filename des zu ladenden Files eindeutig sein muß (keine Wildcards). Die Länge des Filenamens darf 8 Zeichen (+ Typ) nicht überschreiten (siehe Fehlermeldungen). Der Filename muß in Anführungsstrichen stehen.

1. Lädt das nächste Programm von der Kassette und schreibt es auf die Diskette im aktuellen Laufwerk.
2. Lädt das Programm mit dem Namen ‚filename' von der Kassette und schreibt es auf die Diskette im aktuellen Laufwerk.

Fehlermeldungen:

Invalid CP/M filename
Kein korrekter CP/M-Filename. Der gefundene oder gewünschte Filename ist zu lang oder die Anführungsstriche fehlen.

Disc is write proteced
Diskette ist schreibgeschützt.

Cannot read protected cassette files
Kann kein geschütztes Kassettenfile kopieren!

Beispiel:

```
A>CLOAD "test.bas"

CLOAD V2.0
Press PLAY then any key:
Loading TEST block 1

CLOAD V2.0
A>
```

3.3 File kopieren <Diskette - Kassette> (CSAVE)

```
A>CSAVE filename.typ
```

Format: 1) CSAVE filename.typ

Ähnlich dem CP/M-Dienstprogramm CLOAD arbeitet dieses Dienstprogramm. Hier wird allerdings ein File von der Diskette auf die Kassette kopiert.

1) Lädt den File mit dem Name ‚filename.typ' vom aktuellen Laufwerk in den Arbeitsspeicher des Rechners und kopiert ihn anschließend auf die Kassette.

Fehlermeldungen:

Invalid CP/M filename
Kein korrekter CP/M-Filename

CP/M file does not exist
File nicht auf der Diskette vorhanden

Beispiel:

```
A>CSAVE ex1.bas

CSAVE V2.0

Press REC and PLAY then any key:
Saving EX1.BAS block 1

CSAVE finished
A>
```

3.4 File(s) kopieren <Diskette A - Diskette A> (FILECOPY)

```
A>FILECOPY filename.typ
```

Format:
1) FILECOPY filename.typ
2) FILECOPY *.typ
3) FILECOPY filename.*
4) FILECOPY file????.typ

Der oder die zu kopierenden Files werden vom Laufwerk A in den Arbeitsspeicher des Rechners geladen und nach Diskettenwechsel auf das Laufwerk A zurückgeschrieben. Werden Wildcards benutzt, kann aus einer Gruppe der möglichen Filenamen noch einzeln selektiert werden (siehe Beispiel 2).

1. Lädt den File mit dem Namen ‚filename.typ' vom Laufwerk A in den Arbeitsspeicher und kopiert ihn nach Tastenbetätigung auf das Laufwerk A zurück.

2.-4. Durch Benutzung der Wildcards ist eine Gruppe von Programmen angesprochen, die auf Wunsch einzeln bestätigt werden können (Y).

Ambiguous file name: Confirm individual file (Y/N)?

Es erscheinen alle in Frage kommenden Filenamen und es muß bestätigt werden, ob sie zu kopieren sind (Y) oder nicht (N). Es werden dann alle bestätigten Programme nacheinander kopiert (siehe Beispiel 2). Wünschen Sie keine Filebestätigung, werden alle in Frage kommenden Programme nacheinander kopiert.

Fehlermeldungen:

No SOURCE file present on disc
Gewünschtes (Quell-) File nicht auf der Diskette

No SOURCE file present on input line
Kein (Quell-) File in der Kommandozeile angegeben

Ambigous file name: Confirm individual file (Y/N)?
Mehrdeutiger Filename: Files einzeln bestätigen (J/N)?

Beispiel 1:

```
A>FILECOPY ex1.bas

FILECOPY V2.1

Please insert SOURCE disc into drive A then
press any key:
Copying started ....
Please insert DESTINATION disc into drive A then press
any key

EX1      .BAS Copied
Copying complete
A>
```

Beispiel 2:

```
A>FILECOPY *.bas

 FILECOPY V2.1

Please insert SOURCE disc into drive A then press any key

Ambiguous file name: Confirm individual file (Y/N)?Y

          EX1      .BAS  Copy (Y/N) ? N
          EX2      .BAS  COPY (Y/N) ? N

No file to copy
A>
```

3.5 Backup <Diskette A - Diskette A> (DISCCOPY)

A>DISCCOPY

Format: 1) DISCCOPY

Kopiert eine gesamte Diskette vom Laufwerk A auf das Laufwerk A zurück. Ist die Zieldiskette nicht oder anders als die Quelldiskette formatiert, wird sie gleichzeitig auf das Format der Quelldiskette formatiert.

1. Kopiert eine Diskette komplett im Laufwerk A. Nach Einlegen der Quelldiskette werden je 8 Spuren in den Arbeitsspeicher des Rechners geladen und nach dem Wechseln der Diskette zurückgeschrieben. Der Vorgang wiederholt sich 5 mal.

Fehlermeldungen:

You must insert the destination disc into drive A
Zieldiskette fehlt im Laufwerk A

You must insert the source disc into drive A
Quelldiskette fehlt im Laufwerk A

Source and destination discs have different formats
Quell- und Zieldiskette haben unterschiedliche Formate

Failed to write the destination disc correctly: track x sector y
Fehler im Track x, Sektor y aufgetreten

WARNING: Failed to copy disc correctly.
The destination disc should not be use until it is successfully
WARNUNG: Es sind Fehler beim Kopieren aufgetreten

Beispiel:

```
A>DISCCOPY

DISCCOPY V2.0

Please insert source disc into drive A then press any
key:
Copy started
Read track (0-39)
```

```
Please insert destination disc into drive A then press
any key:
Writing track (0-39)
```

..... dieser Vorgang wiederholt sich 5 mal !! .....

```
Copying complete

Do you want to copy another disc (Y/N)?N
A>
```

3.6 Backup <Diskette A - Diskette B> (COPYDISC)

```
A>COPYDISC
```

Format: 1) COPYDISC

COPYDISC ermöglicht das Kopieren einer gesamten Diskette vom Laufwerk A auf das Laufwerk B. Besitzen Sie zwei Laufwerke, ist dieses Dienstprogramm auf jeden Fall dem Programm DISCCOPY vorzuziehen, da es wesentlich schneller und komfortabler arbeitet.

1. Kopiert die Diskette im Laufwerk A komplett nach Laufwerk B. Dabei befindet sich die Quelldiskette im Laufwerk A.

Fehlermeldungen:

Siehe DISCCOPY (3.5)

Beispiel:

```
A>COPYDISC

COPYDISC V1.0

Please insert source disc into drive A and destination
disc into drive B then press any key:
Copy started
Read track (0-39)
Writing track (0-39)
Copying complete
Do you want to copy another disc (Y/N)?N
A>
```

3.7 Kopier- und Übertragungsprogramm (PIP)

A>PIP <laufwerk:>ziel=<laufwerk:>quelle<[p]>

Format:
1) PIP <d:>neufile.typ=<d:>altfile.typ[p]
2) PIP d:=d:*.*[p]
3) PIP <d:>neu.typ=<d:>alt1.typ[p],<d:>alt2.typ[p]
4) PIP dev:=<d:>filename.typ[p]

PIP (Peripheral Interchange Program) ist ein sehr umfangreiches und leistungsstarkes Übertragungs- und Kopierprogramm. Im Gegensatz zu den Dienstprogrammen FILECOPY, etc. ist mit PIP auch das Kopieren einer identischen Datei unter anderem Namen auf der Diskette möglich (siehe Beispiel 1). Durch viele Sonderfunktionen lassen sich Kopien auf dem Bildschirm, Drucker aber auch über eine serielle Schnittstelle (falls vorhanden) durchführen. Ein durch eckige Klammern eingeschlossener Parameter erlaubt das Kopieren mit bestimmten Eigenschaften. Weiterhin ist die Verkettung mehrerer Dateien möglich. Grundsätzlich besitzt das PIP-Kommando folgenden Aufbau

PIP ziel = quelle [parameter]

Die Angabe einer Laufwerkbezeichnung ist wahlfrei. Wird kein Laufwerk angegeben, beziehen sich die Angaben auf das aktuelle Laufwerk.

1. Kopiert den ‚altfile.typ' vom Laufwerk ‚d' nach Laufwerk ‚d' unter dem Namen ‚neufile.typ' (siehe Beispiele).
2. Kopiert alle Dateien vom Quellaufwerk ‚d' zum Ziellaufwerk ‚d'. Ein Kopieren auf dasselbe Laufwerk ist nicht möglich.
3. Verkettet die Dateien ‚alt1.typ' und ‚alt2.typ' von den Laufwerken ‚d'. Die Verkettung wird unter dem Namen ‚neu.typ' auf dem Laufwerk ‚d' abgelegt (siehe Beispiele).
4. Kopiert ‚filename.typ'auf die mit ‚dev:' bezeichnete Einheit (siehe Beispiele).

PIP-Parameter:

B	Blockübertragung. Normalerweise nicht notwendig.
Dn	Bei der Übertragung von zeilenorientierten Dateien werden die ab Spalte ‚n' folgenden Zeichen bis zum Zeilenende nicht mitkopiert.
E	Der Kopiervorgang erfolgt gleichzeitig auf dem Bildschirm (Echo).
F	Filtert das ASCII-Zeichen 0CH (Line Feed) aus der Datei.
Gn	Erlaubt das Kopieren von einer durch ‚n' angegebenen USER-Ebene (n=0 bis 15).
H	Prüft auf korrektes Intel-Hex-Format.
I	Ignoriert alle 00H bei einer Intel-Hex-Übertragung.
L	Umsetzung Groß- in Kleinbuchstaben.
N	Jede Zeile erhält eine zusätzliche Zeilennummer.
N2	Wie ‚N', nur mit 6-stelligen Zeilennummern.
O	Handelt es sich bei der zu übertragenden Datei um keine ASCII-Datei (z.B. Programmfile), ist als Parameter ‚O' anzugeben.
Pn	Seitenvorschub nach jeder ‚n'-ten Zeile (z.B. bei Druckerausgabe).
Qstring^Z	Kopiert die Datei, bis die Zeichenkette ‚string' gefunden wird.
R	Kopieren von Systemdateien (SYS).
Sstring^Z	Kopiert Datei von der gefundenen Zeichenkette ‚string' bis zum Dateiende.
Tn	Setzt TAB-Stop auf die ‚n'-te Spalte.
U	Umsetzung Klein- in Großbuchstaben.
V	Verify. Prüft die Kopie zusätzlich auf Richtigkeit (normalerweise nicht notwendig).
W	Erlaubt das Kopieren von R/O-Dateien.
Z	Setzt das Prüfbit auf 00H.

Fehlermeldungen:

INVALID PIP FORMAT: kommando
Unzulässiges PIP-Format. Auch wenn das Gleichheitszeichen (,=') fehlt.

NO FILE: filename.typ<[parameter]>
Quellfile wurde nicht gefunden.

Beispiele:

`A>PIP a:neudump.asm=a:dump.asm`

Kopiert das Programm ‚dump.asm' unter dem Namen ‚neudump.asm' auf die Diskette.

`A>PIP a:neudump.asm=a:dump.asm[L],a:dump.asm[N2]`

Verkettet zwei Dateien (hier beides Mal die gleiche Datei) und sichert sie unter dem Namen ‚neudump.asm' auf das Laufwerk ‚a:'. Bei der ersten Datei werden alle Zeichen in Kleinbuchstaben umgesetzt, die zweite erhält eine vorgestellte 6-stellige Zeilennummer. Probieren Sie das Beispiel einmal aus und TYPEn dann die Datei ‚neudump.asm'.

```
A>PIP crt:=a:dump.asm
A>PIP crt:=a:dump.asm[N]
A>PIP crt:=a:dump.asm[Qset up stack^Z]
```

Die Beipiele kopieren die Datei ‚dump.asm' direkt auf die Einheit ‚crt:' (cathode ray tube = Monitor), die im Normalfall dem Bildschirm zugeordnet ist (siehe STAT, 3.10). Hier können Sie sehr gut die Auswirkungen der PIP-Parameter erkennen. Das letzte Beispiel ‚kopiert' die Datei ‚dump.asm', bis die Zeichenfolge ‚set up stack' gefunden wurde. Probieren Sie ruhig andere Parameter aus.

`A>PIP lst:=a:dump.asm`

‚Kopiert' die Datei ‚dump.asm' auf die logische Listeinheit ‚lst:', die im Normalfall der Centronic-Schnittstelle zugeordnet ist (siehe STAT, 3.10).

`A>PIP lst:=a:dump.asm[ENLP40]`

Es können mehrere Parameter gleichzeitig gesetzt werden. In diesem Beispiel wird die Datei ‚dump.asm' auf dem Drucker (‚lst:') und auf dem Bildschirm (‚E') ausgegeben. Weiterhin wird jeder Zeile eine Nummer angestellt (‚N') und alle Großbuchstaben in Kleinbuchstaben umgewandelt (‚L'). Jeweils nach 40 Zeilen findet ein Seitenvorschub statt (‚P40').

3.8 Verify <Diskette A - Diskette A> (DISCCHK)

```
A>DISCCHK
```

Format: 1) DISCCHK

Das Dienstprogramm DISCCHK vergleicht zwei Disketteninhalte auf Übereinstimmung (Verify). Nach Einlegen der Quelldiskette werden je 8 Spuren in den Arbeitsspeicher des Rechners geladen und nach Diskettenwechsel mit der anderen Diskette verglichen. Dieser Vorgang wiederholt sich 5 mal. Trifft das Programm auf nicht übereinstimmende Sektoren (Fehler), wird eine Fehlermeldung mit Angaben über Spur und Sektor ausgegeben.

1. Vergleicht den Inhalt zweier Disketten auf Übereinstimmung.

Fehlermeldungen:

You must insert the destination disc into drive A
Zieldiskette fehlt im Laufwerk A

You must insert the source disc into drive A
Quelldiskette fehlt im Laufwerk A

* *Failed to verify the destination disc correctly: track/sector*
Fehler in Spur/Sektor aufgetreten

WARNING: Failed to compare discs correctly
WARNUNG: Fehler beim Vergleich aufgetreten

Beispiel:

```
A>DISCCHK

DISCCHK V2.0

Please insert source disc into drive A then press any
key:
Copy checking started
Read track (0-39)
Please insert destination disc into drive A then press
any key:
Checking track (0-39)
```

..... dieser Vorgang wiederholt sich 5 mal !! .....

```
Checking complete

Do you want to check another disc (Y/N)?N
A>
```

3.9 Verify <Diskette A - Diskette B> (CHKDISC)

A>CHKDISC

Format: 1) CHKDISC

Vergleicht den Inhalt der Quelldiskette (Laufwerk A) mit der Zieldiskette (Laufwerk B). Das Programm arbeitet identisch wie DISCCHK, allerdings wesentlich schneller und komfortabler.

1. Vergleicht die Diskette von Laufwerk A mit der Diskette in Laufwerk B. Dabei befindet sich die Quelldiskette in Laufwerk A.

Fehlermeldungen:
Siehe DISCCHK, Kap. 3.8.

Beispiel:

```
A>CHKDISC

CHKDISC V2.0

Please insert source disc into drive A and destination
disc into
drive B then press any key:
Copy checking started
Reading  track (0-39)
Checking track (0-39)
Copy checking complete
Do you want to check another disc (Y/N)?N
A>
```

3.10 Ausgabe von Diskетteninformationen (STAT)

A>STAT <laufwerk:><optionen>

Format:
1) STAT
2) STAT d:
3) STAT <d:>filename.typ
4) STAT <d:>filename.typ $atr
5) STAT DEV:
6) STAT VAL:
7) STAT log:=phy:
8) STAT USR:
9) STAT DSK:
10) STAT d:=R/O

STAT (Status, Statistik) ist ein ausführliches Disketten-Analyse-Programm. Es ist in mehrere Unterprogramme gegliedert und gibt detaillierte Informationen über die eingelegten Disketten und die angeschlossenen Laufwerke wieder. Weiterhin lassen sich die Zuordnungen zwischen logischen und physikalischen Einheiten ändern.

1. Liefert den freien Speicherplatz aller angeschlossenen Disketten, sowie eine Information, ob die Disketten schreibgeschützt sind.

```
A>STAT
A : R/W, Space 44k
B : R/O, Space 93k
```

2. Zeigt den freien Speicherplatz auf der Diskette im Laufwerk ‚d' an.

```
A>STAT a:
Bytes Remaining On a: 44k
```

3. Zeigt den vom File (es können beliebige Wildcards genutzt werden!) mit dem Namen ‚filename.typ' benötigten Speicherplatz sowie die Attribute an.

```
A>STAT CSAVE.COM
Recs   Bytes   EXT   Acc
  14     2k     1   R/W  A:CSAVE.COM
```

4. Setzt die Attribute der mit ‚filename.typ' bezeichneten Files (Wildcards zulässig !!). Nach dem ‚$'-Zeichen folgt eines der folgenden Attribute ‚atr'

   R/O = File schreibgeschützt (Read Only)
   R/W = Schreibschutz aufgehoben (Read/Write)
   SYS = File erscheint nicht mehr im Inhaltsverzeichnis
   DIR = File im Inhaltsverzeichnis führen

```
A>STAT a:*.bas $sys
EX1.BAS set to SYS
EX2.BAS set to SYS
A>DIR *.bas
A>
```

5. Zeigt die laufende Gerätezuordnung an (siehe nächstes Format).

```
A>STAT DEV:
CON: is CRT:
RDR: is TTY:
PUN: is TTY:
LST: is LPT:
A>
```

6. Zeigt die möglichen Zuordnungen von physikalischen zu logischen Einheiten an. CP/M unterscheidet 4 logische Einheiten

   CON: Konsole (Bildschirm)
   RDR: Lochstreifenleser
   PUN: Lochstreifenstanzer
   LST: Lister (Drucker)

   Diesen logischen Einheiten können diverse physikalische Einheiten zugeordnet werden. Besonders interessant sind zunächst die Einheiten

   CRT: Bildschirm und
   LPT: Centronics-Schnittstelle

   Wie aus dem Kommando ‚STAT DEV:' zu entnehmen war (siehe oben), ist die Konsole dem Bildschirm (CRT:) und der Listeinheit der Centronics-Schnittstelle (LPT:) zugeordnet. Diese Einheiten könnten aber auch anderen physikalischen Einheiten zugeordnet werden, wenn diese im Computer implementiert wären (z.B. Akustikkoppler). ‚STAT VAL:' zeigt die Zuordnungsmöglichkeiten an

```
A>STAT VAL:
Set Indicator : d:filename.typ $R/O $R/W $SYS $DIR
Disk Status   : DSK: d:DSK:
User Status   : USR:
Iobyte Assign :
CON: = TTY: CRT: BAT: UC1:
RDR: = TTY: PTP: UR1: UR2:
PUN: = TTY: PTR: UP1: UP2:
LST: = TTY: CRT: LPT: UL1
```

7. Ordnet einer logischen Einheit eine physikalische Einheit zu. Dabei sind nur die unter 6) beschriebenen Zuordnungen möglich. Das folgende Beispiel zeigt, wie die Druckerausgabe auf den Bildschirm umgeleitet werden kann.

```
A>STAT lst:=crt:
A>^P                    (Drucker aktivieren)
AA>>
```

8. Zeigt die aktuelle Benutzerebene und die auf der Diskette benutzten Ebenen an (siehe USER).

```
A>STAT USR:
Aktive User  : 0     (aktuelle USER-Ebene)
Aktive Files : 0     (Files nur in USER-Ebene vorhanden)
(Z.B.
Aktive Files : 0 1 2 Files in USER-Ebene 0,1,2 vorhanden)
```

9. Zeigt das Aufzeichnungsformat der aktuellen Diskette an.
10. Laufwerk "d:" wird schreibgeschützt.

Fehlermeldungen:

Invalid File Indicator
Unzulässiges Attribut

3.11 Automatische Kommandozeilenausführung (SUBMIT)

```
A>SUBMIT filename.SUB
```

Format: 1) SUBMIT filename<.SUB>
2) SUBMIT filename<.SUB> para 1 ,..para 9

Eine ASCII-Textdatei vom Typ ‚.SUB' enthält eine Liste von CP/M-Kommandos, die in der dort festgelegten Reihenfolge automatisch ausgeführt werden. Eine (ASCII-) SUB-Datei kann z.B. mit dem Editor ‚ED' oder einem beliebigen Textverarbeitungsprogramm erstellt werden. Jede Zeile darf allerdings nur ein Kommando enthalten! Eine SUB-Datei (hier mit drei Zeilen) könnte folgendermaßen aussehen

```
DIR *.BAS
ERA *.BAS
DIR *.BAS
```

1. Ruft die SUBMIT-Datei mit dem Namen ‚filename.SUB' auf und führt alle angegebenen Kommandos automatisch aus.

   Weiterhin erlaubt SUBMIT beim Aufruf einer Stapeldatei bis zu 9 Parameter zu übergeben. Parameter sind immer Filenamen oder Filetypen. Damit besteht die Möglichkeit, eine Stapeldatei universell zu halten. Nach den Stapeldateinamen folgen dann, jeweils durch ein Leerzeichen getrennt, die Parameter (bis zu 9). In der Stapeldatei werden die Parameter bestimmten Symbolen zugeordnet. So enthält „$1" den ersten übergebenen Parameter, „$2" den zweiten, bis „$9", der den 9. Parameter enthält.
2. Ruft die SUMBIT-Datei mit dem Namen ‚filename.SUB' mit bis zu 9 Parametern auf (siehe Beispiel 2).

Fehlermeldungen:

No ‚SUB' File Present
: File nicht vorhanden oder keine Datei vom Typ ‚.SUB'.

Error On Line xxx
: Fehlerhafte Kommandozeile in der SUB-Datei

Beispiel 1:

```
A>SUBMIT loesch.SUB          (siehe Beispiel oben)
                        ..... ab hier automatischer Ablauf ....
A>DIR *.BAS
A : EX1    BAS : EX2     BAS
A>ERA *.BAS
A>DIR *.BAS
NO FILE
A>
```

Beispiel 2:

Zum Verständnis soll eine Stapeldatei erstellt werden, die den Filetyp von 3 beliebigen BAS-Dateien in „TXT" ändert:

```
REN $1.TXT=$1.BAS
REN $2.TXT=$2.BAS
REN $3.TXT=$3.BAS
DIR *.TXT
```

Die Datei wird dann mit

```
SUBMIT TEST<.SUB> name1 name2 name3
```

aufgerufen. Sollen z.B. die Programme „TEXT.BAS", „LIST.BAS" und „EX1.BAS" umbenannt werden, so sieht der Aufruf folgendermaßen aus:

```
A<SUBMIT TEST text list ex1
```

3.12 Automatische Benutzereingabe (XSUB)

A>SUBMIT filename.SUB

Format: 1) SUBMIT filename.SUB

XSUB ist ein Teil des SUBMIT-Kommandos (siehe 3.11). Es ist als Kommando Bestandteil einer SUBMIT-Datei. XSUB ermöglicht es, innerhalb einer SUBMIT-Datei auch Informationen (Antworten) an das aufgerufene Programm zu übergeben. Nach dem XSUB-Kommando in einer SUBMIT-Datei folgen die gewünschten Kommandos mit den entsprechenden Informationen. Z.B. könnte eine SUBMIT-Datei folgendermaßen aussehen (siehe auch SUBMIT, 3.11)

```
XSUB
ERA *.*
N     (Antwort auf die Frage ALL (Y/N))
DIR *.BAS
```

1. Führt die in dem ASCII-File ‚filename.SUB' abgelegten Kommandos automatisch aus (siehe SUBMIT).

Fehlermeldungen:
Siehe SUBMIT, 3.11

Beispiel:

```
A>SUBMIT LOESCH.SUB        (siehe Beispiel oben)
                        ....ab hier automatischer Ablauf....
A>XSUB                     (XSUB wird aufgerufen)
A>ERA *.*
ALL (Y/N)?N                (‚N' wird übergeben)
A>DIR *.BAS
A : EX1      BAS : EX2       BAS
                        ....Ende des automatischen Ablaufs....
A>
```

3.13 Texteditor (ED)

```
A>ED filename.typ
```

Format: 1) ED filename.typ

Der Editor ED ist ein einfacher Texteditor, der zeilen- und zeichenorientiert arbeitet. Der Text befindet sich bei der Erstellung/Bearbeitung im Arbeitsspeicher des Rechners und wird nach Beendigung der Arbeit auf die Diskette gespeichert. Ein sogenannter Character Pointer (CP, Zeichenzeiger) zeigt immer auf das aktuell zu bearbeitende Zeichen. Alle Befehle des Editors bewirken eine Veränderung der CP-Position. Befindet sich beim Aufruf des Editors die gewünschte Textdatei ‚filename.typ' nicht auf der Diskette, legt er eine neue Textdatei an (NEW FILE).

ED-Befehle:

#A	Kopiert die gesamte Datei (Textfile, der beim Aufruf mit ‚filename.typ' angegeben ist) von der Diskette in den ED-Puffer.
nA	Kopiert ‚n' Zeilen der Textdatei in den ED-Puffer.
A	Kopiert eine Zeile in den ED-Puffer.
0A	Kopiert den Text, bis der ED-Puffer maximal zur Häfte gefüllt ist.
B, -B	Positioniert den CP auf den Anfang (B) bzw. auf das Ende (-B) des aktuellen Textes.
nC, -nC	Bewegt den CP um ‚n' Zeichen (vor/zurück).
nD, -nD	Löscht ‚n' Zeichen von der CP-Position aus (vor/zurück).
E	Beendet die Arbeit mit dem Editor. Sichert den Text auf die Diskette.
nFstring^Z	Sucht im Text (ab CP) die Zeichenfolge ‚string'. ‚n' gibt dabei an, wie oft der ‚string' gefunden werden muß, bis der Suchvorgang erfolgreich beendet wird. War der Vorgang erfolgreich, zeigt der CP auf das nächste Zeichen hinter dem Suchbegriff, andernfalls ist die CP-Position unverändert.
I	Eingabemodus. Ende mit ^Z.
Istring^Z	Füge ‚string' hinter CP ein.
Istring	Füge Zeile (‚string') hinter CP ein.

nJfinde^Zersetze^Zende^Z
Sucht den String ‚finde' und ersetzt ihn mit ‚ersetze'. Danach werden alle weiteren Zeichen gelöscht, bis der String ‚ende' gefunden ist. ‚n' gibt an, wie oft der Vorgang zu wiederholen ist.

nK, -nK Löscht ‚n' Zeilen von dem CP aus (vor/zurück).

nL, -nL Bewegt den CP um ‚n' Zeilen (vor/zurück).

nMbefehl^Z Der gültige ED-‚befehl' wird ‚n'-mal ausgeführt.

nNstring^Z Wie ‚F', durchsucht aber auch den Text auf der Diskette.

nP, -nP Zeigt den Text seitenweise an (1 Seite = 23 Zeilen). ‚n' bestimmt, wieviele Seiten gezeigt werden sollen.

Q Programmende, ohne den Text zu sichern !!

Rfilename Liest Datei ‚filename.LIB' von der Diskette und fügt sie nach dem CP ein.

nSsuch^Zersetze^Z
Sucht den String ‚such' und ersetzt ihn mit ‚ersetze'. Der Vorgang wird ‚n'-mal wiederholt.

nT, -nT Zeigt ‚n' Zeilen an (vor/zurück).

0T Zeigt aktuelle Zeile vom Anfang bis CP.

T Zeigt aktuelle Zeile vom CP bis Zeilenende.

0TT Zeigt die gesamte aktuelle Zeile, ohne den CP zu bewegen.

U, -U Umsetzung Klein- in Großbuchstaben (an/aus).

V, -V Zeilennumerierung (an/aus).

0V Gibt belegten und freien Textspeicherraum wieder.

nW Kopiert ‚n' Zeilen in die temporäre Datei und löscht die gesicherten Zeilen aus dem ED-Puffer.

nZ Verzögert die Ausführung um ‚n' Zeiteinheiten (SLEEP).

n: CP auf den Anfang der Zeile ‚n:' setzen.

:m Kommando bis zur Zeile ‚:m' fortsetzen.

n, -n Bewegt CP um ‚n' Zeichen (vor/zurück).

3.14 8080-Assembler (ASM)

```
A>ASM filename <.opt>
```

Format: 1) ASM filename <.opt>

Der 8080-Assembler erzeugt aus einem ASCII-Textfile vom Typ ‚.ASM' eine Hex-Datei (HEX) und eine listbare PRN-Datei. Die erzeugte HEX-Datei kann mit dem Programm LOAD (siehe 3.15) in ein unter CP/M lauffähiges Programm generiert werden. Der Quelltext muß dabei folgenden Aufbau besitzen

(Zeile) Label Mnemonics Operanden ;Kommentar

Zwischen jedem Feld muß mindestens ein Leerzeichen stehen. Eine Zeilennummer kann vorhanden sein. Sind im Text Zeilen vorhanden, die nicht assembliert werden können, so gibt der Assembler diese Zeilen mit einem vorgestellten Fehlercode aus (siehe unter Fehlermeldungen).

1. Assembliert den Quelltext mit dem Namen ‚filename.ASM' auf das durch ‚.opt' gekennzeichnete Laufwerk. Die Option besteht aus drei Buchstaben, wobei das

   1. Zeichen das Laufwerk mit dem Quelltext angibt (ASM),das
   2. Zeichen das Laufwerk, daß die HEX-Datei erhält und das
   3. Zeichen das Laufwerk für die Druckdatei (PRN) bestimmt.

   Ist keine Option angegeben, bezieht sich das Programm ausschließlich auf das Laufwerk A (entspricht ‚ASM filename.AAA').

Fehlermeldungen:

NO SOURCE FILE PRESENT
Quelltext nicht vorhanden oder kein ASM-File.

D (*Data error*) : Data-Anweisung nicht ausführbar
E (*Expression error*) : Ausdruck nicht auswertbar
L (*Label error*) : Label doppelt definiert
N (*Feature not implem.*) : Nicht implementierte Eigenschaft
O (*Overflow error*) : Ausdruck zu komplex
P (*Phase error*) : Wert einer Marke ist nicht definiert

R (*Register error*) : Unvollständige Angabe
S (*Syntax error*) : Falscher Befehl/Anweisung
U (*Undefined symbol*) : Nicht definiertes Label
V (*Value error*) : Fehler im Operanden

Beispiel:

```
A>ASM test.aaa

CP/M ASSEMBLER - VER 2.0
035E                (Endadresse !!)
00H USE FACTOR
END OF ASSEMBLY
A>
```

3.15 CP/M-Programm generieren (LOAD)

A>LOAD filename<.HEX>

Format: 1) LOAD filename.HEX

Das Dienstprogramm LOAD erzeugt aus einer HEX-Datei, die im Intel-Hex-Code abgelegt ist, ein unter CP/M laufähiges Programm vom Typ ,.COM'. Die COM-Datei wird auf dem aktuellen Laufwerk abgelegt (siehe auch 3.14). Fehlt hinter dem Dateinamen ‚filename' die Angabe ‚.HEX', nimmt LOAD automatisch die Datei ‚filename.HEX'. Die erzeugte Datei „filename.COM" ist unter CP/M lauffähig.

1. Erzeugt eine COM-Datei aus einer HEX-Datei auf dem aktuellen Laufwerk.

Fehlermeldungen:

ERROR: CANNOT OPEN SOURCE, LOAD ADDRESS 0100
Fehler: Finde Objektcode (HEX-Datei) nicht.

Beispiel:

```
A>LOAD test.hex

FIRST ADDRESS     0100      Startadresse
LAST  ADDRESS     035E      Endadresse
BYTES READ        025E      Filelänge
RECORDS WRITTEN     05      Recordanzahl (a 128 Bytes)

A>
```

3.16 Hexadezimale Ausgabe eines Files (DUMP)

```
A>DUMP filename.typ
```

Format: 1) DUMP filename.typ
2) DUMP *.typ
3) DUMP filename.*

Das gewünschte File wird in hexadezimaler (sedezimaler) Form aufgelistet. Die Auflistung kann mit ^S unterbrochen werden.

1. Auflistung des mit ‚filename.typ' bezeichneten Files.

2.+3. Sind Wildcards benutzt, wird nur der erste zutreffende File angezeigt.

Fehlermeldungen:

NO INPUT FILE PRESENT ON DISK
File nicht auf der Diskette oder Filename fehlt.

Beispiel:

```
A>DUMP amsdos.com

0000 21 10 01 CD D4 BC 0E 09 11 20 01 CD 05 00 C7 00
0010 42 41 53 49 C3 .............
```

3.17 CP/M-Monitor (DDT)

```
A>DDT <filename.typ>
```

Format: 1) DDT
2) DDT filename.typ

Der DDT (Dynamic Debugging Tool) ist ein umfangreiches Programm, um Maschinenprogramme (8080) zu testen, zu erstellen und zu verändern. Dafür stehen eine Reihe von Befehlen zur Verfügung. Nach erfolgreichem Laden erscheint das ‚-'-Zeichen als Prompt-Zeichen.

1. Lädt und startet das DDT-Programm
2. Lädt den DDT und zusätzlich das Maschinenprogramm ‚filename.typ' in den Arbeitsspeicher. Das zusätzlich geladene Programm liegt dann ab der Adresse 0100H und kann mit dem DDT bearbeitet werden.

DDT-Befehle:

D<<von>,<bis>>
Hexadezimale Ausgabe des gewünschten Speichers.
D : von aktueller Speicherstelle an
D1000 : von Speicherstelle 1000H an
D0100,0300 : von Speicherstelle 0100H bis 0300H

L<<von>,<bis>>
Disassemblierte Ausgabe des gewünschten Speichers.
L : von aktueller Speicherstelle an
L1000 : von Speicherstelle 1000H an
L0100,0300 : von Speicherstelle 0100H bis 0300H

Fvon,bis,wert
Füllt den Speicherraum ‚von'-‚bis' mit dem 8-Bit-Inhalt ‚wert'.

Mvon,bis,nach
Verschiebt den Speicherraum ‚von'-‚bis' zur Adresse ‚nach'.

Aadresse
Eingabe von 8080-Assembler-Befehlen, beginnend bei ‚adresse'. Die Eingabe wird mit der Eingabe eines Dezimalpunktes (‚.') beendet.

Sadresse
Arbeitsspeicher ab ‚adresse' verändern.

Es erscheint der alte Inhalt, der mit der ENTER-Taste beibehalten oder mit Eingabe eines neuen Wertes verändert werden kann. Eingabeende mit dem ‚#'- Zeichen.

Hwert1,wert2

Gibt die Summe und Differenz der beiden 16-Bit-Zahlen (hexadezimale Zahlen 00H bis FFH) ‚wert1' und ‚wert2' aus.

G<<start>,<break1>,break2>>

Startet ein sich im Arbeitsspeicher befindendes Programm. Werden keine Breakpoints gesetzt, so kehrt das Programm nicht mehr in die Kommandoebene von DDT zurück (ggf. „Absturz" oder Warmstart)!! Zum Testen sollte immer mindestens ein Breakpoint gesetzt werden, damit der DDT die Kontrolle wieder erhält. Die Breakpoint-Adresse wird nicht mit ausgeführt.

G	: ab der aktuellen Adresse des Program-Counter.
G0100	: ab Adresse 0100H.
G0100,0300	: ab Adresse 0100 und unterbricht (beendet) das Programm in der Adresse 0300.
G,0300	: ab der aktuellen Program-Counter und unterbricht (beendet) das Programm in der Adresse 0300.
G0100,0300,0500	: ab Adresse 0100 und endet mit Erreichen der Adresse 0300 oder 0500.

T<adresse>

Führt ein sich im Arbeitsspeicher befindendes Programm ab der „adresse" aus und gibt dabei alle Resiterinhalte, Zuständer der Flags und die disassemblierten Werte schrittweise aus. Wird nur ein „T" eingegeben, so wird nur eine Instruktion (ab dem Program-Counter) ausgeführt.

Ubreak

Zeigt nur einmal den Zustand der Register/Flags an und führt dann die nachfolgenden Instruktionen bis zu einem gesetzten Breakpoint aus.

X<register/flag>

Zeigt den aktuellen Zustand der Register und Flags an (X ohne Paramerte). Folgt nach dem „X" ein Parameter in Form eines Buchstabens, so kann der Inhalt eines Registers oder der Zustand eines Flags geändert werden:

Xc	: Carry-Flag	Xb	: BC-Register
Xz	: Zero-Flag	Xd	: DE-Register
Xm	: Minus-Flag	Xh	: HL-Register
Xe	: Parity-Flag (even)	Xs	: Stack-Pointer
Xi	: Interdigit-Flag	Xp	: Program-Counter
Xa	: Akkumulator		

Ifilename.typ

Bereitet das Nachladen des Programms ‚filename.typ' vor (FCB einrichten), das mit dem Befehle ‚R' dann in den Arbeitsspeicher geladen wird.

R<adresse>

Lädt das mit ‚I' vorbereitete Programm in den Arbeitsspeicher (0100H). Ist zusätzlich eine ‚adresse' angegeben, wird das Programm ab dieser Adresse geladen.

3.18 Konfigurations-Sektor ändern (SETUP)

```
A>SETUP
```

Format: 1) SETUP

Mit SETUP können charakteristische Eigenschaften des CPC-464/664 während des CP/M-Betriebs geändert werden. Nach der Bearbeitung des Konfigurations-Sektors wird dieser zurück auf die Diskette geschrieben (Spur 0/Sektor 1) und steht somit ständig zur Verfügung. Viele SETUP-Funktionen haben eine sehr spezielle Bedeutung (z.B. Modifizieren des Übertragungsverhaltens), die im Normalfall nicht geändert werden sollten. Einige Anpassungsfunktionen bieten eine gute Hilfe, um den CPC-464/664 an bestehende Programme anzupassen.

Initial command buffer:

Hier können CP/M-Kommandos eingefügt werden, welche automatisch nach dem Laden des CP/M-Betriebssystems ausgeführt werden.

```
LOGO.COM^M
```

Das Programm LOGO wird sofort nachgeladen (Autostart). Das Steuerzeichen ‚^M' (entspricht Carriage Return) entsteht durch die Eingabe des Potenzzeichens mit einem nachgestellten ‚M' (siehe 1.4).

Sign-on String:

Hier befindet sich die CP/M-Einschaltmeldung. Die ersten Steuerzeichen bestimmen die Farben des Bildschirms und des Borders (siehe spezielle Steuerzeichen, 1.5). Ändern Sie die Meldung Ihrer Sicherheitskopie z.B. in

```
^\@@@^\azz^]@@Sicherheitskopie CP/M 2.2 ^J^M
```

und als Einschaltmeldung erhalten Sie dann ‚Sicherheitskopie CP/M 2.2' in weißer Schrift auf schwarzem Hintergrund.

Printer power-up string:

Die hier eingegebene Zeichenfolge wird nach dem CP/M Start auf dem Drucker ausgegeben (Auch Steuerzeichen zur Normierung des Druckers sind möglich!!).

Keyboard translation

Tasten können einem anderen ASCII-Wert zugeordnet werden (KEY DEF).

Es stehen dazu 4 Funktionen zur Verfügung:

A(dd)	Taste neu belegen
D(elete)	Tastenbelegung löschen
C(lear)	Löschen aller Belegungen
F(inish)	Ende der Umbelegung

Mit der Add-Funktion werden Tasten neu belegt. Das Add-Kommando hat folgenden Aufbau:

A,tastennummer,normal,shift,control

Nach dem Buchstaben ‚A' folgt zunächst die Taste, deren Belegung geändert werden soll. Dazu wird die Tastennummer der gewünschten Taste angegeben. Entnehmen Sie die Tastennummer dem Handbuch (CPC-464 Anhang III/Seite 16, CPC-664 Kapitel 7/Seite 22). Dann folgen die neuen Zeichen in Form ihrer ASCII-Werte. Zunächst der Wert für das normale Zeichen, dann in Kombination mit der SHIFT-Taste und abschließend in Kombination mit der CONTROL-Taste. Entsprechend dieser Werte ist dann die Taste umbelegt worden. Das können Sie jetzt für alle Tasten machen (wenn Sie wollen). Als Beispiel soll die Belegung der Z- und Y-Taste getauscht werden:

A,71,121,89,25
A,43,122,90,26

Delete löscht eine bestimmte Tastenumbelegung:

D,tastennummer

Keyboard expansions string:
Belegt die Funktionsstasten mit einem String. Es stehen dazu wieder die von „keyboard translation" bekannten Funktionen zur Verfügung. Mit

A,nummer,funktionsstring

wird eine Funktionstaste belegt, Z.B.

A,0,DIR^M

legt das DIR-Kommando (plus Carriage Return) auf die Taste 0 des 10er Blocks.

IO Byte Set
Ändert die aktuelle Gerätezuordnung (siehe auch STAT, 3.10).

BIOS messages
Bestimmt, ob die BIOS-Fernmeldungen ausgegeben werden sollen. Bei ‚enabled' werden sie komplett auf dem Bildschirm ausgegeben, bei 'disabled' nur die Meldung ‚BDOS ERROR ON d:'.

Clear/Preserve initial command buffer:
Legt fest, ob der Inhalt des Kommandopuffers Vorrang vor der Eingabe über die Tastatur hat (‚preserve'=Puffer hat Vorrang, ‚clear'= Tastatur hat Vorrang).

Motor on delay
Legt die Zeit in 1/50 Sekunden fest, die zwischen dem Anlaufen des Laufwerks und dem Zugriff vergehen soll (sollte nicht geändert werden).

Motor of delay
Stellt die Nachlaufzeit der Laufwerke ein (1/50 Sekunden).

Stepping rate
Bestimmt die Zeit, die zwischen 2 Stepimpulsen der Laufwerke vergehen soll (normalerweise nicht verändern).

SIO Chanel A/B
Legt das Übertragungsverhalten einer seriellen Schnittstelle fest.

3.19 CP/M anpassen (MOVCPM)

```
A>MOVCPM <optionen>
```

Formate: 1) MOVCPM
2) MOVCPM x
3) MOVCPM x *

MOVCPM generiert eine dem tatsächlich vorhandenen Speicherraum angepaßte CP/M Version. Diese Anpassung ist nur dann notwendig, wenn

a) durch eine Speichererweiterung der Speicherraum vergrößert wurde oder

b) wenn ein bestimmter Speicherraum im Rechner nicht vom CP/M genutzt werden soll.

Die angepaßte CP/M-Version darf nicht kleiner als 16 kByte sein und nicht den tatsächlich vorhandenen Speicherraum überschreiten. ‚x' gibt die Anpassungsgröße in Einheiten je 1/4 kByte an.

1. Generiert eine 48 kByte CP/M-Version. Da im CPC-464/664 allerdings kein 48 kByte Speicherraum frei zur Verfügung steht, wird das vorhandene CP/M zerstört und der Rechner funktionsunfähig (Aus- und Einschalten).
2. Generiert eine ‚x' mal 1/4 kByte CP/M-Version. Dabei muß ‚x' größer gleich 64 und kleiner als 155 sein. Die neu generierte CP/M-Version wird automatisch an die gewünschte Stelle verschoben und von dort aus gestartet (das CP/M liegt direkt hinter dem generierten Arbeitsspeicher).
3. Generiert eine ‚x' mal 1/4 kByte CP/M-Version. Dabei muß ‚x' größer gleich 64 und kleiner als 179 sein. Der Stern (‚*') bewirkt, daß die neugenerierte CP/M-Version im Kernspeicher liegt und mit dem Befehl SYSGEN auf die Systemspuren geschrieben werden kann (die Kontrolle bleibt beim ‚alten' CP/M!). Auch kann mit dem SAVE-Befehl eine Datei mit dem verschobenen System erstellt werden.

Fehlermeldungen:

INVALID MEMORY SIZE

Unzulässige Speichergröße (kleiner als 16 kByte)

Beispiel:

```
A>MOVCPM 100 *

CONSTRUCTING 25k CP/M vers 2.2
READY FOR "SYSGEN" OR
"SAVE 34 CPM25.COM"
A>
```

3.20 Systemspuren kopieren (SYSGEN)

```
A>SYSGEN
```

Formate: 1) SYSGEN

SYSGEN kopiert die CP/M-Systemspuren (Spuren 0 und 1) auf eine andere Diskette. Mit SYSGEN können Sie z.B. eine gekaufte Diskette (z.B. die Programmdiskette zu dieser Buchreihe) mit dem CP/M-Betriebssystem versehen (es darf nicht mit verkauft werden).

1. Kopiert das CP/M-Betriebssystem von der Quelldiskette in den Arbeitsspeicher des Rechners. Nach dem Wechseln der Diskette (Zieldiskette) schreibt SYSGEN das gelesene Betriebssystem zurück auf die Spuren 0 und 1.

Fehlermeldungen:

SOURCE disc missing
: Quelldiskette nicht eingelegt

DESTINATION disc missing
: Zieldiskette nicht eingelegt

disc has unknown format
: Diskette hat ein unbekanntes Format

Beispiel:

```
A>SYSGEN

SYSGEN V2.0
Please insert SOURCE disc into drive A then press
any key:
Please insert DESTINATION disc into drive A then press
any key:
Do you wish to reconfigure another disc (Y/N)? : N
A>
```

3.21 Konfigurations-Sektor kopieren (BOOTGEN)

```
A>BOOTGEN
```

Formate: 1) BOOTGEN

Kopiert den Konfigurations-Sektor (Spur 0/Sektor 1) der CP/M-Systemspuren auf eine andere Diskette. Mit BOOTGEN kann z.B. ein mit SETUP (siehe 3.18) veränderter Konfigurations-Sektor auf beliebig viele Disketten kopiert werden.

1. Kopiert den Konfigurations-Sektor (siehe SETUP, 3.18) von der Quelldiskette auf eine Zieldiskette.

Fehlermeldungen:

Siehe SYSGEN, 3.20.

Beispiel:

```
A>BOOTGEN

BOOTGEN V2.0
Please insert SOURCE disc into drive A then press any
key:
Please insert DESTINATION disc into drive A then press
any key:
Do you wish to reconfigure another disc (Y/N)? : N
A>
```

3.22 Rückkehr ins BASIC (AMSDOS)

```
A>AMSDOS
```

Format: 1) AMSDOS

1. Nach dem Kommando wird die Systemkontrolle wieder an das CPC-Betriebssystem (AMSDOS) übergeben. Dabei wird ein Reset ausgeführt und in den 40-Zeichenmodus umgeschaltet.

Fehlermeldungen:

BASIC not found
Kein BASIC vorhanden (tritt beim CPC-464/664 nicht auf)

Bespiel

```
A>AMSDOS
                                    (Reset! 40-Zeichenmodus)
BASIC 1.0

READY
```

4.0 Das CP/M Betriebssystem

Dieser, der allgemeinen Handhabung und Anwendung der CP/M-Kommandos folgende Teil zum CP/M-Betriebssystem, beschäftigt sich mit der internen Organisation und Programmierung.

Das volle Verständnis der Programmierung setzt grundlegende Kenntnisse des 8080-Assembler-Codes voraus. Es wurde allerdings versucht, die Beispielprogramme so zu dokumentieren, daß auch ohne Kenntnisse des 8080-Assembler-Codes die allgemeine Funktion deutlich wird. Aus diesem Grunde sind die Beispiele sehr einfach und nur auf das Wesentliche beschränkt. Vielleicht ist dieser Teil auch eine Anregung für Sie, sich eingehend mit einer maschinenorientierten Programmiersprache (Assembler) zu beschäftigen.

Die Beispielprogramme können mit einem beliebigen Texteditor erstellt und anschließend mit dem 8080-Assembler ‚ASM' übersetzt werden. Der auf der Systemdiskette mitgelieferte Texteditor ‚ED' bietet diese Möglichkeit. Besitzen Sie ein Textverarbeitungsprogramm (z.B. WORDSTAR oder TASWORD), sollten Sie den Quell-Assembler-Text mit diesem Programm erstellen, da die Handhabung des Editors ‚ED' doch sehr umständlich ist. Der auf der Diskette abgelegte Quelltext muß als ‚ASM'-Dateityp gekennzeichnet sein (ggf. umbenennen). Der Quelltext muß weiterhin eine dem Assembler ‚ASM' verständliche Form aufweisen, damit er ihn einwandfrei übersetzen kann. Jede Zeile sollte mit einem Carriage Return abgeschlossen sein und folgendes Format aufweisen

(Zeilennr) Label Mnemonics Operand ;Kommentar

Jeder Teil der Zeile muß mit mindestens einem Leerzeichen voneinander getrennt sein. Besser ist es, die Zeile übersichlich zu formatieren. Z.B.

```
(1000)    LABEL    MVI    C,00H
(1010)             CALL   BDOS ;Warmstart
```

Das Benutzen einer Zeilennummer ist wahlfrei, da der Assembler ‚ASM' sie ignoriert. Alle nach einem Semikolon (‚;') folgenden Zeichen übersetzt der

Assembler nicht und führt sie in der listbaren ‚.PRN'-Datei als Kommentar.

Für alle diejenigen von Ihnen, die kein Textverarbeitungsprogramm besitzen und denen das Arbeiten mit dem Texteditor ‚ED' zu ungewohnt und umständlich ist, bietet das nachfolgende Programm ‚CPMEDIT' einen einfachen ‚Texteditor', mit dem Sie wie unter BASIC programmieren können. Bei diesem ‚Texteditor' kann die Assemblerzeile wie eine BASIC-Zeile eingegeben und geändert werden, allerdings muß nach der Zeilennummer das Remarkzeichen (Hochkomma) folgen. Ebenfalls müssen die Zeilennummern kleiner als 60000 sein, da sonst das Programm ‚CPMEDIT' überschrieben wird. Die Zeilen könnten z.B. folgendermaßen aussehen

```
1000 'LABEL     MVI   C,00H
1010 '          CALL   BDOS ;Warmstart
```

Nachdem der gesamte Quelltext eingegeben ist, wird das Programm ‚CPMEDIT' mit

```
RUN 60000
```

gestartet. Führen Sie dann mit ‚T' einen Testdurchlauf durch, wobei der Text so aufgelistet wird, wie er später gespeichert wird (Liststop mit der SPACE-Taste). Fehlt nach der Zeilennummer das Remarkzeichen, gibt das Programm eine Fehlermeldung aus. Korrigieren Sie diese Fehler entsprechend. Liegt der Quelltext in einer einwandfreien Form vor, können Sie den Text mit ‚S' auf der Diskette sichern. Der Filename ist ohne Typenbezeichnung anzugeben und darf nicht länger als 8 Zeichen sein. Das Programm erzeugt eine ASCII-Textdatei mit dem Namen ‚filename.ASM' und speichert gleichzeitig das gesamte Programm (CPMEDIT+Textzeilen) unter dem Namen ‚filename.BAS' ab. Damit haben Sie später die Möglichkeit, das Programm erneut zu laden und gegebenenfalls zu korrigieren.

```
60000 REM ************************************ M.W.Thoma **
60010 REM *                                               *
60020 REM *                 CP/M - Texteditor             *
60030 REM *                                               *
60040 REM *    Text in Remarkzeilen (') ablegen (<60000)  *
60050 REM *                Start mit RUN 60000            *
60060 REM *                                               *
60070 REM *************************************************
60080 MODE 2:INK 0,0:INK 1,26:BORDER 0
60090 PRINT:PRINT
60100 PRINT TAB(24);"***** CP/M - EDITOR V1.0 *****"
60110 PRINT:PRINT"   >  (T)est, (S)ichern oder (E)nde ?"
60120 in$=INKEY$:IF in$="" THEN 60120
60130 IF UPPER$(in$)="T" THEN aus=0:GOTO 60190
```

```
60140 IF UPPER$(in$)="E" THEN END
60150 IF UPPER$(in$)<>"S" THEN 60120
60160 INPUT "    >  Prog.Name :";na$
60170 IF LEN(na$)<1 OR LEN(na$)>8 THEN 60090
60180 OPENOUT na$+".asm":aus=9
60190 PRINT:PRINT:PRINT:start=368:feh=0:byte=0
60200 laenge=PEEK(start)+PEEK(start+1)*256
60210 zeinr=PEEK(start+2)+PEEK(start+3)*256
60220 IF zeinr>=60000 OR zeinr=0 THEN 60370
60230 x1=PEEK(start+4):x2=PEEK(start+5)
60240 IF x1=1 AND x2=192 THEN 60300
60250 PRINT USING"* #####";zeinr;
60260 PRINT" > ";CHR$(24);" Fehler !!!! ";CHR$(24)
60270 feh=feh+1
60280 start=start+laenge
60290 GOTO 60200
60300 ls$=INKEY$:IF (ls$=" " AND aus=0) THEN GOSUB 60450
60310 PRINT USING"* #####";zeinr;
60320 PRINT " > ";
60330 FOR i=start+6 TO start+laenge-1
60340 PRINT#aus, CHR$(PEEK(i));:byte=byte+1:NEXT
60350 PRINT#aus,CHR$(13);CHR$(10);
60360 GOTO 60280
60370 PRINT:PRINT
60380 PRINT"         > Ende des ";
60390 IF UPPER$(in$)="T" THEN PRINT"Test !":GOTO 60410
60400 PRINT"sichern !"
60410 PRINT"         > Fehler :";feh
60420 PRINT"         > Bytes :";byte
60430 IF UPPER$(in$)="S" THEN CLOSEOUT:SAVE na$+".bas"
60440 GOTO 60090
60450 ls$=INKEY$:IF ls$="" THEN 60450 ELSE RETURN
60460 REM ************************************************
```

4.1 CP/M Programm erstellen

Nachdem der Assembler-Quelltext mit einem Editor erstellt und als ‚.ASM'-Datei auf der Diskette abgelegt ist, kann er mit dem Assembler ‚ASM' weiter bearbeitet werden. Der Assembler erzeugt dabei zwei weitere Dateien (siehe 3.14)

1. Die ‚.HEX'-Datei ist eine Übersetzung des Assembler-Quelltextes in eine hexadezimale Form (auch Intel-Hex-Format genannt). Diese ‚.HEX'-Datei wird zur Erstellung eines ausführbaren Programms benötigt (siehe 3.15).
2. Die ‚.PRN'-Datei ist eine weitere Textdatei, die im Gegensatz zum Quelltext noch die hexadezimalen Codes der Befehle sowie die Fehlermeldungen enthält. Sie dient vor allem zur Fehlererkennung und Dokumentation. Die Datei ist mit ‚TYPE' listbar.

Hat der Assembler den Quelltext fehlerfrei übersetzt, kann mit Hilfe des LOAD-Programms (siehe 3.15) aus der ‚.HEX'-Datei ein unter CP/M lauffähiges Programm vom Typ ‚.COM' erstellt werden. Um das Programm zu starten, brauchen Sie dann nur noch

```
A>filename
```

eingeben. Der Filetyp ist nicht anzugeben, da unter CP/M nur Dateien vom Typ ‚.COM' ladbar sind. Nach diesen Vorgängen befinden sich dann die folgenden Files auf der Diskette

filename.**ASM** : Ursprungsdatei (Quelltext)
filename.**PRN** : Erweiterter Quelltext (vom ASM)
filename.**HEX** : Quelltext in hexadezimaler Form (vom ASM)
filename.**COM** : Ausführbares Programm (vom LOADer)

Ein Beispiel soll den gesamten Vorgang verdeutlich. Auf dem aktuellen Laufwerk befindet sich bereits ein Assembler-Quelltext mit dem Namen ‚TEST.ASM'

```
A>ASM TEST                    (nimmt automatisch die ‚.ASM'-Datei)

CP/M ASSEMBLER - VER 2.0
035E
000H USE FACTOR
END OF ASSEMBLY
```

```
A>LOAD TEST                (nimmt automatisch die ,.HEX'-Datei)

FIRST ADDRESS        0100
LAST  ADDRESS        035E
BYTES READ           025E
RECORDS WRITTEN        05

A>DIR TEST.*
A : TEST    ASM : TEST    COM : TEST    HEX : TEST    PRN
```

5.0 Interne Organisation

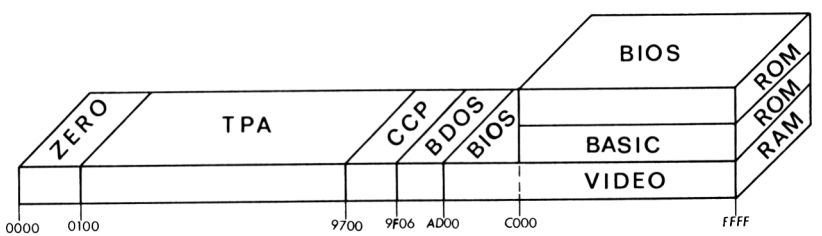
Das CP/M-Betriebssystem besteht aus drei weitgehend voneinander getrennten Teilen. Sie werden beim Laden direkt an das Ende des freien Arbeitsspeichers von 9700H bis BFFFH gelegt (Von C000H an liegt wie vom CPC gewohnt der Bildschirmspeicher). Weiterhin befindet sich parallel zum Bereich C000H bis FFFFH ein ROM (Disketten-ROM), das rechnerabhängige (hardwareabhängige) Funktionen für das CP/M regelt.

CCP Console Command Processor
Interpreter der Kommandozeile

BDOS Basic Disc Operating System
Regelt den Disketten- und Konsolenzugriff

BIOS Basic Input/Output System
Regelt hardwarespezifische Funktionen

Außerdem nutzt das CP/M die ersten 256 Bytes des Arbeitsspeichers (256 Bytes = 1 Page), worin sich wichtige Informationen für das CP/M (Sprungadressen, Kommando- und Datenspeicher etc.) befinden. Dieser Bereich nennt sich ‚Zero Page' oder ‚Base Page'. Der freie, für Programme nutzbare Speicherraum beginnt bei Adresse 0100H (256) und endet mit Adresse 96FFH (38655). Der unter CP/M nutzbare freie Arbeitsspeicher beträgt demnach rund 38 kByte und trägt den Namen ‚TPA' (Transient Program Area). Grafisch ist die Speicheraufteilung des CPC-464/664 unter CP/M folgendermaßen darstellbar (Aus zeichnungstechnischen Gründen ist das Kernal-ROM nicht berücksichtigt worden)



5.1 CCP - Console Command Processor

Der CCP belegt den Speicherraum 9700H bis 9F05H und ist das zentrale Bindeglied zwischen dem Benutzer und dem CP/M-Betriebssystem.

Die Eingabe und das Prüfen der Kommandozeile ist die erste Aufgabe, die der CCP zu leisten hat. Die nach dem Prompt-Zeichen (‚A>') eingegebenen und mit einem Carriage Return abgeschlossenen Zeichen werden zunächst in der Zero Page (Base Page) ab Adresse 0080H abgelegt. Insgesamt steht hier ein 128 Byte (0080H bis 00FFH) großer Speicherraum für die Kommandozeile zur Verfügung.

Daraufhin prüft der CCP, ob es sich bei dem ab Adresse 0080H liegenden Kommando um ein korrektes (eingebautes) Kommando handelt. Ist dies nicht der Fall, sucht der CCP auf dem aktuellen Laufwerk nach einer ‚.COM'-Datei, deren Name mit dem Kommando übereinstimmt. Findet der CCP auch auf dem Laufwerk kein entsprechendes Programm, erfolgt eine Fehlermeldung (‚kommando?').

Findet der CCP das Programm auf der Diskette, lädt er dieses ab der Adresse 0100H in die TPA ein und startet es automatisch. Dabei wird von dem geladenen Programm der Adress-Bereich 0080H bis 00FFH nicht überschrieben und kann aus diesem Grunde zur Übergabe von Informationen an das Programm genutzt werden. Sowie das geladene Programm gestartet ist, liegt die Kontrolle des Rechners beim geladenen Programm und nicht mehr beim CCP!

Programme enden meistens mit einem Warmstart, um die Kontrolle des Rechners wieder dem CCP (d.h. dem CP/M) zu übergeben. Ein Warmstart hat zur Folge, daß das CP/M (CCP und BDOS) erneut von der (System-) Diskette in den Rechner geladen wird, ohne den Arbeitsspeicher zu zerstören. Dieser Warmstart stellt sicher, daß die Kontrolle des Rechners einem vollständigen und funktionsfähigen CP/M übergeben wird. Da der CCP bei der Programmausführung nicht mehr benötigt und durch einen Warmstart nachgeladen wird, ist es durchaus möglich, auch den CCP-Bereich als Arbeitsspeicher zu benutzen. Damit vergrößert sich der nutzbare Arbeitsspeicher um 0806H Bytes (Beginn BDOS) und beträgt somit 40453 Bytes (knapp 40 kBytes).

5.2 BDOS - Basic Disc Operating System

Das BDOS regelt alle Zugriffe auf die Laufwerke (Diskette/Festplatte) und steuert den Zugriff auf die Konsole. Der Ausdruck ‚Konsole' steht dabei für alle mit dem Rechner verbundenen Einheiten wie Tastatur, Bildschirm, Drucker, Lochstreifen, etc..

Das BDOS liegt nach dem Laden in den Adressen 9F06H bis ACFFH und stellt mehrere grundsätzliche Ein- und Ausgaberoutinen zur Verfügung, die mit einem zentralen Einsprung über die Zero Page (Base Page) Adresse 0005H eingeleitet werden können. Die Auswahl einer BDOS-Routine erfolgt durch eine (Kenn-) Wertzuweisung in das C-Register des 8080/Z80-Prozessors. Nach dem Einsprung ins BDOS erkennt dieser anhand des C-Registers, welche Funktion gewünscht ist. Diese Routinen sind sehr vielseitig und ermöglichen es u.a., Zeichen auf dem Bildschirm auszugeben, Tastaturabfrage oder einen Block von der Diskette zu lesen (Einige BDOS-Routinen benötigen weitere Informationen, um Funktionen korrekt einzuleiten bzw. auszuführen).

Eine ausführliche Beschreibung aller BDOS-Funktionen finden Sie in den Kapiteln 6.0 bis 6.38.

5.3 BIOS - Basic Input Output System

Das BIOS ist bei weitem der größte Teil des CP/M-Betriebssystems. Es beginnt mit Adresse AD00H und endet zunächst mit Beginn des Bildschirmspeichers in Adresse BFFFH, setzt sich allerdings in einem Extra-ROM fort (Disketten-ROM).

Das BIOS ist die Einheit, die die eigentliche Arbeit des CP/M-Systems durchführen muß. Alle durch das BDOS vereinheitlichten Funktionen müssen dem jeweiligen Computer angepaßt werden. Schon die Ausgabe von Zeichen auf dem Bildschirm ist durch die technische Gegebenheit eines Computers grundsätzlich verschieden. Das BIOS enthält alle rechnerspezifischen Routinen, damit eine vom BDOS gewünschte Funktion auf dem Rechner das entsprechende Ergebnis liefert. D.h. das BIOS ist bei jedem CP/M-fähigen Computertyp verschieden!!! (Das BIOS gehört nicht zum Lieferumfang des CP/M-Vertreibers ‚Digital Research', sondern der Computerhersteller paßt das BIOS selbst an den Computer an)

Das BIOS beginnt mit einer umfangreichen Sprungtabelle (AD00H bis AD32H), über die das BDOS ins BIOS springt. Diese Sprungtabelle besitzt eine einheitliche Reihenfolge von 17 Sprungbefehlen (CP/M 2.2), die den Beginn der entsprechenden BIOS-Routinen zeigen (Computerspezifisch, hier Sprung ins BIOS-ROM). Mit Hilfe des ‚DDT' (siehe 3.17) können Sie sich diese Tabelle ansehen

```
AD00  JMP  C1B2    ;Kaltstart
AD03  JMP  C2BE    ;Warmstart
AD06  JMP  C2E1    ;Konsolenstatus holen
AD09  JMP  C2C6    ;Konsoleneingabe
AD0C  JMP  C2C8    ;Konsolenausgabe
AD0F  JMP  C2D2    ;Listausgabe (Drucker)
AD12  JMP  C2D7    ;Punchausgabe
AD15  JMP  C2DC    ;Readereingabe
AD18  JMP  C2E9    ;Schreib/Lesekopf auf Spur 0/Sektor 0
AD1B  JMP  C2F2    ;Diskettenauswahl
AD1E  JMP  C524    ;Spur auswählen
AD21  JMP  C529    ;Sektor auswählen
AD24  JMP  C51A    ;DMA-Adresse auswählen
AD27  JMP  C2F7    ;Sektor lesen
AD2A  JMP  C2FC    ;Sektor beschreiben
AD2D  JMP  C2CD    ;Liststatus
AD30  JMP  C55A    ;Sektor anpassen
```

Alle Routinen können auch mit einem Sprung über die angegebenen Adressen eingeleitet werden (z.B. CALL AD00). Die eigentliche Routine befindet sich im (Disketten-) ROM und endet mit einem Return-Befehl (RET).

Besonders interessant sind die Routinen zum Positionieren des Schreib-Lese-Kopfes und zum Einlesen eines bestimmten Records, da diese Funktionen nicht direkt über das BDOS aufrufbar sind. Im Kapitel 7.4 finden Sie ein ausführliches Anwendungsbeispiel.

5.4 Zero Page (Base Page)

Die ersten 256 Bytes des Arbeitsspeichers (0000H bis 00FFH) werden vom CP/M-System sehr vielseitig genutzt. Neben dem Einsprung ins BDOS (0005H) und zum BIOS-Warmstart (0000H) befinden sich hier vor allem Zwischenspeicher (Puffer) für verschiedene Aufgaben.

Ein aus dem Abschnitt zum CCP schon bekannter Puffer ist der Bereich von 0080H bis 00FFH (128 Bytes). Hier ist bei einer Kommandoeingabe z.B. die gesamte Kommandozeile zur Prüfung zwischengespeichert. Weiterhin wird dieser Puffer zur Speicherung von Daten, die zwischen dem Rechner und den angeschlossenen Laufwerken ausgetauscht werden, benutzt. Beim Lesen oder Schreiben eines bestimmten Sektors (128 Bytes) von der Diskette dient standardmäßig dieser Speicherbereich als Zwischenspeicher und trägt dann den Namen DMA (Direct Memory Access). Die DMA muß nicht an diesem Platz liegen, sondern kann frei innerhalb der gesamten TPA liegen (siehe 6.27).

Ein zweiter sehr wichtiger Speicherbereich innerhalb der Zero Page ist der sogenannte FCB (File Control Block). Dieser 36 Bytes große Speicher beginnt normalerweise bei 005CH. Für jede Datei, auf die das CP/M zugreifen soll, muß ein FCB vorhanden sein. Auch der FCB muß nicht in diesem Bereich liegen, sondern kann sich wie die DMA an einem beliebigen Platz innerhalb der TPA befinden. Werden z.B. gleichzeitig zwei Dateien bearbeitet, so muß zumindest ein FCB an einem anderen Platz liegen.

In dem FCB befinden sich die Informationen, um welchen File es sich handelt, wo er sich befindet, sowie auf welchen Teil (Record) des Files zuzugreifen ist (bei Direktzugriff). Einige Bereiche des FCB müssen vom Benutzer mit eindeutigen Informationen gefüllt werden, andere sind für das CP/M-System reserviert oder werden von diesem versorgt.

Aufbau des File-Control-Blocks

Bytes	Funktion
00	Laufwerkbestimmung (0 bis 16) 0 = aktuelles Laufwerk (!!) 1 = Laufwerk A 2 = Laufwerk B (Standard CP/M bis 16 = Laufwerk P)
01-08	Filename in Großbuchstaben
09-11	Filetyp in Großbuchstaben
12	Extensionsnummer
13-31	Reserviert für CP/M-System
32	Aktuelle Blocknummer
33-35	Recordnummer für Direktzugriff Die Bytes 33 und 34 bilden eine 16-Bit-Zahl (Byte 33 = Lowbyte), die eine Recordnummer zwischen 0 und 65535 kennzeichnen. Byte 35 zeigt einen Überlauf an.

Die Bytes 1 bis 11 sind vom Benutzer mit ASCII-Werten zu füllen, die den Filenamen und Filetyp bestimmen. Alle anderen Bytes sind normalerweise auf Null (00H) zu setzen.

Die Bytes 33 bis 35 sind nur bei dem Direktzugriff (Direct Access) auf bestimmte Records notwendig (siehe 6.34).

Nachdem der FCB vorbereitet ist, kann mit Hilfe einer BDOS-Routine die bezeichnete Datei auf der Diskette gesucht werden. Ist sie gefunden, lagert das CP/M konkrete Informationen über den Ort der Datei auf der Diskette in den FCB ein (Bytes 12 bis 33). Gleichzeitig bereitet das CP/M den Zugriff auf den ersten Record der Datei vor. Ein Zugriff (z.B. Lesen) liest dann diesen Record in die DMA ein.

6.0 Die BDOS-Routinen

Die nächsten Seiten gehen ausführlich auf die Funktion und Anwendung (Programmierung) aller BDOS-Routinen ein. Fast alle BDOS-Funktionen sind an Hand eines kleinen Beispielprogramms erläutert. Es ist immer der Ausdruck, der durch die Assemblierung entstandenen ,.PRN'-Datei ist (siehe 4.1), aufgelistet. Obwohl alle Beispiele ausschließlich im 8080-Assembler-Code geschrieben sind, sollten sie durch die Dokumentation auch für Maschinensprachelaien verständlich sein.

Die BDOS-Funktionen zeigen ganz deutlich den elementaren Vorteil des CP/M-Betriebssystems gegenüber anderen, nicht vereinheitlichten Betriebssystemen. An einem Beispiel möchte ich dieses noch einmal verdeutlichen.

Um ein Zeichen auf dem Bildschirm eines beliebigen Computers auszugeben, sind umfangreiche Kenntnisse über den Aufbau und Funktion des jeweiligen Computers notwendig. Beim COMMODORE 64 z.B. ist der Aufbau anders als beim APPLE oder beim CPC-464/664. D.h. ein (Maschinen-) Programm ist immer auf den jeweiligen Computer zugeschnitten und läuft ausschließlich auf diesem. Auf einem anderen Computer ist dieses Programm vollkommen unbrauchbar. Sind allerdings alle wesentlichen Ein- und Ausgaberoutinen wie unter CP/M einheitlich erreichbar, braucht der Programmierer keine Kenntnisse über den Aufbau des Computers (Betriebssystem), sondern muß nur wissen, wie er eine bestimmte Routine aufrufen kann. Dabei kann es dem Benutzter egal sein, wie der Computer (BIOS) dieses Problem löst!

Das CP/M (BDOS) verfügt über eine Reihe von solchen vereinheitlichten Routinenaufrufen, so z.B. auch für die Aufgabe, ein Zeichen auf dem Bildschirm auszugeben. Jede vereinheitlichte Ein-/Ausgaberoutine besitzt eine ‚Kennzahl' die dem BDOS mitgeteilt werden muß. Anhand dieser ‚Kennzahl' erkennt das BDOS, welche Routine gewünscht ist und führt alle weiteren Aufgaben bis zur Ausgabe eines Zeichens eigenständig durch. Dabei ist es egal, ob Sie unter CP/M auf einem COMMODORE, IBM, APPLE oder

CPC arbeiten, die ‚Kennzahlen' und der Aufruf sind identisch. Analog dazu ist ein auf dem IBM erstelltes CP/M-Programm auch auf dem CPC-464/664 lauffähig.

Insgesamt stehen 39 verschiedene Routinen zur Verfügung (CP/M 2.2) und diese tragen die ‚Kennzahlen' 0 bis 38. Die ‚Kennzahl' ist dem BDOS im C-Register des 8080/Z80-Prozessors zu übermitteln.

```
MVI  C,kennzahl          ;Lade C-Register mit Kennzahl
```

Dann folgt der Einsprung ins BDOS. Hier wird der Wert des C-Registers geprüft und zur entsprechenden Routine verzweigt. Der Einsprung ins BDOS ist einheitlich über die Adresse 0005H möglich

```
CALL 0005H               ;Einsprung ins BDOS
```

Viele BDOS-Funktioen benötigen weitere Parameter, um die Funktion korrekt auszuführen. Um ein Zeichen auf dem Bildschirm auszugeben, muß dem BDOS natürlich noch mitgeteilt werden, welches Zeichen auszugeben ist. Diese Informationen müssen dann in bestimmte Register des Prozessors gelegt werden. Andere BDOS-Funktionen liefern ein Ergebnis (z.B. Zeichen von der Tastatur holen), welches sich nach der Rückkehr aus dem BDOS in bestimmten Registern befindet (die genauen Übergaberegister entnehmen Sie bitte den Beschreibungen der jeweiligen BDOS-Funktionen). Das Programm zur Ausgabe eines Zeichens auf dem Bildschirm sieht z.B. folgendermaßen aus

```
MVI  C,02H               ;Kennzahl 2 in das C-Register laden
                         ;Funktion: Zeichen ausgeben
MVI  E,41H               ;Übergaberegister E mit ASCII-Wert
                          des
                         ;Zeichens laden (41H = 65 = ‚A')
CALL 0005H               ;Sprung ins BDOS. Ausführen der
                          Funktion
RET                      ;Programmende
```

Der Aufruf aller anderen Funktionen ist identisch (gegebenenfalls mehr oder andere Übergaberegister).

6.1 BDOS-Funktion 0: System Reset (BOOT)

Die erste BDOS-Funktion bewirkt einen Warmstart des CP/M-Systems (siehe auch ^C). Dabei wird das CP/M neu geladen und gestartet. Der (normale) Arbeitsspeicher bleibt dabei unverändert. Diese Funktion stellt beim Programmende sicher, daß die Kontrolle einem einwandfreien und vollständigen CP/M übergeben wird (Der CCP kann als TPA genutzt werden !!). Vor Aufruf der Funktion muß sichergestellt sein, daß die aktuelle Diskette die CP/M-Systemspuren enthält. Diese BDOS-Funktion kann auch mit dem Sprungbefehl

JMP 0000H

eingeleitet werden.

Übergabe-Parameter:

Eingang: C-Register = 00H

Ausgang: Keine

Beispiel:

Das Beispielprogramm führt einen Warmstart aus. Dabei ist deutlich zu bemerken, daß der CCP und das BDOS erneut geladen werden. Diese Funktion bietet sich als Programmende an.

```
                ;*********************************************************
                ;*                                                       *
                ;*              BDOS-Funktion No. 00                     *
                ;*      Systemneustart (system reset , boot)             *
                ;*                                                       *
                ;*    <C>=00H (in)                                       *
                ;*                                                       *
                ;*********************************************************
                ;
0005 =          BDOS    EQU     0005H
0000 =          BOOT    EQU       00H
                ;
0100                    ORG     0100H
0100 0E00               MVI    C,BOOT ;<C>=00H (warm start)
0102 CD0500             CALL     BDOS ;ausfuehren
                                      ;Prog. kommt nicht hierher, da
                                      ;Funktion an CCP uebergeben ist
0105 C9                 RET
0106            END
```

6.2 BDOS-Funktion 1: Console Input (CONIN)

Übergabe eines Zeichens von der Tastatur an das laufende Programm. Nach Aufruf der Funktion wartet die BDOS-Funktion auf ein Zeichen und kehrt erst nach Übergabe eines Zeichens zurück. Das Zeichen befindet sich im Akkumulator und wird gleichzeitig auf dem Bildschirm ausgegeben (Echo).

Übergabe-Parameter:

Eingang: C-Register = 01H

Ausgang: Akkumulator = ASCII-Zeichenwert

Beispiel:

Wartet auf die Eingabe eines Zeichens von der Tastatur. Das Zeichen wird zusätzlich auf dem Bildschirm ausgegeben. Das Programm endet mit der Betätigung der ENTER-Taste (wartet auf Carriage Return).

```
                ;*****************************************************
                ;*                                                   *
                ;*              BDOS-Funktion No. 01                 *
                ;*      Zeichen von Tastatur holen (Console Input)   *
                ;*                                                   *
                ;*    <C>=01H (in)       <A>=ASCII-Zeichen (out)     *
                ;*                                                   *
                ;*****************************************************
                ;
0005 =          BDOS    EQU     0005H
0001 =          CONIN   EQU       01H
                ;
0100                    ORG     0100H
0100 0E01       LABEL   MVI   C,CONIN ;Lade <C> mit 01H um Zeichen
0102 CD0500             CALL     BDOS ;in den <A> zu holen (wartet)
0105 FE0D               CPI       0DH ;Vergleiche <A> mit CR
0107 C20001             JNZ     LABEL ;Ungleich ? Ja, dann springe
010A C9                 RET
010B            END
```

6.3 BDOS-Funktion 2: Console Output (CONOUT)

Ein ASCII-Zeichen, dessen Wert sich im E-Register befindet, wird an der aktuellen Cursorposition des Bildschirms ausgegeben. Anzumerken ist, daß sich nach Aufruf dieser Funktion alle Register des Prozessors in einem undefinierten Zustand befinden. Es ist daher ratsam, alle wichtigen (vom Programm benutzten) Register vorher zu ‚retten' (PUSH) und nach dieser Funktion wiederzuholen (POP).

Übergabe-Parameter:

Eingang: C-Register = 02H
E-Register = ASCII-Zeichenwert

Ausgang: Keine

Beispiel:

Das E-Register wird mit dem ASCII-Wert für das Zeichen ‚*' geladen, welches nach dem Aufruf der BDOS-Funktion 2 auf dem Bildschirm ausgegeben wird.

```
                ;*******************************************************
                ;*                                                     *
                ;*              BDOS-Funktion No. 02                   *
                ;*Zeichen auf dem Bildschirm ausgeben (Console output)*
                ;*                                                     *
                ;*    <C>=01H (in)        <E>=ASCII-Zeichen (in)       *
                ;*                                                     *
                ;*******************************************************
                ;
0005 =          BDOS    EQU     0005H
0002 =          CONOUT  EQU       02H
                ;
0100                    ORG     0100H
0100 1E2A               MVI     E,'*' ;<E> mit ASCII-Wert laden
0102 0E02               MVI   C,CONOUT ;<C>=02H
0104 CD0500             CALL     BDOS ;und ausgeben
0107 C9                 RET
0108            END
```

6.4 BDOS-Funktion 3: Reader Input (READER)

Holt ein Zeichen von dem Lochstreifenleser (Reader) in den Akkumulator und kehrt erst dann zurück, wenn ein Zeichen gelesen wurde. Diese Funktion arbeitet identisch wie die BDOS-Funktion 1 (CONIN).

Übergabe-Parameter:

Eingang: C-Register = 03H

Ausgang: Akkumulator = ASCII-Zeichenwert

6.5 BDOS-Funktion 4: Punch Output (PUNCH)

Sendet einen ASCII-Zeichenwert im E-Register zum Lochstreifenleser (Punch). Disese Funktion arbeitet identisch wie die BDOS-Funktion 2 (CONOUT).

Übergabe-Parameter:

Eingang: C-Register = 04H
E-Register = ASCII-Zeichenwert

Ausgang: Keine

6.6 BDOS-Funktion 5: List Output (LIST)

Sendet einen ASCII-Zeichenwert im E-Register zum Drucker. Dabei muß sichergestellt sein, daß der Drucker angeschlossen und druckbereit ist.

Übergabe-Parameter:

Eingang: C-Register = 05H
E-Register = ASCII-Zeichenwert

Ausgang: Keine

Beispiel:

Das E-Register wird mit dem ASCII-Wert für das Zeichen ‚*' geladen und anschließend der BDOS-Funktion übergeben. Ist der Drucker nicht aktiv, wartet die Funktion solange, bis der Drucker aktiv ist!

```
                ;*************************************************
                ;*                                               *
                ;*              BDOS-Funktion No. 05             *
                ;*      Zeichen zum Drucker senden (List output) *
                ;*                                               *
                ;*    <C>=05H (in)       <E>=ASCII-Zeichen (in)  *
                ;*                                               *
                ;*************************************************
                ;
0005 =          BDOS    EQU     0005H
0005 =          LIST    EQU       05H
                ;
0100                    ORG     0100H
0100 1E2A               MVI      E,'*' ;<E> mit ASCII-Wert laden
0102 0E05               MVI    C,LIST ;<C>=05H
0104 CD0500             CALL     BDOS ;und ausgeben
0107 C9                 RET
0108            END
```

6.7 BDOS-Funktion 6: Direct Console In/Out (CONDIR)

Holt oder übergibt einen ASCII-Zeichenwert von/an die Konsole. Es werden allerdings keine CP/M-Steuerzeichen (^C, ^S, etc.) berücksichtigt. Ob ein ASCII-Zeichen geholt oder gesendet wird, ist abhängig von dem Zustand des E-Registers

E-Register = 0FFH : Hole Zeichen
E-Register < 0FFH : Sende ASCII-Zeichenwert (E-Register)

Ist das E-Register kleiner als 0FFH, so wird der Wert dieses Registers als der gewünschte Zeichenwert interpretiert.

Nach dem ‚Holen' eines ASCII-Zeichenwertes (E-Register = 0FFH) befindet sich das Zeichen im Akkumulator.

Diese BDOS-Funktion arbeitet prinzipiell ähnlich wie die BDOS-Funktionen 1 und 2 (CONIN und CONOUT). Da keine CP/M-Steuerzeichen berücksichtigt werden können, sollte diese Funktion vermieden werden.

Übergabe-Parameter:

Eingang: C-Register = 06H
E-Register = 0FFH (Hole Zeichen)
oder E-Register < 0FFH (zu sendendes Zeichen)

Ausgang: Akkumulator = übergebenes Zeichen (wenn E-Register =0FFH)

6.8 BDOS-Funktion 7: Get IO-Byte (GETIO)

Kopiert das IO-Byte (Speicherstelle 0003H der Zero Page) in den Akkumulator. Das IO-Byte ist wesentlich verantwortlich für die Zuordnung der physikalischen zu den logischen Einheiten. Insgesamt unterstützt CP/M die 4 logischen Einheiten, die jeweils durch 2 Bits des IO-Bytes dargestellt werden

CON:	Konsole	Bit 0 und 1
RDR:	Lochstreifenleser	Bit 2 und 3
PUN:	Lochstreifenstanzer	Bit 4 und 5
LST:	Listeinheiten	Bit 6 und 7

Je nach Kombination der Bitpaare findet eine physikalische Zuordnung statt (siehe auch 3.10, STAT DEV: und STAT VAL:)

Übergabe-Parameter:

Eingang: C-Register = 07H

Ausgang: Akkumulator = IO-Byte

Beispiel:

Das IO-Byte wird in den Akkumulator geladen und anschließend als duale Zahl ausgegeben.

```
                ;*********************************************************
                ;*                                                       *
                ;*              BDOS-Funktion No. 07                     *
                ;*         IO-Byte auslesen (Get IO-Byte)                *
                ;*                                                       *
                ;*    <C>=07H (in)           <A> IO-Byte (out)           *
                ;*                                                       *
                ;*********************************************************
                ;
0005 =          BDOS    EQU     0005H
0002 =          CONOUT  EQU       02H
0009 =          STROUT  EQU       09H
0007 =          GETIO   EQU       07H
                ;
0100                    ORG     0100H
0100 112D01             LXI   D,STRING ;<DE> mit Stringadresse laden
0103 0E09               MVI   C,STROUT ;<C>=09H
0105 CD0500             CALL      BDOS ;und ausgeben
                ;
0108 0E07               MVI   C,GETIO ; IO-Byte in den <A> holen
010A CD0500             CALL      BDOS
                ;
                ; Umrechnung in eine Dual-Zahl
```

```
010D 0608              MVI     B,08H ; Zaehler auf 8 setzen
010F 17        BINAER  RAL           ; Schiebe <A> nach links
0110 C5                PUSH      B ; Register retten
0111 F5                PUSH      A
0112 DA1F01            JC      LABEL ; Bit gesetzt ? Ja, springe
0115 0E02              MVI   C,CONOUT ; Ausgabe einer 0
0117 1E30              MVI     E,30H
0119 CD0500            CALL     BDOS
011C C32601            JMP    LABEL1
011F 0E02      LABEL   MVI   C,CONOUT ; Ausgabe einer 1
0121 1E31              MVI     E,31H
0123 CD0500            CALL     BDOS
0126 F1        LABEL1  POP       A ; Register wieder holen
0127 C1                POP       B
0128 05                DCR       B
0129 C20F01            JNZ    BINAER ; <B>=0 ? Nein, weiter
012C C9                RET
               ;
012D 4461732049 STRING DB 'Das IO-Byte ist : $'
0140           END
```

6.9 BDOS-Funktion 8: Set IO-Byte (SETIO)

Kopiert den Inhalt des E-Registers in die Speicherstelle 0003H der Zero Page und setzt damit das IO-Byte (siehe auch BDOS-Funktion 7, sowie Kapitel 3.10, STAT VAL:, STAT DEV:).

Übergabe-Parameter:

Eingang:	C-Register = 07H
	E-Register = IO-Byte
Ausgang:	Keine

6.10 BDOS-Funktion 9: Print String (STROUT)

Eine sehr komfortable Routine bietet die BDOS-Funktion 9. Eine Folge von ASCII-Zeichen, die durch das DE-Register (16-Bit-Register) adressiert ist (Stringbeginn), wird auf dem Bildschirm ausgegeben. Das Ende der Zeichenfolge muß mit dem Dollarzeichen ‚$' gekennzeichnet sein. Alle Steuerzeichen werden berücksichtigt und entsprechend ausgeführt.

Übergabe-Register:

Eingang: C-Register = 09H
DE-Register = Stringadresse (Stringbeginn)

Ausgang: Keine

Beispiel:

Eine Zeichenkette (hier ‚Teststring' plus Line Feed und Carriage Return), die ab der Adresse ‚STRING' liegt, wird auf dem Bildschirm ausgegeben. Das ‚$' in Adresse 0115H kennzeichnet das Stringende.

```
                ;*****************************************************
                ;*                                                   *
                ;*              BDOS-Funktion No. 09                 *
                ;*        Zeichenkette ausgeben (print string)       *
                ;*                                                   *
                ;*    <C>=09H (in)       <DE>=Stringadresse (in)     *
                ;*                                                   *
                ;*****************************************************
                ;
0005 =          BDOS      EQU     0005H
0009 =          STROUT    EQU       09H
                ;
0100                      ORG     0100H
0100 110901               LXI   D,STRING ;<DE> mit Stringadresse laden
0103 0E09                 MVI   C,STROUT ;<C>=09H
0105 CD0500               CALL      BDOS ;und ausgeben
0108 C9                   RET
                ;
0109 5465737473 STRING    DB 'Teststring';Text
0113 0A0D                 DB  0AH,0DH    ;LF und CR
0115 24                   DB '$'         ;markiert Stringende
0116            END
```

6.11 BDOS-Funktion 10: Read Console Buffer (BUFIN)

Mit Hilfe dieser Funktion ist es möglich, eine Folge von ASCII-Zeichenwerten dem Programm zu übergeben (wie die Kommandoeingabe des CCP). Dazu ist ein Speicherbereich in der TPA als Puffer (Buffer) zu reservieren und durch das DE-Register zu adressieren. Alle Zeichen, die über die Tastatur eingegeben werden (mit Echo), befinden sich dann in dem adressierten Puffer. Die Größe des Puffers ist frei zwischen 1 und 255 Zeichen festzulegen (1.Byte des Puffers). Die Eingabe wird beendet, wenn ein Carriage Return (ENTER-Taste oder ^M) oder ein Line Feed (^L) eingegeben wird oder aber der Puffer mehr Zeichenwerte aufnehmen soll als er kann (Pufferüberlauf). Die Anzahl der gültigen Zeichen im Puffer zeigt nach Funktionsende das 2.Byte des Puffers an. Der Puffer besitzt daher folgenden Aufbau

1.Byte: Maximale Puffergröße (1-255 Bytes). Muß vom Benutzer gesetzt werden.

2.Byte: Hier liegt nach Ausführung der Funktion die Anzahl der eingegebenen Zeichen (0 bis maximale Puffergröße).

3.Byte: Beginn des eigentlichen Zeichenpuffers. Ab hier sind mindestens so viele Bytes zu reservieren, wie aufgenommen werden sollen.

Mit dieser BDOS-Funktion steht eine komfortable Routine zur Verfügung, um Informationen dem Programm zu übergeben, die anschließend auf Kriterien geprüft werden sollen.

Übergabe-Parameter:

Eingang: C-Register = 0AH
DE-Register = Pufferadresse

Ausgang: ASCII-Zeichenfolge im adressierten Puffer

Beispiel:

Liest eine komplette (Kommando-) Zeile von der Tastatur in den adressierten Puffer ein. Die Puffergröße ist hier auf 32 Zeichen (20H, Adresse 0127H) festgelegt. Nach der Eingabe wird zunächst ein Zeilenvorschub ausgelöst, dann wird der Inhalt des Puffers zur Kontrolle noch einmal auf dem Bildschirm ausgegeben.

```
                ;*******************************************************
                ;*                                                     *
                ;*              BDOS-Funktion No. 10                   *
                ;*Zeichen in den Puffer einlesen (read console buffer)*
                ;*                                                     *
                ;*    <C>=0AH (in)       <DE>=Pufferadresse (in)      *
                ;*    Zeichen liegen im Puffer (out)                  *
                ;*                                                     *
                ;*******************************************************
                ;
0005 =          BDOS      EQU     0005H
0002 =          CONOUT    EQU       02H
0009 =          STROUT    EQU       09H
000A =          BUFIN     EQU       0AH
                ;
0100                      ORG     0100H
0100 112701               LXI  D,PUFFER ;<DE>=Pufferadresse
0103 0E0A                 MVI   C,BUFIN ;<C>=0AH
0105 CD0500               CALL      BDOS ;Puffer einlesen. Eingabeende
                                         ;mit CR oder Ueberlauf
                ;
0108 112401               LXI  D,STRING ;CR und LF ausgeben
010B 0E09                 MVI  C,STROUT
010D CD0500               CALL      BDOS
                ;
                ;Die Zeichen aus dem Puffer wieder auslesen/ausgeben
                ;
0110 212801               LXI H,POINTER ;Anzahl der vorhandenen Zeichen
0113 46                   MOV       B,M ;ins <B> laden
0114 23         LOOP      INX         H ;<HL>=<HL>+1
0115 5E                   MOV       E,M ;<E>=adressiertes Byte (HL)
0116 0E02                 MVI  C,CONOUT ;<C>=02H (Zeichen ausgeben)
0118 C5                   PUSH        B ;<BC> und <HL> merken da nach
0119 E5                   PUSH        H ;BDOS-Sprung undeferent
011A CD0500               CALL     BDOS
011D E1                   POP         H ;<BC> und <HL> wieder holen
011E C1                   POP         B
011F 05                   DCR         B ;<B>=<B>-1
0120 C21401               JNZ      LOOP ;Alle Zeichen ? Nein, springe
0123 C9                   RET
                ;
0124 0A0D24     STRING    DB 0AH,0DH,'$';CR und LF
                ;
0127 20         PUFFER    DB 20H         ;Max. Speicherbedarf (32)
0128 10         POINTER   DB 10H         ;Nach lesen Anzahl d.g. Zeichen
0129                      DS 20H         ;Hier liegt die Zeichenfolge
0149            END
```

6.12 BDOS-Funktion 11: Get Console Status (CONSTAT)

Prüft, ob an der Tastatur (Konsole) ein Zeichen ansteht. Der Akkumulator enthält den Wert 00H, falls kein Zeichen ansteht (sonst 0FFH). Mit dieser Funktion kann z.B. auf die Betätigung einer Taste gewartet werden (siehe Beispiel).

Übergabe-Parameter:

Eingang:	C-Register	= 0BH
Ausgang:	Akkumulator	= 00H wenn kein Zeichen ansteht
		= 0FFH wenn Zeichen ansteht

Beispiel:

Es wird eine Meldung auf dem Bildschirm ausgeben (Weiter mit einer Taste!). Innerhalb der Schleife WARTE kreist das Programm solange, bis ein Zeichen von der Tastatur ansteht (Akkumulator < > 00H).

```
                ;*****************************************************
                ;*                                                   *
                ;*              BDOS-Funktion No. 11                 *
                ;*    Prueft ob ein Zeichen ansteht (console status) *
                ;*                                                   *
                ;*    <C>=0BH (in)       <A>=Status (out)           *
                ;*                                                   *
                ;*****************************************************
                ;
0005 =          BDOS      EQU     0005H
0009 =          STROUT    EQU       09H
000B =          CONSTAT   EQU       0BH
                ;
0100                      ORG     0100H
0100 111301               LXI   D,STRING ;<DE>=Stringadresse
0103 0E09                 MVI   C,STROUT ;String ausgeben
0105 CD0500               CALL     BDOS
                ;
0108 0E0B       WARTE     MVI C,CONSTAT ;Warte auf ein Zeichen
010A CD0500               CALL     BDOS ;von der Tastatur
010D FE00                 CPI       00H ;<A>=00H kein Zeichen
010F CA0801               JZ      WARTE ;<A>=00H ? Ja, springe
0112 C9                   RET
                ;
0113 5765697465 STRING    DB 'Weiter mit einer Taste !$'
012C            END
```

6.13 BDOS-Funktion 12: Get Versions Number (GETNO)

Holt in das HL-Register die Informationen, um welches Betriebssystem es sich handelt (H-Register), sowie die Versionsnummer (L-Register). Die Inhalte der beiden Register unterscheiden sich wie folgt

H-Register	= 00H	:CP/M
	= 01H	:MP/M
L-Register	= 00H	:Version vor 2.0
	= 20H	:Version 2.0
	= 21H	:Version 2.1
	...	.....
	= 2FH	:Version 2.15

Übergabe-Parameter:

Eingang:	C-Register	= 0CH
Ausgang:	H-Register	= System
	L-Register	= Versionsnummer

6.14 BDOS-Funktion 13: Disc Reset (RESET)

Ermöglicht das Rücksetzen aller angeschlossenen Diskettenstationen (R/W, Laufwerk A wird ausgewählt und der DMA-Speicher auf 0080H gesetzt). Diese Funktion sollte nach jedem programmbedingten Diskettenwechsel ausgeführt werden!!

Übergabe-Parameter:

Eingang:	C-Register = 0DH
Ausgang:	Keine

Beispiel:

Es wird eine Meldung auf dem Bildschirm ausgegeben (‚Diskette wechseln, dann eine Taste !') und auf die Betätigung einer Taste gewartet (siehe BDOS-Funktion 11). Anschließend wird ein Diskettenreset durchgeführt.

```
                ;*****************************************************
                ;*                                                   *
                ;*              BDOS-Funktion No. 13                 *
                ;*    Fuehrt ein Diskettenreset durch (disc reset)   *
                ;*                                                   *
                ;*    <C>=00H                                        *
                ;*                                                   *
                ;*****************************************************
                ;
0005 =          BDOS      EQU     0005H
0009 =          STROUT    EQU       09H
000B =          CONSTAT   EQU       0BH
000D =          RESET     EQU       0DH
                ;
0100                      ORG     0100H
0100 111801               LXI   D,STRING ;Meldung ausgeben
0103 0E09                 MVI   C,STROUT
0105 CD0500               CALL      BDOS
                ;
0108 0E0B       WARTE     MVI C,CONSTAT ;Warte auf ein Zeichen
010A CD0500               CALL      BDOS ;von der Tastatur
010D FE00                 CPI         00H ;<A>=00H kein Zeichen
010F CA0801               JZ        WARTE ;<A>=00H ? Ja, springe
                ;
0112 0E0D                 MVI    C,RESET ;Discreset durchfuehren
0114 CD0500               CALL      BDOS
0117 C9                   RET
                ;
0118 4469736B65 STRING    DB 'Diskette wechseln, dann eine Taste !$'
013D            END
```

6.15 BDOS-Funktion 14: Select Disc (SELECT)

Wählt ein durch das E-Register bestimmtes Laufwerk aus. Hiernach beziehen sich alle Diskettenzugriffe auf das gewählte Laufwerk (bis zum Warmstart).

E-Register	=	00H	Laufwerk A
	=	01H	Laufwerk B
		...	..... .
	=	0FH	Laufwerk P

Falls das gewünschte Laufwerk nicht vorhanden ist, erfolgt eine Fehlermeldung (siehe Beispiel).

Übergabe-Parameter:

Eingang:	C-Register = 0EH E-Register = Laufwerk
Ausgang:	Keine

Beispiel:

Es wird das Laufwerk B als aktuelles Laufwerk ausgewählt (E-Register = 01H). Ist es nicht angeschlossen, erfolgt die Fehlermeldung ‚DISC MISSING'.

```
                ;*************************************************
                ;*                                               *
                ;*            BDOS-Funktion No. 14               *
                ;*     Aktuelles Laufwerk wechseln (select disc) *
                ;*                                               *
                ;*    <C>=0EH (in)        <E>=Laufwerk (in)      *
                ;*                                               *
                ;*************************************************
                ;
0005 =          BDOS    EQU     0005H
000E =          SELECT  EQU       0EH
                ;
0100                    ORG     0100H
0100 1E01               MVI     E,01H ;Laufwerk B (A=0,B=1 ...)
0102 0E0E               MVI  C,SELECT ;auswaehlen und
0104 CD0500             CALL     BDOS ;wechseln
0107 C9                 RET
0108            END
```

6.16 BDOS-Funktion 15: Open File (OPEN)

Diese Funktion bereitet den Zugriff auf eine bereits vorhandene Datei vor. In einem korrekt angelegten FCB (siehe 5.4), der durch das DE-Register adressiert ist, befinden sich die Informationen, um welche Datei es sich handelt (Name, Typ). Nach Aufruf der Funktion prüft das BDOS, ob sich auf der Diskette die angegebene Datei befindet. Wird sie gefunden, legt das BDOS alle wichtigen Informationen der Datei in den FCB (damit kann die Datei gelesen oder beschrieben werden). Ist die gewünschte Datei nicht vorhanden, enthält der Akkumulator den Wert 0FFH, andernfalls einen Wert zwischen 00H und 03H. Die Benutzung von Wildcards ist zulässig, wobei dann die erste relevante Datei ausgewählt wird.

Übergabe-Parameter:

Eingang:	C-Register	=	0FH
	DE-Register	=	FCB-Adresse
Ausgang:	Akkumulator	=	00H bis 03H (gefunden)
		=	0FFH (nicht gefunden)

Beispiel:

Siehe Programm HEXDUMP, Kapitel 7.2.

6.17 BDOS-Funktion 16: Close File (CLOSE)

Jede Datei, die mit der BDOS-Funktion 15 (OPEN) oder 22 (MAKE) eröffnet und bearbeitet wurde, muß mit der Funktion CLOSE wieder ordnungsgemäß geschlossen werden (wird eine Datei nur gelesen, ist das Schließen der Datei nicht unbedingt notwendig). Dabei erfolgt eine Aktualisierung des Directory-Eintrags. Der Wert des Akkumulators zeigt wieder an, ob das Schließen der Datei erfolgreich durchgeführt wurde (0FFH = Datei nicht gefunden, 00H bis 03H ordnungsgemäß geschlossen).

Übergabe-Parameter:

Eingang:	C-Register	=	10H
	DE-Register	=	FCB-Adresse
Ausgang:	Akkumulator	=	00H bis 03H (ordnungsgemäß geschlossen)
		=	0FFH (Schließen nicht möglich)

Beispiel:

Siehe Programm HEXDUMP, Kapitel 7.2, Programmteil ENDE.

6.18 BDOS-Funktion 17: First Directory (FIRST)

Durchsucht die Directory nach dem ersten passenden Eintrag. In dem FCB (adressiert durch das DE-Register) befindet sich der Name der gesuchten Datei (Wildcards sind zulässig). Ist der gewünschte Eintrag nicht vorhanden, besitzt der Akkumulator den Wert 0FFH. Ist der Eintrag vorhanden, liegt im Akkumulator ein Wert zwischen 00H und 03H und gleichzeitig kopiert das BDOS den entsprechenden Directory-Record (128 Bytes) in die DMA. Jeder Directory-Record besitzt die Informationen über 4 Einträge (je 32 Bytes). Anhand des Akkumulatorwertes ist die Position des gesuchten Eintrags innerhalb der DMA zu ermitteln (Akkumulator*32).

Akkumulator	= 00H	:Eintrag ab Byte 0
	= 01H	:Eintrag ab Byte 32
	= 02H	:Eintrag ab Byte 64
	= 03H	:Eintrag ab Byte 96

Übergabe-Parameter:

Eingang:	C-Register	= 11H
	DE-Register	= FCB-Adresse
Ausgang:	Akkumulator	= 00H bis 03H (gefunden)
		= 0FFH (nicht gefunden)

(DMA mit Directory-Record gefüllt, wenn Akkumulator < > 0FFH)

Beispiel:

Siehe Programm XDIR, Kapitel 7.3, Adresse 0100H bis 010AH.

6.19 BDOS-Funktion 18: Next Directory (NEXT)

Diese Funktion arbeitet wie die BDOS-Funktion 17, sucht aber nach einem weiteren relevanten Eintrag in der Directory (wichtig bei der Benutzung von Wildcards). Ist kein weiterer Eintrag vorhanden, besitzt der Akkumulator den Wert 0FFH, andernfalls liegt der Wert zwischen 00H und 03H und bestimmt damit die Position des Eintrags in der DMA (siehe BDOS-Funktion 17, FIRST). Gleichzeitig wird dann der entsprechende Directory-Record in die DMA geladen.

Übergabe-Parameter:

Eingang:	C-Register	=	12H
	DE-Register	=	FCB-Adresse
Ausgang:	Akkumulator	=	00H bis 03H (gefunden)
		=	0FFH (kein Eintrag mehr vorhanden)

(DMA mit Directory-Record gefüllt, wenn Akkumulator < > 0FFH)

Beispiel:

Siehe Programm XDIR, Kapitel 7.3, Programmteil WEITER.

6.20 BDOS-Funktion 19: Delete File (ERASE)

Alle Files, deren Namen sich mit dem Eintrag des FCB (adressiert durch das DE-Register) decken, werden aus der Directory gestrichen (gelöscht). Sind Wildcards benutzt, werden alle relevanten Files gelöscht. Das erste Byte des entsprechenden Directory-Eintrags besitzt dann als ‚Löschkennzeichen' den Wert 0E5H (siehe auch 7.3). Ist der gewünschte File nicht vorhanden, enthält der Akkumulator den Wert 0FFH, andernfalls einen Wert zwischen 00H und 03H (siehe auch BDOS-Funktion 17). Diese Routine entspricht dem ERA-Kommando.

Übergabe-Parameter:

Eingang:	C-Register	=	13H
	DE-Register	=	FCB-Adresse
Ausgang:	Akkumulator	=	00H bis 003H (gelöscht)
		=	0FFH (löschen nicht möglich)

Beispiel:

Löscht den oder die Files, deren Namen (und Typ) mit dem Eintrag im FCB übereinstimmen, aus der Directory. Der FCB wird beim Aufruf über den CCP eigenständig ab Adresse 005CH angelegt. Das Programm wird mit

```
A>BDOS19 filename.typ
```

aufgerufen. ACHTUNG: Die Verwendung von Wildcards ist zulässig!!

```
                ;*****************************************************
                ;*                                                   *
                ;*              BDOS-Funktion No. 19                 *
                ;*             Datei loeschen (erase)                *
                ;*                                                   *
                ;*    <C>=13H (in)       <DE>=FCB-Adresse (in)       *
                ;*    <A>=Directory-Code (out)                       *
                ;*                                                   *
                ;*****************************************************
                ;
0005 =          BDOS    EQU     0005H
0009 =          STROUT  EQU       09H
0013 =          ERASE   EQU       13H
005C =          FCB     EQU     005CH ;Standard vom CCP angelegt
                ;
```

```
0100                    ORG     0100H
0100 115C00             LXI     D,FCB ;Adresse des FCB in <DE> laden
0103 0E13               MVI   C,ERASE ;loeschen vornehmen
0105 CD0500             CALL      BDOS
0108 FEFF               CPI       0FFH ;<A>=0FFH dann Fehler
010A CA1601             JZ      LABEL1 ;und springe
010D 111D01             LXI D,STRING1 ;OK. Meldung ausgeben
0110 0E09       LABEL2  MVI   C,STROUT
0112 CD0500             CALL      BDOS
0115 C9                 RET
                ;
0116 113A01     LABEL1  LXI D,STRING2 ;Fehlermeldung
0119 C31001             JMP     LABEL2 ;ausgeben
011C C9                 RET
                ;
011D 4F4B2E2046 STRING1 DB 'OK. File(s) sind geloescht !$'
013A 46696C6528 STRING2 DB 'File(s) nicht vorhanden !$'
0154            END
```

6.21 BDOS-Funktion 20: Sequential Read (SEQREAD)

Aus einer Datei (bestimmt durch den FCB), die entweder mit der BDOS-Funktion 15 eröffnet oder mit der BDOS-Funktion 22 kreiert (erzeugt) ist, wird der nächste, 128 Bytes große Sektor in die DMA gelesen. Gleichzeitig wird der FCB für den nächsten Zugriff vorbereitet. War der Lese-Zugriff erfolgreich, besitzt der Akkumulator den Wert 00H. Ist das Ende des Files erreicht, befindet sich im Akkumulator ein Wert ungleich 00H.

Übergabe-Parameter:

Eingabe:	C-Register	=	14H
	DE-Register	=	FCB-Adresse
Ausgabe:	Akkumulator	=	00H (Lese-Zugriff war erfolgreich) <>00H (Dateiende erreicht)
(Ist der Akkumulator		=	00H; liegt der gelesene Record in der DMA)

Beispiel:

Siehe Programm HEXDUMP, Kapitel 7.2, Programmteil MAINPRG.

6.22 BDOS-Funktion 21: Sequential Write (SEQWRITE)

Arbeitet identisch wie die BDOS-Funktion 20, schreibt allerdings den Inhalt der DMA in den nächsten Block der offenen (!!) Datei. Gleichzeitig wird der FCB auf den nächsten Schreib-/Lesezugriff vorbereitet. War das Beschreiben erfolgreich, besitzt der Akkumulator den Wert 00H. Ist der Akkumulatorwert ungleich 00H, zeigt dieses eine volle Diskette an. Nach Beendigung der Schreiboperationen muß die Datei ordnungsgemäß geschlossen werden (siehe BDOS-Funktion 16) !!

Übergabe-Parameter:

Eingang:	C-Register	=	15H
	DE-Register	=	FCB-Adresse
Ausgang:	Akkumulator	=	00H (Record erfolgreich beschrieben)
			<>00H (Diskette voll)

Beispiel:

Siehe BDOS-Funktion 22, Kapitel 6.23, Programmteil SCHREIB.

6.23 BDOS-Funktion 22: Make File (MAKE)

Erstellt (kreiert) eine neue, nicht auf der Diskette vorhandene Datei, deren Name in dem FCB abgelegt und durch das DE-Register adressiert wird. Beim Erstellen einer Datei ist umbedingt darauf zu achten, daß der (neue) Dateiname sich nicht schon auf der Diskette befindet, da dieses von der Funktion nicht geprüft wird (ggf. löschen) !!! Eine neu erstellte Datei muß nicht noch zusätzlich eröffnet werden (BDOS-Funktion 15), da die BDOS-Funktion 22 alle notwendigen Vorbereitungen im FCB trifft. Ein erfolgreiches Erstellen wird durch die Werte 00H bis 03H im Akkumulator angezeigt. Enthält der Akkumulator den Wert 0FFH, ist die Diskette voll.

Übergabe-Parameter:

Eingang:	C-Register	=	16H
	DE-Register	=	FCB-Adresse
Ausgang:	Akkumulator	=	00H bis 03H (kreieren erfolgreich)
		=	0FFH (Diskette voll, nicht kreiert)

Beispiel:

Das Beispiel legt eine neue Datei unter dem Namen ‚HALLO.ASC' auf dem aktuellen Laufwerk an. Der FCB ist fest ab Adresse 0134H angelegt, ebenfalls die DMA (ab Adresse 0155H) mit genau 128 ASCII-Zeichen (1 Record). Nachdem die Datei erfolgreich kreiert ist, wird der Inhalt der DMA in die Datei geschrieben. Das Schließen der Datei (ENDE) ist umbedingt notwendig!! Achten Sie bitte vor Ausführung dieses Programms darauf, daß sich auf der Diskette keine Datei mit dem Namen ‚HALLO.ASC' befindet. Ist die Datei angelegt, können Sie diese mit

```
A>TYPE HALLO.ASC
```

auf dem Bildschirm (oder Drucker) auflisten.

```
                ;*******************************************************
                ;*                                                     *
                ;*              BDOS-Funktion No. 22                   *
                ;*      Erstellen einer neuen Datei (Make File)        *
                ;*                                                     *
                ;*    <C>=16H (in)      <DE>=FCB-Adresse (in)          *
                ;*    <A>=00H oder <> 00H (out)                        *
                ;*                                                     *
                ;*******************************************************
                ;
0005 =          BDOS      EQU     0005H
0009 =          STROUT    EQU       09H
0010 =          CLOSE     EQU       10H
0015 =          SEQWRIT   EQU       15H
0016 =          MAKE      EQU       16H
001A =          SETDMA    EQU       1AH
                ;
0100                      ORG     0100H
0100 115501               LXI     D,DMA ;<DE>=DMA-Adresse
0103 0E1A                 MVI   C,SETDMA ;neu setzen
0105 CD0500               CALL     BDOS
                ;
0108 0E16                 MVI    C,MAKE ;Neue Datei eroeffnen
010A 113401               LXI     D,FCB ;<DE>=FCB-Adresse
010D CD0500               CALL     BDOS ;und kreieren
0110 FE00                 CPI       00H ;<A>=00H dann Fehler
0112 C21E01               JNZ   SCHREIB ;OK. Dann springe
0115 11D501     ERROR     LXI   D,STRING ;Fehlermeldung
0118 0E09                 MVI   C,STROUT ;ausgeben
011A CD0500               CALL     BDOS
011D C9                   RET
                ;
011E 113401     SCHREIB   LXI     D,FCB ;<DE>=FCB-Adresse
0121 0E15                 MVI C,SEQWRIT ;
0123 CD0500               CALL     BDOS ;DMA auf Disk schreiben
0126 FE00                 CPI       00H ;<A>=00 dann Ok
0128 C21501               JNZ     ERROR
                ;
012B 0E10       ENDE      MVI   C,CLOSE ;Datei schliessen
012D 113401               LXI     D,FCB ;<DE>=FCB-Adresse
0130 CD0500               CALL     BDOS
0133 C9                   RET

0134 00         FCB       DB 00H        ;Laufwerk
0135 48414C4C4F           DB 'HALLO   ' ;Name
013D 415343               DB 'ASC'      ;Typ
0140 00000000             DB 00H,00H,00H,00H
0144                      DS 10H
0154 00                   DB 00H

0155 4869657220 DMA       DB 'Hier liegt die DMA.Es ist ein Puffer '
017A 696E206465           DB 'in der Groesse von 128 Bytes,der hier'
019F 6665737420           DB 'fest mit Werten gefuellt ist.Das muss'
01C4 2061626572           DB ' aber nicht sein.'

01D5 4665686C65 STRING    DB 'Fehler$'
01DC            END
```

6.24 BDOS-Funktion 23: Rename File (RENAME)

Umbenennen einer Datei, deren Name sich in den ersten 16 Bytes des FCBs (adressiert durch das DE-Register) befindet. Der neue Name liegt in den folgenden 16 Bytes des FCBs. War das Umbenennen erfolgreich, besitzt der Akkumulator einen Wert zwischen 00H und 03H. Ist der Akkumulatorinhalt gleich 0FFH, konnte die Datei nicht gefunden werden. Eine Benutzung von Wildcards ist nicht zulässig. Werden Wildcards benutzt, enthält der Akkumulator ebenfalls den Wert 0FFH.

Übergabe-Parameter:

Eingang:	C-Register	=	17H
	DE-Register	=	FCB-Adresse
Ausgang:	Akkumulator	=	00H bis 03H (Umbenennen erfolgreich)
		=	0FFH (File nicht gefunden oder Wildcards)

Beispiel:

Der FCB wird beim Aufruf vom CCP automatisch angelegt (Standard FCB-Adresse), so daß in diesem Fall keine Vorbereitungen des FCBs notwendig sind. Das Programm wird dann mit

```
A>BDOS23 altname.typ neuname.typ
```

aufgerufen.

```
                ;*****************************************************
                ;*                                                   *
                ;*              BDOS-Funktion No. 23                 *
                ;*            Datei umbenennen (rename)              *
                ;*                                                   *
                ;*    <C>=17H (in)       <DE>=FCB-Adresse (in)       *
                ;*    <A>=Directory-Code (out)                       *
                ;*                                                   *
                ;*****************************************************
                ;
0005 =          BDOS      EQU     0005H
0009 =          STROUT    EQU       09H
0017 =          RENAME    EQU       17H
005C =          FCB       EQU     005CH ;Standard vom CCP angelegt
                ;
```

```
0100                    ORG     0100H
0100 115C00             LXI     D,FCB ;Adresse des FCB in <DE> laden
0103 0E17               MVI   C,RENAME ;Umbenennung vornehmen
0105 CD0500             CALL      BDOS
0108 FEFF               CPI       0FFH ;<A>=0FFH dann Fehler
010A CA1601             JZ      LABEL1 ;und springe
010D 111D01             LXI D,STRING1 ;Meldung ausgeben
0110 0E09       LABEL2  MVI   C,STROUT
0112 CD0500             CALL      BDOS
0115 C9                 RET
                ;
0116 113201     LABEL1  LXI D,STRING2 ;Fehlermeldung
0119 C31001             JMP     LABEL2 ;ausgeben
011C C9                 RET
                ;
011D 4F4B2E2046 STRING1 DB 'OK. File umbenannt !$'
0132 46696C6520 STRING2 DB 'File nicht vorhanden !$'
0149            END
```

6.25 BDOS-Funktion 24: Get Login Vector (LOGIN)

Holt ein Verzeichnis aller aktiven Laufwerke in das HL-Register. Dabei repräsentiert jedes Bit eines der 16 möglichen (beim Schneider CPC nur 2) Laufwerke. Ein gesetztes Bit (‚1') kennzeichnet ein aktives, ein gelöschtes Bit (‚0') ein inaktives Laufwerk.

L-Register: Bit 0 = Laufwerk A
Bit 1 = Laufwerk B
. . . . . .
Bit 7 = Laufwerk H

H-Register: Bit 0 = Laufwerk I
Bit 1 = Laufwerk J
. . . . . .
Bit 7 = Laufwerk P

Übergabe-Parameter:

Eingang: C-Register = 18H

Ausgang: HL-Register = Zustand der Laufwerke A bis P

6.26 BDOS-Funktion 25: Return Current Disc (CURRENT)

Holt die Bezeichnung des aktuellen Laufwerks. Der Akkumulator enthält einen Wert zwischen 0 und 15 und kennzeichnet damit das momentan aktuelle Laufwerk.

Akkumulator

=00H : Laufwerk A aktiv
=01H : Laufwerk B aktiv
. . . . . .
=0FH : Laufwerk P aktiv

Übergabe-Parameter:

Eingang: C-Register = 19H

Ausgang: Akkumulator = Aktuelles Laufwerk

Beispiel:

Der Wert für das aktuelle Laufwerk wird geholt, anschließend in ein ASCII-Zeichen zwischen ‚A' und ‚P' umgeformt (Akkumulatorinhalt +41H) und dann auf dem Bildschirm ausgegeben.

```
                ;*********************************************************
                ;*                                                       *
                ;*              BDOS-Funktion No. 25                     *
                ;*     Hole aktuelle Laufwerknummer (current disc)       *
                ;*                                                       *
                ;*    <C>=19H (in)        <A>=aktuelles Laufwerk (out)   *
                ;*                                                       *
                ;*********************************************************
                ;
0005 =          BDOS      EQU     0005H
0002 =          CONOUT    EQU       02H
0009 =          STROUT    EQU       09H
0019 =          CURRENT   EQU       19H
                ;
0100                      ORG     0100H
0100 111601               LXI   D,STRING ;Meldung ausgeben
0103 0E09                 MVI   C,STROUT
0105 CD0500               CALL      BDOS
                ;
0108 0E19                 MVI C,CURRENT ;Hole
010A CD0500               CALL      BDOS ;Aktuelle Laufwerknr. in <A>
010D C641                 ADI        41H ;<A> nach ASCII 'A'-'Z' wandeln
010F 5F                   MOV        E,A ;<E>=<A>
```

```
0110 0E02                MVI  C,CONOUT ;Zeichen <E> ausgeben
0112 CD0500              CALL     BDOS
0115 C9                  RET
                    ;
0116 416B747565 STRING   DB 'Aktuelles Laufwerk : $'
012C            END
```

6.27 BDOS-Funktion 26: Address (SETDMA)

Setzt den DMA-Puffer auf einen neuen, durch das DE-Register adressierten Speicherbereich. Die neue DMA-Adresse bleibt bis zu einem Kaltstart, Warmstart oder Rücksetzen des Disksystems aktuell. Im Einschaltzustand (Kalt/Warmstart, Rücksetzen) befindet sich die DMA automatisch ab Adresse 0080H (Standard DMA-Adresse).

Übergabe-Parameter:

Eingang: C-Register = 1AH
DE-Register = DMA-Adresse

Ausgang: Keine

Beispiel:

Im Beispielprogramm zur BDOS-Funktion 22 (siehe 6.23) finden Sie in den Adressen 0100H bis 0105H ein typisches Beispiel zum Setzen der DMA-Adresse.

6.28 BDOS-Funktion 27: Get Allocation Address (ALLOC)

Holt die Startadresse der sogenannten ‚Blockbelegungstabelle' in das HL-Register. Hier befindet sich eine Tabelle der benutzten und unbenutzten Blöcke der aktuellen Diskette (insgesamt 170 Bits). Die Blockbelegungstabelle wird von CP/M immer dann angelegt, wenn ein Kalt- oder Warmstart oder ein Diskettenreset ausgeführt wird. Die 21 Byte große Tabelle wird bitweise interpretiert: Ein gesetztes Bit (‚1') kennzeichnet einen belegten Block, ein gelöschtes Bit (‚0') einen freien Block. Anhand der Blockbelegungstabelle entscheidet das CP/M, in welchen (freien) Blöcken z.B. ein Programm gespeichert werden kann. Beim Löschen eines Files werden die entsprechenden Blöcke wieder als frei gekennzeichnet (auf ‚0' gesetzt). Das erste Bit der Tabelle ist immer ‚1' und kennzeichnet die Directory. Bei einer neuformatierten Disketten wären alle Bits (außer dem ersten Bit) ‚0', da noch kein Block von der Diskette belegt ist.

Übergabe-Parameter:

Eingang:	C-Register	=	1BH
Ausgang:	HL-Register	=	Adresse des Blockbelegungsverzeichnis

6.29 BDOS-Funktion 28: Write Protect Disc (PROTECT)

Setzt das aktuelle Laufwerk auf ‚Read only' (schreibgeschützt). Dieser Schreibschutz bleibt bis zu einem Kalt-/Warmstart oder dem Rücksetzen der Diskette aktiv. Wird versucht, auf diese Diskette zu schreiben, erfolgt die Fehlermeldung

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

Übergabe-Parameter:

Eingang: C-Register = 1CH

Ausgang: Keine

Beispiel:

Wenn Sie das Beispielprogramm ausgeführt haben, versuchen Sie bitte einmal etwas auf der Diskette zu sichern (z.B. A>SAVE 1 TEST.COM). Sie werden die Fehlermeldung ‚BDOS ERR ON d:R/O' erhalten (Rücksetzen mit ^C).

```
                ;*************************************************
                ;*                                               *
                ;*            BDOS-Funktion No. 28               *
                ;*      Schreibschutzt setzen (write protect)    *
                ;*                                               *
                ;*    <C>=1CH (in)                               *
                ;*                                               *
                ;*************************************************
                ;
0005 =          BDOS      EQU     0005H
001C =          PROTECT   EQU       1CH
                ;
0100                      ORG     0100H
0100 0E1C                 MVI C,PROTECT ;Aktuelles Laufwerk wird
0102 CD0500               CALL      BDOS ;schreibgeschuetzt (R/O)
0105 C9                   RET
0106            END
```

6.30 BDOS-Funktion 29: Get R/O Vector (GETRO)

Nach Aufruf der Funktion zeigt das HL-Register an, welche Laufwerke (A-P) momentan schreibgeschützt sind. Jedes Bit des HL-Registers repräsentiert ein Laufwerk

L-Register Bit 0 = Laufwerk A
Bit 1 = Laufwerk B
. . . . . .
Bit 7 = Laufwerk H

H-Register Bit 0 = Laufwerk I
Bit 1 = Laufwerk J
. . . . . .
Bit 7 = Laufwerk P

Ist ein Bit gesetzt (‚1'), so ist das entsprechende Laufwerk momentan schreibgeschützt (Read Only, R/O).

Übergabe-Parameter:

Eingabe: C-Register = 1DH

Ausgabe: HL-Register = Zustand der Laufwerke A bis P

6.31 BDOS-Funktion 30: Set File Atributes (SETATR)

Mit Hilfe dieser BDOS-Funktion lassen sich die Attribute eines Files neu setzen. Dabei muß im FCB (durch das DE-Register adressiert) ein eindeutiger Dateiname mit gesetzten Attributen im Filetyp vorliegen

Byte 9	Bit 7 :	0 =	Datei Read/Write (R/W)
		1 =	Datei Read/Only (schreibgeschützt)
Byte 10	Bit 7 :	0 =	Datei wird in der Directory geführt
		1 =	SYS-Datei, wird nicht in der Directory geführt.

Die Bits 0 bis 6 der Bytes 9, 10 und 11 bestimmen dem Filetyp (siehe 5.3). Ein Setzen bzw. Löschen der Attribute ist mit folgenden Verknüpfungen möglich

(Byte) ORI 80H	Setzt Bit 7
(Byte) ANI 7FH	Löscht Bit 7

Die BDOS-Funktion durchsucht die Directory nach dem gewünschten Eintrag und ändert dann diesen Directory-Eintrag. War der Vorgang erfolgreich, befindet sich im Akkumulator ein Wert zwischen 00H und 03H, andernfalls der Wert 0FFH.

Übergabe-Parameter:

Eingang:	C-Register	=	1EH
	DE-Register	=	FCB-Adresse
Ausgang:	Akkumulator	=	00H bis 03H (Attribute gesetzt)
		=	0FFH (Setzen nicht möglich)

6.32 BDOS-Funktion 31: Get Disc Parameter (PARMS)

Nach dieser Funktion befindet sich im HL-Register die Startadresse einer 15 Bytes großen Tabelle (Disk-Parameter-Puffer), die alle wesentlichen Informationen über das aktuelle Laufwerk enthält (siehe auch STAT DSK:). Die Tabelle baut sich wie folgt auf

Byte 0+ 1 : Anzahl der Sektoren
Byte 2 : Sektoren pro Spur (2^Wert)
Byte 3 : Sektoren pro Block (Wert+1)
Byte 4 : Blockgröße ((Wert+1)*1kByte))
Byte 5+ 6 : Gesamtanzahl der Blöcke (16-Bit-Zahl)
Byte 7+ 8 : Maximale Anzahl der Directory-Einträge
Byte 9+10 : Belegungszustand der Directory
Byte 11+12 : Anzahl der zu prüfenden Einträge
Byte 13+14 : Track Offset (+gewünschte Tracknummer)

Übergabe-Parameter:

Eingang: C-Register = 1FH

Ausgang: HL-Register = Disk-Parameter-Pufferadresse

6.33 BDOS-Funktion 32: Get/Set User (USER)

Die Funktion erfüllt zwei verschiedene Aufgaben: Einmal setzt sie die aktuelle Benutzerebene (USER) und zum anderen holt sie die aktuelle Benutzerebene. Ist das E-Register mit dem Wert 0FFH gefüllt, so wird im Akkumulator die aktuelle Benutzerebene (00H bis 0FH) übergeben. Enthält das E-Register einen Wert zwischen 00H und 0FH, so schaltet die Funktion auf die entsprechende Ebene um.

Übergabe-Parameter:

Eingang:	C-Register	=	20H
	E-Register	=	USER-Nummer
	E-Register	=	0FFH (siehe Ausgang)
Ausgang:	Akkumulator	=	00H bis 0FH (USER-Nummer)

Beispiel:

Das Programm setzt das System zunächst auf die USER-Ebene 1 (Adresse 0100H bis 0104H) und liest anschließend die USER-Nummer erneut aus (umrechnen des Akkumulatorinhalts auf ein Zeichen zwischen ‚0' und ‚9').

```
                ;*****************************************************
                ;*                                                   *
                ;*              BDOS-Funktion No. 32                 *
                ;*        Benutzerebene setzen/holen (user)          *
                ;*                                                   *
                ;*    <C>=20H (in)       <E>=Benutzernr. o. FFH (in)  *
                ;*    <A>=aktuelle Benutzerebene (out)               *
                ;*                                                   *
                ;*****************************************************
                ;
0005 =          BDOS    EQU     0005H
0002 =          CONOUT  EQU       02H
0020 =          USER    EQU       20H
                ;
0100                    ORG     0100H
0100 1E01               MVI     E,01H ;Wenn <A> <>FFH dann auf die
0102 0E20               MVI     C,USER ;Ebene <A> wechseln
0104 CD0500             CALL      BDOS ;Hier auf USER 1
                ;
0107 1EFF               MVI     E,0FFH ;Usernummer holen
0109 0E20               MVI     C,USER
010B CD0500             CALL      BDOS ;<A>=Usernummer
                ;
```

```
010E C630                ADI       30H ;<A>=<A>+48  (ASCII)
0110 5F                  MOV       E,A ;<E>=<A>
0111 0E02                MVI  C,CONOUT ;Zeichen ausgeben 0-9
0113 CD0500              CALL      BDOS
0116 C9                  RET
0117            END
```

6.34 BDOS-Funktion 33: Read Random (RANREAD)

Ähnlich der BDOS-Funktion 15 (SEQREAD) arbeitet auch diese BDOS-Funktion. Allerdings ist es hierbei möglich, einen bestimmten Record der Datei (FCB adressiert durch das DE-Register) in die DMA einzulesen (Datei vorher mit der BDOS-Funktion 15 oder 22 eröffnen oder erstellen !!). Die gewünschte Recordnummer (0-65535) muß vom Benutzer vor dem Aufruf der Funktion in den Bytes 33 bis 35 des FCBs folgendermaßen abgelegt sein:

FCB-Byte 33 : Lowbyte der Recordnummer
FCB-Byte 34 : Highbyte der Recordnummer
FCB-Byte 35 : auf 00H setzen (zeigt ggf. einen Überlauf an)

Nach Aufruf der BDOS-Funktion enthält der Akkumulator den Wert 00H, wenn die Leseoperation erfolgreich war. Der Inhalt des Record befindet sich dann im aktuellen DMA-Puffer. Trat bei der Leseoperation ein Fehler auf, besitzt der Akkumulator einen von 00H abweichenden Wert

Akkumulator	= 00H	Leseoperation war erfolgreich
	= 01H	Unbeschriebener Record
	= 04H	Unbeschriebener Record
	= 06H	Es wurde versucht, einen Record zu lesen, der hinter dem physikalischen Ende der Datei liegt (Überlauf: Byte 35 des FCB ist ungleich 00H!!)

Übergabe-Parameter:

Eingabe:	C-Register	= 21H
	DE-Register	= FCB-Adresse
	FCB-Bytes 33, 34	= Recordnummer
	FCB-Byte 35	= 00H
Ausgabe:	Akkumulator	= 00H (Leseoperation OK)
		<>00H (siehe Fehlerliste)

(Die DMA ist mit dem gewünschten Record gefüllt)

6.35 BDOS-Funktion 34: Write Random (RANWRITE)

Schreibt den Inhalt der DMA in den durch die Bytes 33 und 34 des FCBs bestimmten Record (siehe BDOS-Funktion 33). Der FCB wird durch das DE-Register adressiert. Die zu bearbeitende Datei muß vorher geöffnet oder angelegt sein (BDOS-Funktion 15 oder 22). Der Akkumulator zeigt an, ob die Schreiboperation erfolgreich abgeschlossen wurde

Akkumulator	= 00H	Schreiboperation war erfolgreich
	= 05H	Datei läßt sich nicht erweitern, da die Directory voll belegt ist (neue Extension)

Übergabe-Parameter:

Eingang:	C-Register	=	22H
	DE-Register	=	FCB-Adresse
	FCB-Byte 33, 34	=	Recordnummer
	FCB-Byte 35	=	00H
Ausgang:	Akkumulator	=	00H (Schreiboperation erfolgreich) <>00H (Fehler)

6.36 BDOS-Funktion 35: Compute File Size (Size)

Ermittelt die Größe der aktuellen Datei, deren FCB durch das DE-Register adressiert ist. Nach Aufruf der Funktion zeigen die Bytes 33 und 34 (Low- und Highbyte) auf die Recordnummer, die dem Dateiende folgt (Endrecord+1). Ist das Überlaufbyte 35 des FCBs gleich 01H, so ist die maximale Dateigröße von 65535 Records erreicht. Diese Funktion bietet sich an, um weitere Daten an eine Datei ‚anzuhängen' oder die Größe einer Datei zu ermitteln.

Übergabe-Parameter:

Eingang:	C-Register	=	23H
	DE-Register	=	FCB-Adresse
Ausgang:	FCB-Bytes 33,34	=	Recordnummer (Ende+1)
	FCB-Byte 35	=	01H, dann max. Dateigröße erreicht

6.37 BDOS-Funktion 36: Set Random Record (SETREC)

Die BDOS-Funktion 36 errechnet die Nummer eines Records aus der aktuellen Position der sequentiellen Bearbeitung. Die Recordnummer befindet sich dann in den Bytes 33 und 34 des FCBs (Byte 35 = 00H). Mit dieser Funktion ist u.a. eine ‚Umschaltung' von der sequentiellen zur satzorientierten (wahlfreien) Bearbeitung möglich.

Übergabe-Parameter:

Eingang: C-Register = 24H
DE-Register = FCB-Adresse

Ausgang: Setzt die FCB-Bytes 33, 34 und 35

6.38 BDOS-Funktion 37: Drive Reset (DRIVE)

Bewirkt einen Reset aller durch das DE-Registers bestimmten Laufwerke. Jedes Bit des DE-Register (16-Bit-Wert) repräsentiert ein Laufwerk

D-Register Bit 0 = Laufwerk A
Bit 1 = Laufwerk B
. . . . . .
Bit 7 = Laufwerk H

E-Register Bit 0 = Laufwerk I
Bit 1 = Laufwerk J
. . . . . .
Bit 7 = Laufwerk P

Ist ein Bit gesetzt (‚1'), wird das entsprechende Laufwerk zurückgesetzt (Reset).

Übergabe-Parameter:

Eingang: C-Register = 24H
DE-Register = Laufwerkangabe

Ausgang: Keine

7.0 CP/M Programme

Nachdem alle BDOS-Funktionen beschrieben und an vielen Beispielen erläutert wurden, möchte ich an dieser Stelle zeigen, wie man doch sehr einfach auch umfangreichere Programme unter CP/M erstellen kann. Anhand von vier Programmen soll die Anwendung einiger BDOS-Funktionen vertieft und der Aufbau der Diskette grundlegend erklärt werden. Diejenigen unter Ihnen, die sich bisher noch gar keine Gedanken über den Aufbau der Diskette gemacht haben, möchte ich zunächst auf den 2. Band dieser Reihe verweisen, der sich schwerpunktmäßig mit der Diskettenstation beschäftigt (siehe Band 2, 10.3 bis 10.6). Aber auch für diejenigen unter Ihnen, die sich nicht so sehr für die Programmierung unter CP/M interessieren, bieten diese Programme eine unerläßliche Hilfe für das Verstehen des Schneider CPC-464/664 im CP/M-Modus.

Nach dem Quellisting folgt zusätzlich noch der HEX-Dump des jeweiligen Programms.

7.1 HELPER - Benutzerhilfe

Das erste Programm ist sehr einfach zu verstehen, da es ausschließlich die Aufgabe hat, einen Text auf dem Bildschirm auszugeben. Der Text zeigt die Tastenbelegung der CP/M-Steuerzeichen und ihrer Funktionen an. Der Text befindet sich in einem geschützten Bildschirmbereich (WINDOW) und bleibt somit ständig auf dem Bildschirm bestehen. Weiterhin werden die Zeichen- und Bildschirmfarben (Schwarz/Weiß) neu gesetzt.

Der Ausgabe-String liegt ab der Adresse 010AH und endet ordnungsgemäß mit dem Dollar-Zeichen ('$') in Adresse 034CH. Am Anfang des Strings befinden sich eine Reihe von CPC-Steuerzeichen (siehe 1.5), die folgende Funktionen ausführen

```
04H,02H          : MODE 2 (80-Zeichen-Modus)
1CH,00H,00H,00H  : INK 0, 0, 0 (Faß 0 mit Schwarz füllen)
1CH,01H,1AH,1AH: INK 1,26,26 (Faß 1 mit Weiß füllen)
1DH,00H,00H      : BORDER 0, 0 (Borderfarbe Schwarz)
```

Anschließend folgt der Text (ggf. Leerzeichen), welcher in Adresse 0346H endet. Abschließend wird die Größe des Fensters 0 neu gesetzt

```
1AH,01H,50H,0BH,19H : WINDOW#0,1,80,11,25
```

Alle weiteren Bildschirmausgaben finden nun ausschließlich in der unteren Hälfte des Bildschirmes statt und der Hilfstext bleibt somit ständig bestehen. Das Fenster kann mit

```
^Db
```

jederzeit aufgehoben werden (^Db entspricht MODE 2).

```
                ;*****************************************************
                ;*                                                   *
                ;*                      H E L P                      *
                ;*        Zeigt staendig alle Steuerzeichen an       *
                ;*                                                   *
                ;*****************************************************
                ;
0005 =          BDOS    EQU     0005H
0009 =          STROUT  EQU       09H
                ;
0100                    ORG     0100H
0100 110A01             LXI  D,STRING ;<DE>=Stringadresse
0103 0E09               MVI  C,STROUT ;String ausgeben
0105 CD0500             CALL     BDOS
```

```
0108 C9                 RET
0109 C9                 RET
                ;
010A 04021C0000 STRING  DB 04H,02H,1CH,00H,00H,00H,1CH,01H,1AH,1AH
0114 1D0000             DB 1DH,00H,00H
0117 48454C5045         DB 'HELPER VER 1.0',0AH,0DH,0AH
0128 5E43203A20         DB '^C : Rueckkehr zur Kommandoebene        '
0150 5E45203A20         DB '^E : Kommando i.d.n. Zeile fortsetzen   '
0178 5E48203A20         db '^H : Loescht Zeichen links vom Cursor   '
01A0 5E49203A20         DB '^I : Cursor zum naechsten TAB-Stop      '
01C8 5E4A203A20         DB '^J : Kommando abschliessen <cr>         '
01F0 5E4D203A20         DB '^M : Kommando abschliessen <cr>         '
0218 5E50203A20         DB '^P : Ausgabe umschalten (Drucker/Video) '
0240 5E52203A20         DB '^R : Kommandozeile komplett zeigen      '
0268 5E53203A20         DB '^S : Unterbricht laufende Anzeige       '
0290 5E55203A20         DB '^U : Bricht Kommandozeile ab            '
02B8 5E56203A20         DB '^V : Bricht Kommandozeile ab            '
02E0 5E58203A20         DB '^X : Loescht Kommandozeile              '
0308 0A0D               DB 0AH,0DH
030A 2020202020         DB '             <d:> <DIR>  ERA>  <REN>  '
0331 3C54595045         DB '<TYPE>  <USER>  <SAVE>'
0347 1A01500B19         DB 1AH,01H,50H,0BH,19H,'$'
034D            END

A>
```

```
ADR>  -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F 0123456789ABCDEF
0100  11 0A 01 0E 09 CD 05 00 C9 C9 04 02 1C 00 00 00 ................
0110  1C 01 1A 1A 1D 00 00 48 45 4C 50 45 52 20 56 45 .......HELPER VE
0120  52 20 31 2E 30 0A 0D 0A 5E 43 20 3A 20 52 75 65 R 1.0...^C : Rue
0130  63 6B 6B 65 68 72 20 7A 75 72 20 4B 6F 6D 6D 61 ckkehr zur Komma
0140  6E 64 6F 65 62 65 6E 65 20 20 20 20 20 20 20 20 ndoebene
0150  5E 45 20 3A 20 4B 6F 6D 6D 61 6E 64 6F 20 69 2E ^E : Kommando i.
0160  64 2E 6E 2E 20 5A 65 69 6C 65 20 66 6F 72 74 73 d.n. Zeile forts
0170  65 74 7A 65 6E 20 20 20 5E 48 20 3A 20 4C 6F 65 etzen   ^H : Loe

ADR>  -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F 0123456789ABCDEF
0180  73 63 68 74 20 5A 65 69 63 68 65 6E 20 6C 69 6E scht Zeichen lin
0190  6B 73 20 76 6F 6D 20 43 75 72 73 6F 72 20 20 20 ks vom Cursor
01A0  5E 49 20 3A 20 43 75 72 73 6F 72 20 7A 75 6D 20 ^I : Cursor zum
01B0  6E 61 65 63 68 73 74 65 6E 20 54 41 42 2D 53 74 naechsten TAB-St
01C0  6F 70 20 20 20 20 20 20 5E 4A 20 3A 20 4B 6F 6D op      ^J : Kom
01D0  6D 61 6E 64 6F 20 61 62 73 63 68 6C 69 65 73 73 mando abschliess
01E0  65 6E 20 3C 63 72 3E 20 20 20 20 20 20 20 20 20 en <cr>
01F0  5E 4D 20 3A 20 4B 6F 6D 6D 61 6E 64 6F 20 61 62 ^M : Kommando ab

ADR>  -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F 0123456789ABCDEF
0200  73 63 68 6C 69 65 73 73 65 6E 20 3C 63 72 3E 20 schliessen <cr>
0210  20 20 20 20 20 20 20 20 5E 50 20 3A 20 41 75 73         ^P : Aus
0220  67 61 62 65 20 75 6D 73 63 68 61 6C 74 65 6E 20 gabe umschalten
0230  28 44 72 75 63 6B 65 72 2F 56 69 64 65 6F 29 20 (Drucker/Video)
0240  5E 52 20 3A 20 4B 6F 6D 6D 61 6E 64 6F 7A 65 69 ^R : Kommandozei
0250  6C 65 20 6B 6F 6D 70 6C 65 74 74 20 7A 65 69 67 le komplett zeig
0260  65 6E 20 20 20 20 20 20 5E 53 20 3A 20 55 6E 74 en      ^S : Unt
0270  65 72 62 72 69 63 68 74 20 6C 61 75 66 65 6E 64 erbricht laufend

ADR>  -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F 0123456789ABCDEF
0280  65 20 41 6E 7A 65 69 67 65 20 20 20 20 20 20 20 e Anzeige
0290  5E 55 20 3A 20 42 72 69 63 68 74 20 4B 6F 6D 6D ^U : Bricht Komm
02A0  61 6E 64 6F 7A 65 69 6C 65 20 61 62 20 20 20 20 andozeile ab
02B0  20 20 20 20 20 20 20 20 5E 56 20 3A 20 42 72 69         ^V : Bri
02C0  63 68 74 20 4B 6F 6D 6D 61 6E 64 6F 7A 65 69 6C cht Kommandozeil
02D0  65 20 61 62 20 20 20 20 20 20 20 20 20 20 20 20 e ab
02E0  5E 58 20 3A 20 4C 6F 65 73 63 68 74 20 4B 6F 6D ^X : Loescht Kom
02F0  6D 61 6E 64 6F 7A 65 69 6C 65 20 20 20 20 20 20 mandozeile

ADR>  -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F 0123456789ABCDEF
0300  20 20 20 20 20 20 20 20 0A 0D 20 20 20 20 20 20          ..
0310  20 20 20 20 20 20 20 3C 64 3A 3E 20 20 3C 44 49        <d:>  <DI
0320  52 3E 20 20 45 52 41 3E 20 20 3C 52 45 4E 3E 20 R>  ERA>  <REN>
0330  20 3C 54 59 50 45 3E 20 20 3C 55 53 45 52 3E 20  <TYPE>  <USER>
0340  20 3C 53 41 56 45 3E 1A 01 50 0B 19 24 00 00 00  <SAVE>..P..$...
0350  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0360  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0370  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

End of dumping
```

7.2 HEXDUMP - verbessertes DUMP-Programm

Das zur Diskettenstation DDI-1 mitgelieferte CP/M-Dienstprogramm ‚DUMP' (siehe 3.16) weist einige nicht unerhebliche Nachteile auf:

1. Das als Hex-Dump aufzulistende Programm muß sich auf der Diskette befinden, auf der sich auch das DUMP-Programm befindet.
2. Für das schnelle Finden von Texten fehlt eine ASCII-Ausgabe.
3. Die Ausgabe ist auf Grund der fehlenden Byteangabe sehr unübersichtlich.

Das etwas umfangreichere Programm HEXDUMP arbeitet wie das DUMP-Programm, allerdings ohne die oben genannten Nachteile.

Es besteht aus einer Reihe von Unterprogrammen, die auch im Programm DISCMON genutzt werden (oder von Ihnen bei eigenen Projekten benutzt werden können). Dort finden Sie die Umrechnungen in hexadezimale Zahlen, die Ausgabe eines Zeichens, die Ausgabe einer Leerzeile, etc.. Ein größeres Unterprogramm ist die Routine zur Ausgabe der DMA. Sie besteht aus den Teilen DMAOUT, ASCII und ADROUT. Das DMAOUT-Unterprogramm hat die Aufgabe, die DMA (128 Bytes) auf dem Bildschirm auszugeben. Als erstes wird die Speicheradresse (ADROUT) ausgegeben, dann folgen 16 Bytes als hexadezimale Zahlen. Anschließend werden diese 16 Bytes als ASCII-DUMP (ASCII) ausgegeben. Dieser Vorgang wiederholt sich, bis die gesamte DMA ausgegeben ist (damit ist der gesamte Sektor gelistet).

Das Programm beginnt mit der Aufforderung, die Diskette, auf der sich das gewünschte Programm befindet, in das Laufwerk (A) zu legen und eine Taste zu betätigen. Danach findet ein Disketten-Reset statt. Nun erfolgt das Öffnen der gewünschten Datei. Ist sie nicht vorhanden, gibt das Programm eine Fehlermeldung aus und beendet seine Arbeit mit einem Sprung in ein Unterprogramm zum Programmende (ENDE). Andernfalls wird ein Record in die DMA eingelesen (MAINPRG). Ist der Record erfolgreich in die DMA eingelesen (kein Dateiende erreicht), erfolgt die Ausgabe der DMA (DMA-OUT). Dieser Vorgang wiederholt sich, bis das Dateiende erreicht ist. Der Programmteil zum Beenden des Programms (ENDE) gibt eine Meldung aus, schließt die geöffnete Datei (in diesem Fall nicht notwendig), fordert den Benutzer auf, eine Diskette mit CP/M-Systemspuren in das Laufwerk zu legen und eine Taste zu betätigen. Abschließend erfolgt ein CP/M-Warmstart (JUMP 0000H).

Der Aufruf des Programms erfolgt mit dem Kommando

```
A>HEXDUMP filename.typ
```

```
                ;*************************************************
                ;*                                               *
                ;*              H E X D U M P   VER 1.0          *
                ;*                                               *
                ;*              A>HEXDUMP filename.typ           *
                ;*                                               *
                ;*************************************************
0000 =          BOOT    EQU     0000H
0005 =          BDOS    EQU     0005H
0001 =          CONIN   EQU       01H
0002 =          CONOUT  EQU       02H
0009 =          STROUT  EQU       09H
000D =          RESET   EQU       0DH
000F =          OPEN    EQU       0FH
0010 =          CLOSE   EQU       10H
0014 =          SEQREAD EQU       14H
005C =          FCB     EQU     005CH
0080 =          DMA     EQU     0080H
                ;
0100                    ORG     0100H
0100 111902             LXI D,STRING1 ;Titelmeldung ausgeben
0103 0E09               MVI  C,STROUT
0105 CD0500             CALL     BDOS
0108 0E01               MVI   C,CONIN ;Wartet auf Zeichen v. Tastatur
010A CD0500             CALL     BDOS
010D 0E0D               MVI   C,RESET ;Diskettenreset ausfuehren
010F CD0500             CALL     BDOS
                ;
0112 CDFF01             CALL     CRLF ;Zeilenvorschub
0115 115C00             LXI      D,FCB ;<DE>=FCB-Adresse
0118 0E0F               MVI    C,OPEN ;Suche File auf der Disk
011A CD0500             CALL     BDOS
011D FEFF               CPI      0FFH ;<A>=0FFH File nicht vorhanden
011F C22D01             JNZ        OK
                ;
0122 116E02     ERROR   LXI D,STRING2 ;Fehlermeldung ausgeben
0125 0E09               MVI  C,STROUT
0127 CD0500             CALL     BDOS
012A C34901             JMP      ENDE
                ;
012D 21F000     OK      LXI   H,00F0H ;Speicheradresse vorbereiten
0130 221702             SHLD  ADRESSE ;und merken
                ;
0133 CDFF01     MAINPRG CALL     CRLF ;Zeilenvorschub
0136 115C00             LXI      D,FCB ;<DE>=FCB-Adresse
0139 0E14               MVI C,SEQREAD ;Record einlesen
013B CD0500             CALL     BDOS
013E FE00               CPI       00H ;Wenn <A> <> 0 dann
0140 C24901             JNZ      ENDE ;folgt kein weiterer Record
```

```
0143 CD6901             CALL   DMAOUT ;Ausgabe des DMA-Inhaltes
0146 C33301             JMP    MAINPRG
                ;
0149 11D002     ENDE    LXI D,STRING4 ;Meldung ausgeben
014C 0E09               MVI  C,STROUT
014E CD0500             CALL     BDOS
0151 115C00             LXI      D,FCB ;Datei wieder schliessen
0154 0E10               MVI    C,CLOSE ;beim Lesen nicht notwendig
0156 CD0500             CALL     BDOS
0159 11E202             LXI D,STRING5 ;Meldung ausgeben
015C 0E09               MVI  C,STROUT
015E CD0500             CALL     BDOS
0161 0E01               MVI    C,CONIN ;Wartet auf Zeichen v. Tastatur
0163 CD0500             CALL     BDOS
0166 C30000             JMP      BOOT ;Programmende
                ;
                ;*********************************************************
                ;****                 DMA-Inhalt ausgeben             ****
                ;*********************************************************
0169 118702     DMAOUT  LXI D,STRING3 ;Kopfzeile ausgeben
016C 0E09               MVI  C,STROUT
016E CD0500             CALL     BDOS
0171 0680               MVI      B,80H ;<B>=80H Zaehler auf 128
0173 218000             LXI      H,DMA ;<HL>=DMA-Adresse
0176 CDBC01     DMAOUT1 CALL    ADROUT ;Adresse ausgeben
0179 1610               MVI      D,10H ;<D>=10H Zaehler auf 16
017B 7E         DMAOUT2 MOV        A,M ;<A> mit adressierten Byte laden
017C CDD901             CALL    HEXOUT ;und als Hexzahl ausgeben
017F CDF901             CALL     SPACE ;Leerzeichen ausgeben
0182 23                 INX          H ;<HL>=<HL>+1
0183 05                 DCR          B ;<B>=<B>-1
0184 C28B01             JNZ   DMAOUT3 ;Kein Recordende erreicht
0187 CD9501             CALL     ASCII ;16 Zeichen (ASCII) ausgeben
018A C9                 RET            ;Recordende erreicht ENDE
018B 15         DMAOUT3 DCR          D ;<D>=<D>-1
018C C27B01             JNZ   DMAOUT2 ;Kein Zeilenende ereicht
018F CD9501             CALL     ASCII ;Zeilenende  ASCII-DUMP
0192 C37601             JMP   DMAOUT1 ;Naechste Zeile
                ;
                ;*********************************************************
                ;****          Ausgabe von 16 ASCII-Zeichen           ****
                ;*********************************************************
0195 C5         ASCII   PUSH         B ;<BC> merken
0196 0610               MVI      B,10H ;<B>=10H Zaehler auf 16
0198 2B         ASCII1  DCX          H ;<HL>=<HL>-1
0199 05                 DCR          B ;<B>=<B>-1
019A C29801             JNZ    ASCII1 ;und das 16 mal
019D 0610               MVI      B,10H ;<B>=10H Zaehler auf 16
019F 7E         ASCII2  MOV        A,M ;<A>=adressiertes Byte
01A0 FE20               CPI        20H ;Vergleiche <A> mit 32 (<Space)
01A2 DAB701             JC     ASCII4 ;Kleiner, dann PUNKT
01A5 FE80               CPI        80H ;Vergleiche <A> mit 128
01A7 D2B701             JNC    ASCII4 ;Groesser, dann PUNKT
01AA CD0A02     ASCII3  CALL  AUSGABE ;Zeichen ausgeben
01AD 23                 INX          H ;<HL>=<HL>+1
```

```
01AE 05                 DCR         B ;<B>=<B>-1
01AF C29F01             JNZ     ASCII2 ;<B>=0 Nein, dann weiter
01B2 CDFF01             CALL      CRLF ;Ende. Zeilenvorschub
01B5 C1                 POP         B ;<BC> wieder holen
01B6 C9                 RET           ;Unterprogrammende
01B7 3E2E       ASCII4  MVI       A,'.' ;<A>=2EH
01B9 C3AA01             JMP     ASCII3
                ;
                ;*************************************************************
                ;*****          Speicheradresse ausgeben              *****
                ;*************************************************************
01BC E5         ADROUT  PUSH        H ;<HL> merker
01BD C5                 PUSH        B ;<BC> merken
01BE 2A1702             LHLD  ADRESSE ;<HL>=Adresse
01C1 011000             LXI   B,0010H ;<BC>=0010H
01C4 09                 DAD         B ;<HL>=<HL>+<BC>
01C5 221702             SHLD  ADRESSE ;und <HL> wieder speichern
01C8 7C                 MOV       A,H ;<A>=<H> Highbyte
01C9 CDD901             CALL   HEXOUT ;nach HEX wandeln und ausgeben
01CC 7D                 MOV       A,L ;<A>=<L> Lowbyte
01CD CDD901             CALL   HEXOUT ;nach HEX wandeln und ausgeben
01D0 CDF901             CALL    SPACE ;Leerzeichen ausgeben
01D3 CDF901             CALL    SPACE ;Leerzeichen ausgeben
01D6 C1                 POP         B ;<BC> wieder holen
01D7 E1                 POP         H ;<HL> wieder holen
01D8 C9                 RET           ;Unterprogrammende
                ;
                ;*************************************************************
                ;****      Umrechnung/Ausgabe <A> nach HEX            ******
                ;*************************************************************
01D9 F5         HEXOUT  PUSH      PSW ;<A> merken
01DA 0F                 RRC           ;<A> 4 mal rechts schieben
01DB 0F                 RRC
01DC 0F                 RRC
01DD 0F                 RRC
01DE CDE601             CALL  HEXOUT1 ;Nibble ausgeben
01E1 F1                 POP       PSW ;<A> wieder holen
01E2 CDE601             CALL  HEXOUT1 ;Nibble ausgeben
01E5 C9                 RET           ;Unterprogrammende
01E6 E60F       HEXOUT1 ANI       0FH ;Bit 0-3 isolieren
01E8 FE0A               CPI       0AH ;Vergleiche <A> mit 10
01EA D2F301             JNC   HEXOUT2 ;<A> groesser 10
01ED C630               ADI       30H ;<A>=<A>+30H - nach ASCII formen
01EF CD0A02             CALL  AUSGABE ;und ausgeben
01F2 C9                 RET
01F3 C637       HEXOUT2 ADI       37H ;<A>=<A>+37H - nach ASCII formen
01F5 CD0A02             CALL  AUSGABE ;und ausgeben
01F8 C9                 RET
                ;
                ;*************************************************************
                ;****            Leerzeichen ausgeben                 ******
                ;*************************************************************
01F9 3E20       SPACE   MVI       A,' ' ;<A>=20H
01FB CD0A02             CALL  AUSGABE
01FE C9                 RET
```

```
                        ;
                        ;*********************************************************
                        ;****              Zeilenvorschub                     ****
                        ;*********************************************************
01FF 3E0A       CRLF    MVI     A,0AH ;Line Feeed
0201 CD0A02             CALL  AUSGABE ;ausgeben
0204 3E0D               MVI     A,0DH ;Return
0206 CD0A02             CALL  AUSGABE ;ausgeben
0209 C9                 RET
                        ;
                        ;*********************************************************
                        ;****           Zeichen <A> ausgeben                  ****
                        ;*********************************************************
020A C5         AUSGABE PUSH        B ;Register merken, da nach
020B D5                 PUSH        D ;Einsprung ins BDOS alle
020C E5                 PUSH        H ;undeferent.
020D 5F                 MOV       E,A ;<E>=<A> Zeichen ins <E> bringen
020E 0E02               MVI  C,CONOUT ; und ausgeben
0210 CD0500             CALL     BDOS
0213 E1                 POP         H ;Alle Register zurueck holen
0214 D1                 POP         D
0215 C1                 POP         B
0216 C9                 RET
                        ;
                        ;*********************************************************
                        ;****    Speicherbereich fuer Strings & Adressen    ****
                        ;*********************************************************
0217 0000       ADRESSE DW 0000H
0219 04021C0000 STRING1 DB 04H,02H,1CH,00H,00H,00H,1CH,01H,1AH,1AH
0223 1D00004845         DB 1DH,00H,00H,'HEXDUMP VER. 1.0'
0236 0A0D0A             DB 0AH,0DH,0AH
0239 506C656173         DB 'Please insert disc into drive A '
0259 7468656E20         DB 'then press any key:_$'
026E 4552524F52 STRING2 DB 'ERROR: File not found ',0AH,0DH,'$'
0287 4144523E20 STRING3 DB 'ADR>  -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 '
02AB 2D41202D42         DB '-A -B -C -D -E -F 0123456789ABCDEF'
02CD 0A0D24             DB 0AH,0DH,'$'
02D0 456E64206F STRING4 DB 'End of dumping',0AH,0DH,0AH,'$'
02E2 506C656173 STRING5 DB 'Please insert CP/M-Systemdisc into '
0305 6472697665         DB 'drive A then press any key:_$'
                        ;
0322            END
```

```
ADR>  -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F 0123456789ABCDEF
0100  11 19 02 0E 09 CD 05 00 0E 01 CD 05 00 0E 0D CD ................
0110  05 00 CD FF 01 11 5C 00 0E 0F CD 05 00 FE FF C2 ......\.........
0120  2D 01 11 6E 02 0E 09 CD 05 00 C3 49 01 21 F0 00 -..n.......I.!..
0130  22 17 02 CD FF 01 11 5C 00 0E 14 CD 05 00 FE 00 ".......\........
0140  C2 49 01 CD 69 01 C3 33 01 11 D0 02 0E 09 CD 05 .I..i..3........
0150  00 11 5C 00 0E 10 CD 05 00 11 E2 02 0E 09 CD 05 ..\.............
0160  00 0E 01 CD 05 00 C3 00 00 11 87 02 0E 09 CD 05 ................
0170  00 06 80 21 80 00 CD BC 01 16 10 7E CD D9 01 CD ...!.......~....

ADR>  -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F 0123456789ABCDEF
0180  F9 01 23 05 C2 8B 01 CD 95 01 C9 15 C2 7B 01 CD ..#..........{..
0190  95 01 C3 76 01 C5 06 10 2B 05 C2 98 01 06 10 7E ...v....+......~
01A0  FE 20 DA B7 01 FE 80 D2 B7 01 CD 0A 02 23 05 C2 . ...........#..
01B0  9F 01 CD FF 01 C1 C9 3E 2E C3 AA 01 E5 C5 2A 17 .......>......*.
01C0  02 01 10 00 09 22 17 02 7C CD D9 01 7D CD D9 01 .....".."|...}...
01D0  CD F9 01 CD F9 01 C1 E1 C9 F5 0F 0F 0F 0F CD E6 ................
01E0  01 F1 CD E6 01 C9 E6 0F FE 0A D2 F3 01 C6 30 CD ..............0.
01F0  0A 02 C9 C6 37 CD 0A 02 C9 3E 20 CD 0A 02 C9 3E ....7....> ....>

ADR>  -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F 0123456789ABCDEF
0200  0A CD 0A 02 3E 0D CD 0A 02 C9 C5 D5 E5 5F 0E 02 ....>........_..
0210  CD 05 00 E1 D1 C1 C9 00 00 04 02 1C 00 00 00 1C ................
0220  01 1A 1A 1D 00 00 48 45 58 44 55 4D 50 20 56 45 ......HEXDUMP VE
0230  52 2E 20 31 2E 30 0A 0D 0A 50 6C 65 61 73 65 20 R. 1.0...Please
0240  69 6E 73 65 72 74 20 64 69 73 63 20 69 6E 74 6F insert disc into
0250  20 64 72 69 76 65 20 41 20 74 68 65 6E 20 70 72  drive A then pr
0260  65 73 73 20 61 6E 79 20 6B 65 79 3A 5F 24 45 52 ess any key:_$ER
0270  52 4F 52 3A 20 46 69 6C 65 20 6E 6F 74 20 66 6F ROR: File not fo

ADR>  -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F 0123456789ABCDEF
0280  75 6E 64 20 0A 0D 24 41 44 52 3E 20 20 2D 30 20 und ..$ADR>  -0
0290  2D 31 20 2D 32 20 2D 33 20 2D 34 20 2D 35 20 2D -1 -2 -3 -4 -5 -
02A0  36 20 2D 37 20 2D 38 20 2D 39 20 2D 41 20 2D 42 6 -7 -8 -9 -A -B
02B0  20 2D 43 20 2D 44 20 2D 45 20 2D 46 20 30 31 32  -C -D -E -F 012
02C0  33 34 35 36 37 38 39 41 42 43 44 45 46 0A 0D 24 3456789ABCDEF..$
02D0  45 6E 64 20 6F 66 20 64 75 6D 70 69 6E 67 0A 0D End of dumping..
02E0  0A 24 50 6C 65 61 73 65 20 69 6E 73 65 72 74 20 .$Please insert
02F0  43 50 2F 4D 2D 53 79 73 74 65 6D 64 69 73 63 20 CP/M-Systemdisc

ADR>  -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F 0123456789ABCDEF
0300  69 6E 74 6F 20 64 72 69 76 65 20 41 20 74 68 65 into drive A the
0310  6E 20 70 72 65 73 73 20 61 6E 79 20 6B 65 79 3A n press any key:
0320  5F 24 00 00 00 00 00 00 00 00 00 00 00 00 00 00 _$..............
0330  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0340  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0350  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0360  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0370  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
```

7.3 XDIR - die andere Directory

Für jeden Eintrag in der Directory steht ein Bereich von 32 Bytes zur Verfügung. Der Befehl ‚DIR' gibt nur den Filenamen, Typ und die Länge des Files wieder, obwohl weitere interessante Informationen in diesem Eintrag stehen. Das nachfolgende Programm ‚XDIR' gibt alle Informationen des Directory-Eintrags aus und ermöglicht somit ein besseres Verständnis des Diskettenaufbaus.

Im Directory-Eintrag stehen neben dem Namen und dem Filetyp vor allem die Informationen, wo sich der File auf der Diskette befindet. Die letzten 16 Bytes des Eintrags geben die ‚Blocknummern' wieder, in denen das entsprechende File liegt. Jeder Block kann 1024 Bytes (1 kByte) Informationen aufnehmen. Das entspricht 8 CP/M-Sektoren a 128 Bytes (8*128=1024). Auf jeder der 40 Spuren (0-39) befinden sich also 36 CP/M-Sektoren (Records). Da die Spuren 0 und 1 vom Betriebssystem belegt sind, beginnt der eigentliche freie Speicherbereich erst ab Spur 2. Für die Directory sind weiterhin 16 Sektoren a 128 Bytes reserviert, so daß sie maximal 64 Einträge aufnehmen kann. Diese 16 Sektoren befinden sich in der Spur 2/Sektor 0 bis Spur 2/Sektor 15 (16*128 = 2048= 2 Blöcke = Block 0 und 1) und damit beginnt der erste ‚freie' Sektor in Spur 2/Sektor 16.

Die letzten 16 Bytes eines Directory-Eintrags könnten folgendermaßen aussehen (hexadezimale Werte)

02 03 04 05 00 00 00 00 00 00 00 00 00 00 00 00

Das entsprechende Programm liegt auf der Diskette in den Blöcken 2,3,4 und 5 (Eine ‚0' bedeutet nicht benutzt). Da jeder Block 1 kByte (8 CP/M-Sektoren) groß ist, beträgt der Umfang des Programms rund 4 kByte. Anhand der Blocknummer läßt sich errechnen, in welcher Spur und welchem Sektor das Programm beginnt. Blocknummer 02 bedeutet, daß der Block im Sektor 16 beginnt (Block * 8 = Sektor). Da die Spuren 0 und 1 vom CP/M-Betriebssystem belegt sind, entspricht dieses Spur 2/Sektor 16. Der Block 03 beginnt dementsprechend bei Spur 2/Sektor 24 (3*8 = 24).

Block 0 = 0*8 = Sektor 0- 7 / Spur 2 (Directory)
Block 1 = 1*8 = Sektor 8-15 / Spur 2 (Directory)
Block 2 = 2*8 = Sektor 16-23 / Spur 2
Block 3 = 3*8 = Sektor 24-31 / Spur 2
Block 4 = 4*8 = Sektor 32-35 / Spur 2
(Übertrag auf nächste Spur)
Sektor 0- 3 / Spur 3 (!!!)

Der gesamte Block 4 paßt nicht mehr komplett auf die Spur 2 (36 Sektoren). Die nächsten Sektoren befinden sich auf der nächsten Spur. Mit der folgenden Gleichung kann für eine beliebige Blocknummer die Startposition (Spur/Sektor) des Blocks auf der Diskette errechnet werden

```
  help = INT(block*8/36)
  spur = help+2
sektor = (block*8)-(help*36)
```

Da ein Directory-Eintrag nur eine Speichergröße von 16 kByte adressieren kann (16 Blöcke * 1 kByte), tritt ein Problem auf, wenn das Programm mehr Speicherplatz benötigt. Dieses Problem wird umgangen, indem einfach weitere Directory-Einträge unter demselben Namen angelegt werden. Ein Programm mit bis zu 32 kByte benötigt 2 Einträge, mit bis zu 48 kByte 3 Einträge, etc..

Die erweiterten Einträge werden als sogenannte ‚Extension' geführt und beim DIR-Befehl nicht mit angezeigt. Das Byte 12 des Eintrags bestimmt, um welche Extension es sich handelt (0 = 1.Eintrag, 1 = 2.Eintrag etc.).

Alle Bytes eines Directory-Eintrags auf einen Blick

Byte	0	Benutzerebene, unter der die Datei gesichert wurde (USER 0 bis 15). Befindet sich hier der Wert 0E5H, so ist die Datei gelöscht!
Byte	1- 8	Filename in Großbuchstaben.
Byte	9-11	Filetyp in Großbuchstaben. Das hochwertige Bit (Bit 7) der Bytes 9 und 10 kennzeichnet weiterhin die Dateiattribute (siehe auch BDOS-Funktion 30): Bit 7 Byte 9 = 0: R/W = 1: R/O (Schreibschutz) Bit 7 Byte 10 = 0: Normaler Eintrag = 1: SYS-Datei, wird bei DIR nicht gelistet
Byte	12	Extension
Byte	13	Unbenutzt
Byte	14-15	16-Bit-Wert gibt die Anzahl der tatsächlich benutzten Sektoren an (1-128).
Byte	16-31	Blocknummern

Das Programm ‚XDIR' zeigt alle oben aufgeführten Informationen der Directory an. Der Aufruf des Programms erfolgt mit Angaben zum Filenamen und Filetypen. Hierbei sind Wildcards zulässig.

A>XDIR *.*	Listet alle Einträge
A>XDIR *.COM	Listet alle COM-Datei-Einträge
A>XDIR TEST.BAS	Listet nur den Eintrag für TEST.BAS

(Anmerkung: Eine sehr umfangreiche und detaillierte Darstellung des Diskettenaufbaus fand bereits in Band 2 dieser Reihe statt. Bitte sehen Sie ggf. dort noch einmal nach.)

```
                ;*********************************************************
                ;*                                                       *
                ;*                      X D I R                          *
                ;*                A>XDIR filename.typ                    *
                ;*                                                       *
                ;*********************************************************
                ;
0000 =          BOOT      EQU     0000H
0005 =          BDOS      EQU     0005H
0001 =          CONIN     EQU       01H
0002 =          CONOUT    EQU       02H
0009 =          STROUT    EQU       09H
0011 =          FIRSTD    EQU       11H
0012 =          NEXTDIR   EQU       12H
005C =          FCB       EQU     005CH
0080 =          DMA       EQU     0080H
                ;
0100                      ORG     0100H
0100 0E11                 MVI   C,FIRSTD ;Suche 1. DIR-Eintrag der
0102 115C00               LXI     D,FCB ;im FCB gekennzeichneten File(s)
0105 CD0500               CALL     BDOS
0108 FEFF                 CPI       0FFH ;Wenn <A>=FFH, dann sind keine
010A CA8301               JZ        ENDE ;Daten mehr vorhanden - ENDE

010D 11CC01               LXI D,STRING1 ;Meldung ausgeben
0110 0E09                 MVI   C,STROUT
0112 CD0500               CALL     BDOS
                ;
0115 218000     START     LXI     H,DMA ;Adresse der DMA in <HL> laden
0118 112000               LXI     D,20H
011B B7         PRUEF     ORA        A ;<A>=0,1,2,3 Position des
011C CA2401               JZ      USER ;Fileeintrags in der DMA (<A>*32)
011F 19                   DAD        D ;<HL>=<HL>+<DE>
0120 3D                   DCR        A ;<A>=<A>-1
0121 C31B01               JMP     PRUEF
                ;
0124 CDBB01     USER      CALL    SPACE ;Leerzeichen ausgeben
0127 7E                   MOV       A,M ;1.Byte = Usernummer
0128 CD8E01               CALL   HEXOUT ;ausgeben
```

```
012B CDBB01             CALL    SPACE ;Leerzeichen ausgeben
012E CDBB01             CALL    SPACE
                ;
0131 0E08       NAME    MVI     C,08H ;Naechste 8 Zeichen = NAME
0133 23         NAME1   INX       H ;<HL>=<HL>+1
0134 7E                 MOV       A,M ;Zeichen holen
0135 CDAE01             CALL  AUSGABE ;und ausgeben
0138 0D                 DCR         C ;<C>=<C>-1
0139 C23301             JNZ     NAME1 ;8 Zeichen gelesen ? Nein, springe
                ;
013C 3E2E       PUNKT   MVI     A,'.' ;Punkt
013E CDAE01             CALL  AUSGABE ;ausgeben
                ;
0141 0E03       TYP     MVI     C,03H ;Naechste 3 Zeichen = TYP
0143 23         TYP1    INX       H ;<HL>=<HL>+1
0144 7E                 MOV       A,M ; Zeichen holen und
0145 CDAE01             CALL  AUSGABE ;ausgeben
0148 0D                 DCR         C ;<C>=<C>-1
0149 C24301             JNZ      TYP1 ;3 Zeichen gelesen ? Nein, springe
014C CDBB01             CALL    SPACE
014F 23         EXTEN   INX       H ;Naechstes Byte gibt die Groesse
0150 7E                 MOV       A,M ;in 16k Bloecken an
0151 CD8E01             CALL   HEXOUT
0154 CDBB01             CALL    SPACE
                ;
0157 23                 INX       H ;Naechstes Byte ueberlesen
0158 23                 INX       H
                ;
0159 7E         SEC     MOV       A,M ;Naechste 2 Bytes geben die
015A CD8E01             CALL   HEXOUT ;Satzanzahl an (16 Bit-Wert)
015D 23                 INX       H
015E 7E                 MOV       A,M
015F CD8E01             CALL   HEXOUT
0162 CDBB01             CALL    SPACE
                ;
0165 0E10       BLOCK   MVI     C,10H ;Naechste 16 Bytes zeigen die
0167 23         BLOCK1  INX       H ;Blockbelegung des Files  an
0168 7E                 MOV       A,M
0169 CD8E01             CALL   HEXOUT
016C CDBB01             CALL    SPACE
016F 0D                 DCR         C
0170 C26701             JNZ    BLOCK1
0173 CDC101             CALL     CRLF
                ;
                ;
0176 115C00     WEITER  LXI     D,FCB ;Naechsten DIR-Block holen
0179 0E12               MVI C,NEXTDIR
017B CD0500             CALL     BDOS
017E FEFF               CPI      0FFH ;Wenn <A>=FFH dann folgt
0180 C21501             JNZ     START ;kein weiterer DIR-Block
                ;
0183 111802     ENDE    LXI D,STRING2 ;Meldung ausgeben
0186 0E09               MVI  C,STROUT
0188 CD0500             CALL     BDOS
018B C30000             JMP      0000H ; Warmstart
```

```
                        ;
                        ;*********************************************************
                        ;****       Umrechnung/Ausgabe <A> nach HEX          ****
                        ;*********************************************************
018E F5         HEXOUT  PUSH        A ;<A> merken
018F 0F                 RRC           ;<A> 4 mal rechts schieben
0190 0F                 RRC
0191 0F                 RRC
0192 0F                 RRC
0193 CD9B01             CALL   NIBBLE ;Nibble ausgeben
0196 F1                 POP         A ;<A> holen
0197 CD9B01             CALL   NIBBLE ;Nibble ausgeben
019A C9                 RET           ;HEXOUT-Ende !!!!!!
019B E60F       NIBBLE  ANI       0FH ;Bit 0-3 isolieren
019D FE0A               CPI       0AH ;vergleiche <A> mit 0AH
019F D2A801             JNC  GROESSER ;<A> groesser 10 ... springe
01A2 C630       KLEINER ADI       30H ;<A>=<A>+30H - nach ASCII formen
01A4 CDAE01             CALL  AUSGABE ;und ausgeben
01A7 C9                 RET
01A8 C637       GROESSER ADI      37H ;<A>=<A>+37H - nach ASCII formen
01AA CDAE01             CALL  AUSGABE ;und ausgeben
01AD C9                 RET
                        ;
                        ;*********************************************************
                        ;****            Zeichen <A> ausgeben                 ****
                        ;*********************************************************

01AE C5         AUSGABE PUSH        B ;Register merken, da nach
01AF D5                 PUSH        D ;Einsprung ins BDOS alle
01B0 E5                 PUSH        H ;undeferent !!!!
01B1 5F                 MOV       E,A ;<E>=<A> Zeichen ins <E>
01B2 0E02               MVI  C,CONOUT ;und ausgeben
01B4 CD0500             CALL     BDOS ;Zeichen in <E> ausgeben
01B7 E1                 POP         H ;Alle Register wieder holen
01B8 D1                 POP         D
01B9 C1                 POP         B
01BA C9                 RET
                        ;
                        ;*********************************************************
                        ;****           Leerzeichen ausgeben                  ****
                        ;*********************************************************
01BB 3E20       SPACE   MVI     A,' ' ;<A>=20H laden
01BD CDAE01             CALL  AUSGABE ;und ausgeben
01C0 C9                 RET
                        ;
                        ;*********************************************************
                        ;****                Zeilenvorschub                   ****
                        ;*********************************************************
01C1 3E0A       CRLF    MVI     A,0AH ;Line feed
01C3 CDAE01             CALL  AUSGABE ;ausgeben
01C6 3E0D               MVI     A,0DH ;Return
01C8 CDAE01             CALL  AUSGABE ;ausgeben
01CB C9                 RET
```

```
                ;
                ;*********************************************************
                ;****   Speicherbereich fuer Strings & Adressen      ******
                ;*********************************************************
01CC 5553455220 STRING1  DB 'USER NAME    .TYP EX SEC. ---------------'
01F5 2D2D2D2D2D          DB '-----Bloecke--------------------'
0215 0A0D24              DB 0AH,0DH,'$'
0218 4E4F20284D STRING2  DB 'NO (MORE) FILE(S)$'
022A            END
```

```
ADR>  -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F 0123456789ABCDEF
0100  0E 11 11 5C 00 CD 05 00 FE FF CA 83 01 11 CC 01 ...\............
0110  0E 09 CD 05 00 21 80 00 11 20 00 B7 CA 24 01 19 .....!... ...$..
0120  3D C3 1B 01 CD BB 01 7E CD 8E 01 CD BB 01 CD BB =......~........
0130  01 0E 08 23 7E CD AE 01 0D C2 33 01 3E 2E CD AE ...#~.....3.>...
0140  01 0E 03 23 7E CD AE 01 0D C2 43 01 CD BB 01 23 ...#~.....C....#
0150  7E CD 8E 01 CD BB 01 23 23 7E CD 8E 01 23 7E CD ~......##~...#~.
0160  8E 01 CD BB 01 0E 10 23 7E CD 8E 01 CD BB 01 0D .......#~.......
0170  C2 67 01 CD C1 01 11 5C 00 0E 12 CD 05 00 FE FF .g.....\........

ADR>  -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F 0123456789ABCDEF
0180  C2 15 01 11 18 02 0E 09 CD 05 00 C3 00 00 F5 0F ................
0190  0F 0F 0F CD 9B 01 F1 CD 9B 01 C9 E6 0F FE 0A D2 ................
01A0  A8 01 C6 30 CD AE 01 C9 C6 37 CD AE 01 C9 C5 D5 ...0.....7......
01B0  E5 5F 0E 02 CD 05 00 E1 D1 C1 C9 3E 20 CD AE 01 ._.........> ...
01C0  C9 3E 0A CD AE 01 3E 0D CD AE 01 C9 55 53 45 52 .>....>.....USER
01D0  20 4E 41 4D 45 20 20 20 20 2E 54 59 50 20 45 58  NAME    .TYP EX
01E0  20 53 45 43 2E 20 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D  SEC. ----------
01F0  2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 42 6C 6F 65 63 6B ----------Bloeck

ADR>  -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F 0123456789ABCDEF
0200  65 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D e---------------
0210  2D 2D 2D 2D 2D 0A 0D 24 4E 4F 20 28 4D 4F 52 45 -----..$NO (MORE
0220  29 20 46 49 4C 45 28 53 29 24 00 00 00 00 00 00 ) FILE(S)$......
0230  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0240  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0250  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0260  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0270  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

End of dumping
```

7.4 DISCMON - Zugriff auf jeden Sektor

Ein Diskettenmonitor ermöglicht den Zugriff auf jeden Sektor der Diskette. Er kann in den Rechner (DMA) eingeladen und dann betrachtet, geändert und zurückgeschrieben werden. Durch den direkten Einfluß auf den Inhalt eines beliebigen Sektors ist es u.a. möglich, eine gelöschte Datei zu ‚retten', den Filenamen zu ändern oder Programme auf eine andere Benutzerebene zu legen. Besonders wertvoll ist ein Diskettenmonitor auch zur Analyse des Diskettenaufbaus.

Der direkte Zugriff auf Sektoren, die nicht einer Datei zugeordnet sind, ist mit den BDOS-Routinen nicht möglich. Wie schon in Kapitel 4.3 erläutert, kann aber über bestimmte Einsprünge auch direkt auf Routinen des BIOS zugegriffen werden. Besonders interessant sind die Routinen

CALL C524H	Spur auswählen (SETTRACK)
CALL C529H	Sektor auswählen (SETSEC)
CALL C2F7H	Sektor in DMA lesen (READ)
CALL C2FCH	DMA in Sektor schreiben (WRITE)

Mit den Routinen SETSEC und SETTRACK ist der Zugriff auf einen bestimmten Sektor vorzubereiten (das C-Register enthält jeweils die Spur oder den Sektor). Nach dem Setzen der Spur und dem Sektor kann mit den Routinen READ oder WRITE der entsprechende Sektor gelesen oder beschrieben werden.

Der DISCMON bietet nach dem Laden 6 Befehle (alle in Großbuchstaben eingeben !!)

X	Programmende
?	Zeigt alle Befehle an (Hilfe)
M	Zeigt die aktuelle DMA (Sektorinhalt) an
R&&&&	Liest einen bestimmten Sektor in die DMA und zeigt ihn gleichzeitig an. Nach dem Befehl ‚R' folgen zwei 2-stellige Hex-Zahlen (Spur/Sektor). Bei dem Versuch einen unzulässigen Sektor einzulesen (oder keine Hex-Zahl), erfolgt die Fehlermeldung ‚?'.
W&&&&	Schreibt den Inhalt der DMA auf den angegebenen Sektor (Spur/Sektor in Hex) zurück.
A&&&&	Ermöglicht das Ändern der aktuellen Inhalts des DMA. Die beiden 2-stelligen hexadezimalen Zahlen geben den

zu ändernden Bereich ‚von/bis' an. Nacheinander erscheinen die alten Inhalte, die entweder mit der ENTER-Taste bestätigt oder aber durch Eingabe einer 2-stelligen Hex-Zahl geändert werden können. Mit der Eingabe einer unzulässigen Hex-Zahl (z.B. dem Dezimalpunkt ‚.') ist ein vorzeitiges Eingabeende möglich.

Der DISCMON benutzt eine Reihe von Unterprogrammen, die auch im Programm HEXDUMP vorhanden sind. Damit Sie diese Teile nicht doppelt eingeben müssen, benutzen Sie einfach das Quellprogramm vom HEXDUMP und löschen die überflüssigen Zeilen.

Seien Sie bitte beim Arbeiten mit dem DISCMON besonders vorsichtig. Ein falsches Zurückschreiben (W) kann gerade im Directorybereich katastrophale Folgen haben!!

```
                ;*************************************************
                ;*                                               *
                ;*              DISK-MONITOR  VER 1.0            *
                ;*                                               *
                ;*                    A>DISKMON                  *
                ;*                                               *
                ;*************************************************
0000 =          BOOT    EQU     0000H
0005 =          BDOS    EQU     0005H
C524 =          SETTRACK EQU    0C524H
C529 =          SETSEC  EQU     0C529H
C2F7 =          READ    EQU     0C2F7H
C2FC =          WRITE   EQU     0C2FCH
0001 =          CONIN   EQU       01H
0002 =          CONOUT  EQU       02H
0009 =          STROUT  EQU       09H
000A =          BUFIN   EQU       0AH
001A =          SETDMA  EQU       1AH
                ;
0100            ORG     0100H
0100 119C03             LXI D,STRING1 ;Meldung ausgeben
0103 0E09               MVI  C,STROUT
0105 CD0500             CALL     BDOS
0108 111503             LXI     D,DMA ;DMA-Adresse neu setzen
010B 0E1A               MVI  C,SETDMA
010D CD0500             CALL     BDOS
                ;
0110 CDFB02     INPUT   CALL     CRLF ;Zeilenvorschub
0113 3E3A               MVI     A,':' ;Prompt-Zeichen ausgeben
0115 CD0603             CALL  AUSGABE
0118 119503             LXI  D,PUFFER ;Puffereingabe
011B 0E0A               MVI   C,BUFIN
011D CD0500             CALL     BDOS ;Kehrt erst nach <CR> zurueck
```

```
0120 219603             LXI H,POINTER ;Pruefe Laenge der Eingabe
0123 7E                 MOV       A,M ;<A>=adressiertes Byte
0124 FE00               CPI       00H ;Kein Zeichen vorhanden?
0126 CA1001             JZ      INPUT ;dann neue Eingabe
                ;
0129 23         CHECK   INX         H ;Zeigt auf 1.Zeichen
012A 7E                 MOV       A,M ;in <A> laden
012B FE58               CPI       'X' ;Pruefe auf X
012D CA5F01             JZ      ENDE
0130 FE4D               CPI       'M' ;Pruefe auf M
0132 CA7001             JZ      MARK
0135 FE3F               CPI       '?' ;Pruefe auf ?
0137 CA6201             JZ      HELP
013A 219603             LXI H,POINTER ;Pruefe ob Kommando aus
013D 7E                 MOV       A,M ;5 Zeichen besteht.
013E FE05               CPI       05H
0140 C25401             JNZ     ERROR ;Nein, dann Fehlermeldung
0143 23                 INX         H ;<HL>=<HL>+1
0144 7E                 MOV       A,M
0145 FE52               CPI       'R' ;Pruefe auf R
0147 CA7F01             JZ      LESE
014A FE57               CPI       'W' ;Pruefe auf W
014C CAA301             JZ    SCHREIB
014F FE41               CPI       'A' ;Pruefe auf A
0151 CAC701             JZ      ALTER
0154 CDFB02     ERROR   CALL     CRLF ;Zeilenvorschub
0157 3E3F               MVI     A,'?' ;Fehlermeldung ausgeben
0159 CD0603             CALL  AUSGABE
015C C31001             JMP     INPUT ;Neue Eingabe
                ;
                ;#########################################################
                ;#### <X>              Programmende                  ####
                ;#########################################################
015F C30000     ENDE    JMP      BOOT
                ;
                ;#########################################################
                ;#### <?>           Hilfstext ausgeben               ####
                ;#########################################################
0162 CDFB02     HELP    CALL     CRLF ;Zeilenvorschub
0165 110C04             LXI   D,HTEXT ;Hilfstext ausgeben
0168 0E09               MVI  C,STROUT
016A CD0500             CALL     BDOS
016D C31001             JMP     INPUT ;Neue Eingabe
                ;
                ;#########################################################
                ;#### <M>             Record ausgeben                ####
                ;#########################################################
0170 CDFB02     MARK    CALL     CRLF ;Zeilenvorschub
0173 21F0FF             LXI  H,0FFF0H ;Speicheradresse vorbereiten
0176 221303             SHLD  ADRESSE ;und merken
0179 CD6502             CALL   DMAOUT ;Record ausgeben
017C C31001             JMP     INPUT ; Neue Eingabe
                ;
                ;#########################################################
                ;#### <R>             Sektor einlesen                ####
```

```
                ;#########################################################
017F CD3602     LESE      CALL   HEXDEZ ;Naechste 2 Zeichen in DEX
0182 DA5401               JC      ERROR ;C-Flag=1 dann Fehler
0185 FE28                 CPI       28H ;Track >39
0187 D25401               JNC     ERROR ;Ja, dann Fehler
018A 4F                   MOV       C,A ;<C>=<A>
018B CD24C5               CALL SETTRACK ;<C>=Track
018E CD3602               CALL   HEXDEZ
0191 DA5401               JC      ERROR ;C-Flag=1 dann Fehler
0194 FE24                 CPI       24H ;Sector >35
0196 D25401               JNC     ERROR ;Ja, dann Fehler
0199 4F                   MOV       C,A ;<C>=<A>
019A CD29C5               CALL   SETSEC ;<C>=Sector
019D CDF7C2               CALL    READ  ;Sector einlesen
01A0 C37001               JMP     MARK  ;und ausgeben
                ;
                ;#########################################################
                ;#### <W>       Sektor zurueckschreiben              ####
                ;#########################################################
01A3 CD3602     SCHREIB   CALL   HEXDEZ ;Naechste 2 Zeichen in HEX
01A6 DA5401               JC      ERROR ;C-Flag=1 dann Fehler
01A9 FE28                 CPI       28H ;Track >39
01AB D25401               JNC     ERROR ;Ja, dann Fehler
01AE 4F                   MOV       C,A ;<C>=<A>
01AF CD24C5               CALL SETTRACK ;<C>=Track
01B2 CD3602               CALL   HEXDEZ ;Naechste 2 Zeichen in HEX
01B5 DA5401               JC      ERROR ;C-Flag=1 dann Fehler
01B8 FE24                 CPI       24H ;Sector >35
01BA D25401               JNC     ERROR ;Ja, dann Fehler
01BD 4F                   MOV       C,A ;<C>=<A>
01BE CD29C5               CALL   SETSEC ;<C>=Sector
01C1 CDFCC2               CALL    WRITE ;Sector zurueckschreiben
01C4 C31001               JMP     INPUT ;Neue Eingabe
                ;
                ;#########################################################
                ;#### <A>             Sektor aendern                  ####
                ;#########################################################
01C7 CDFB02     ALTER     CALL     CRLF ;Zeilenvorschub
01CA CD3602               CALL   HEXDEZ ;Holen vom Anfang
01CD DA5401               JC      ERROR ;Fehler
01D0 FE80                 CPI       80H ;Anfang >7FH
01D2 D25401               JNC     ERROR ;dann Fehler
01D5 47                   MOV       B,A ;Anfang ins <B>
01D6 CD3602               CALL   HEXDEZ ;Holen von Ende
01D9 DA5401               JC      ERROR ;Fehler
01DC FE80                 CPI       80H ;Ende >7FH
01DE D25401               JNC     ERROR ;dann Fehler
01E1 90                   SUB         B ;<A>=<A>-B Differenz
01E2 DA5401               JC      ERROR ;<A> > 0 ? Fehler
01E5 48                   MOV       C,B ;<C>=<B>
01E6 0600                 MVI     B,00H ;<B>=0
01E8 211503               LXI     H,DMA ;<HL> auf DMA
01EB 09                   DAD         B ;<HL>=<HL>+<BC>
01EC 47                   MOV       B,A ;<B>=<A>
01ED 79         ALTER1    MOV       A,C ;Adresse
```

```
01EE CDD502             CALL   HEXOUT ;ausgeben
01F1 3E3E               MVI     A,'>' ;Zeichen ausgeben
01F3 CD0603             CALL  AUSGABE
01F6 CDF502             CALL    SPACE ;Leerzeichen
01F9 7E                 MOV       A,M ;<A>=adressiertes Byte
01FA CDD502             CALL   HEXOUT ;Nach Hex wandeln/ausgeben
01FD CDF502             CALL   SPACE  ;Leerzeichen
0200 3E3A               MVI     A,':' ;Zeichen ausgeben
0202 CD0603             CALL  AUSGABE
0205 C5                 PUSH        B ;<BC> merken
0206 E5                 PUSH        H ;<HL> merken
0207 119503             LXI  D,PUFFER ;Puffereingabe
020A 0E0A               MVI   C,BUFIN ;Neuer Wert
020C CD0500             CALL     BDOS
020F 219603             LXI H,POINTER ;<HL>=Anzahl d. Zeichen
0212 7E                 MOV       A,M
0213 FE00               CPI       00H
0215 CA2D02             JZ     ALTER3 ;gleich 0? Ja, alter Wert
0218 CD3602             CALL   HEXDEZ ;Nach Dez wandeln
021B DA3102             JC     ALTER4 ;Fehler
021E E1                 POP         H ;<HL> wieder holen
021F 77                 MOV       M,A ;<A> nach adressiertes Byte
0220 23         ALTER2  INX         H ;<HL=<HL>+1
0221 C1                 POP         B ;<BC> wieder holen
0222 0C                 INR         C ;<C>=<C>+1 Adresse
0223 05                 DCR         B ;<B>=<B>-1 Zaehler
0224 CA1001             JZ      INPUT ;Ende des Aenderns
0227 CDFB02             CALL     CRLF ;Zeilenvorschub
022A C3ED01             JMP    ALTER1 ;Weitermachen
022D E1         ALTER3  POP         H ;Stapel korregieren
022E C32002             JMP    ALTER2 ;und weiter
0231 E1         ALTER4  POP         H ;Stapel korregieren
0232 C1                 POP         B
0233 C31001             JMP     INPUT ;und Ende
                ;
                ;*************************************************
                ;****          Umrechnung HEX nach DEZ            ****
                ;*************************************************
0236 23         HEXDEZ  INX         H ;Auf naechstes Zeichen zeigen
0237 7E                 MOV       A,M ;und in <A> laden
0238 CD4902             CALL  HEXDEZ1
023B D8                 RC            ;Ende wenn C-Flag=1 Fehler
023C 17                 RAL           ;<A> 4 mal nach rechts schieben
023D 17                 RAL
023E 17                 RAL
023F 17                 RAL
0240 4F                 MOV       C,A ;<C>=<A>
0241 23                 INX         H ;Naechstes Zeichen
0242 7E                 MOV       A,M ;in den <A>
0243 CD4902             CALL  HEXDEZ1
0246 D8                 RC            ;Ende wenn C-Flag=1 Fehler
0247 81                 ADD         C ;<A>=<A>+<C>
0248 C9                 RET
                ;
```

```
0249 FE30       HEXDEZ1   CPI       30H ;Kleiner ASCII 0
024B DA6302               JC    HEXDEZ3 ;dann Fehler
024E FE47                 CPI       47H ;Groesser ASCII F
0250 D26302               JNC   HEXDEZ3 ;dann Fehler
0253 FE41                 CPI       'A' ;Groesser ASCII A
0255 D26002               JNC   HEXDEZ2 ;dann OK
0258 FE3A                 CPI       3AH ;Groesser ASCII 9
025A D26302               JNC   HEXDEZ3 ;dann Fehler
025D D630                 SUI       30H ;<A>=<A>-30H
025F C9                   RET
0260 D637       HEXDEZ2   SUI       37H ;<A>=<A>-37H
0262 C9                   RET
0263 37         HEXDEZ3   STC
0264 C9                   RET
                ;
                ;#########################################################
                ;####                 DMA-Inhalt ausgeben               ####
                ;#########################################################
0265 11C303     DMAOUT    LXI D,STRING3 ;Kopfzeile ausgeben
0268 0E09                 MVI  C,STROUT
026A CD0500               CALL     BDOS
026D 0680                 MVI     B,80H ;<B>=80H Zaehler auf 128
026F 211503               LXI     H,DMA ;<HL>=DMA-Adresse
0272 CDB802     DMAOUT1   CALL   ADROUT ;Adresse ausgeben
0275 1610                 MVI     D,10H ;<D>=10H Zaehler auf 16
0277 7E         DMAOUT2   MOV       A,M ;<A> mit adressierten Byte laden
0278 CDD502               CALL   HEXOUT ;und als Hexzahl ausgeben
027B CDF502               CALL    SPACE ;Leerzeichen ausgeben
027E 23                   INX         H ;<HL>=<HL>+1
027F 05                   DCR         B ;<B>=<B>-1
0280 C28702               JNZ   DMAOUT3 ;Kein Recordende erreicht
0283 CD9102               CALL    ASCII ;16 Zeichen (ASCII) ausgeben
0286 C9                   RET           ;Recordende erreicht ENDE
0287 15         DMAOUT3   DCR         D ;<D>=<D>-1
0288 C27702               JNZ   DMAOUT2 ;Kein Zeilenende erreicht
028B CD9102               CALL    ASCII ;Zeilenende  ASCII-DUMP
028E C37202               JMP   DMAOUT1 ;Naechste Zeile
                ;
                ;#########################################################
                ;####          Ausgabe von 16 ASCII-Zeichen             ####
                ;#########################################################
0291 C5         ASCII     PUSH        B ;<BC> merken
0292 0610                 MVI     B,10H ;<B>=10H Zaehler auf 16
0294 2B         ASCII1    DCX         H ;<HL>=<HL>-1
0295 05                   DCR         B ;<B>=<B>-1
0296 C29402               JNZ    ASCII1 ;und das 16 mal
0299 0610                 MVI     B,10H ;<B>=10H Zaehler auf 16
029B 7E         ASCII2    MOV       A,M ;<A>=adressiertes Byte
029C FE20                 CPI       20H ;Vergleiche <A> mit 32 (<Space)
029E DAB302               JC     ASCII4 ;Kleiner, dann PUNKT
02A1 FE80                 CPI       80H ;Vergleiche <A> mit 128
02A3 D2B302               JNC    ASCII4 ;Groesser, dann PUNKT
02A6 CD0603     ASCII3    CALL  AUSGABE ;Zeichen ausgeben
02A9 23                   INX         H ;<HL>=<HL>+1
02AA 05                   DCR         B ;<B>=<B>-1
```

```
02AB C29B02             JNZ    ASCII2 ;<B>=0 Nein, dann weiter
02AE CDFB02             CALL     CRLF ;Ende. Zeilenvorschub
02B1 C1                 POP         B ;<BC> wieder holen
02B2 C9                 RET           ;Unterprogrammende
02B3 3E2E       ASCII4  MVI      A,'.' ;<A>=2EH
02B5 C3A602             JMP    ASCII3
                ;
                ;*************************************************
                ;****          Speicheradresse ausgeben           ****
                ;*************************************************
02B8 E5         ADROUT  PUSH        H ;<HL> merken
02B9 C5                 PUSH        B ;<BC> merken
02BA 2A1303             LHLD  ADRESSE ;<HL>=Adresse
02BD 011000             LXI   B,0010H ;<BC>=0010H
02C0 09                 DAD         B ;<HL>=<HL>+<BC>
02C1 221303             SHLD  ADRESSE ;und <HL> wieder speichern
02C4 7C                 MOV       A,H ;<A>=<H> Highbyte
02C5 CDD502             CALL   HEXOUT ;nach HEX wandeln und ausgeben
02C8 7D                 MOV       A,L ;<A>=<L> Lowbyte
02C9 CDD502             CALL   HEXOUT ;nach HEX wandeln und ausgeben
02CC CDF502             CALL    SPACE ;Leerzeichen ausgeben
02CF CDF502             CALL    SPACE ;Leerzeichen ausgeben
02D2 C1                 POP         B ;<BC> wieder holen
02D3 E1                 POP         H ;<HL> wieder holen
02D4 C9                 RET           ;Unterprogrammende
                ;
                ;*************************************************
                ;****      Umrechnung/Ausgabe <A> nach HEX        ****
                ;*************************************************
02D5 F5         HEXOUT  PUSH      PSW ;<A> merken
02D6 0F                 RRC           ;<A> 4 mal rechts schieben
02D7 0F                 RRC
02D8 0F                 RRC
02D9 0F                 RRC
02DA CDE202             CALL  HEXOUT1 ;Nibble ausgeben
02DD F1                 POP       PSW ;<A> wieder holen
02DE CDE202             CALL  HEXOUT1 ;Nibble ausgeben
02E1 C9                 RET           ;Unterprogrammende
02E2 E60F       HEXOUT1 ANI       0FH ;Bit 0-3 isolieren
02E4 FE0A               CPI       0AH ;Vergleiche <A> mit 10
02E6 D2EF02             JNC   HEXOUT2 ;<A> groesser 10
02E9 C630               ADI       30H ;<A>=<A>+30H - nach ASCII formen
02EB CD0603             CALL  AUSGABE ;und ausgeben
02EE C9                 RET
02EF C637       HEXOUT2 ADI       37H ;<A>=<A>+37H - nach ASCII formen
02F1 CD0603             CALL  AUSGABE ;und ausgeben
02F4 C9                 RET
                ;
                ;*************************************************
                ;****           Leerzeichen ausgeben          ****
                ;*************************************************
02F5 3E20       SPACE   MVI      A,' ' ;<A>=20H
02F7 CD0603             CALL  AUSGABE
02FA C9                 RET
                ;
```

```
                        ;*********************************************************
                        ;****              Zeilenvorschub                     ****
                        ;*********************************************************
02FB 3E0A               CRLF        MVI     A,0AH ;Line Feed
02FD CD0603                         CALL  AUSGABE ;ausgeben
0300 3E0D                           MVI     A,0DH ;Return
0302 CD0603                         CALL  AUSGABE ;ausgeben
0305 C9                             RET
                        ;
                        ;*********************************************************
                        ;****            Zeichen <A> ausgeben                 ****
                        ;*********************************************************
0306 C5                 AUSGABE     PUSH        B ;Register merken, da nach
0307 D5                             PUSH        D ;Einsprung ins BDOS alle
0308 E5                             PUSH        H ;undeferent.
0309 5F                             MOV       E,A ;<E>=<A> Zeichen ins <E> bringen
030A 0E02                           MVI  C,CONOUT ; und ausgeben
030C CD0500                         CALL     BDOS
030F E1                             POP         H ;Alle Register wieder holen
0310 D1                             POP         D
0311 C1                             POP         B
0312 C9                             RET
                        ;
                        ;*********************************************************
                        ;*       Speicherbereich fuer Strings & Adressen       *
                        ;*********************************************************
0313 0000               ADRESSE     DW 0000H
0315                    DMA         DS 80H
0395 05                 PUFFER      DB 05H
0396 00                 POINTER     DB 00H
0397                                DS 05H
039C 04021C0000 STRING1     DB 04H,02H,1CH,00H,00H,00H,1CH,01H,1AH,1AH
03A6 1D00004449             DB 1DH,00H,00H,'DISK-MONITOR  VER. 1.0'
03BF 0A0D0A24               DB 0AH,0DH,0AH,'$'
03C3 4144523E20 STRING3     DB 'ADR>  -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 '
03E7 2D41202D42             DB '-A -B -C -D -E -F 0123456789ABCDEF'
0409 0A0D24                 DB 0AH,0DH,'$'
040C 0A0D0A2A2A HTEXT       DB 0AH,0DH,0AH,'**** Befehlsuebersicht ****'
042A 0A0D                   DB 0AH,0DH
042C 5820202020             DB 'X     :Programmende',0AH,0DH
0440 3F20202020             DB '?     :Befehlsuebersicht',0AH,0DH
0459 4D20202020             DB 'M     :Zeigt Inhalt des Sektors (DMA)',0AH,0DH
047F 5226262626             DB 'R&&&&:Liest Track/Sektor ein',0AH,0DH
049D 5726262626             DB 'W&&&&:Schreibt DMA in Track/Sektor',0AH,0DH
04C1 4126262626             DB 'A&&&&:Aendert DMA von/bis',0AH,0DH,'$'
                        ;
04DD                    END

A>
```

```
ADR>  -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F 0123456789ABCDEF
0100  11 9C 03 0E 09 CD 05 00 11 15 03 0E 1A CD 05 00 ................
0110  CD FB 02 3E 3A CD 06 03 11 95 03 0E 0A CD 05 00 ...>:...........
0120  21 96 03 7E FE 00 CA 10 01 23 7E FE 58 CA 5F 01 !..~.....#~.X._.
0130  FE 4D CA 70 01 FE 3F CA 62 01 21 96 03 7E FE 05 .M.p..?.b.!..~..
0140  C2 54 01 23 7E FE 52 CA 7F 01 FE 57 CA A3 01 FE .T.#~.R...W.....
0150  41 CA C7 01 CD FB 02 3E 3F CD 06 03 C3 10 01 C3 A......>?.......
0160  00 00 CD FB 02 11 0C 04 0E 09 CD 05 00 C3 10 01 ................
0170  CD FB 02 21 F0 FF 22 13 03 CD 65 02 C3 10 01 CD ...!..".e.......

ADR>  -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F 0123456789ABCDEF
0180  36 02 DA 54 01 FE 28 D2 54 01 4F CD 24 C5 CD 36 6..T..(.T.O.$..6
0190  02 DA 54 01 FE 24 D2 54 01 4F CD 29 C5 CD F7 C2 ..T..$.T.O.)....
01A0  C3 70 01 CD 36 02 DA 54 01 FE 28 D2 54 01 4F CD .p..6..T..(.T.O.
01B0  24 C5 CD 36 02 DA 54 01 FE 24 D2 54 01 4F CD 29 $..6..T..$.T.O.)
01C0  C5 CD FC C2 C3 10 01 CD FB 02 CD 36 02 DA 54 01 ...........6..T.
01D0  FE 80 D2 54 01 47 CD 36 02 DA 54 01 FE 80 D2 54 ...T.G.6..T....T
01E0  01 90 DA 54 01 48 06 00 21 15 03 09 47 79 CD D5 ...T.H..!...Gy..
01F0  02 3E 3E CD 06 03 CD F5 02 7E CD D5 02 CD F5 02 .>>......~......

ADR>  -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F 0123456789ABCDEF
0200  3E 3A CD 06 03 C5 E5 11 95 03 0E 0A CD 05 00 21 >:.............!
0210  96 03 7E FE 00 CA 2D 02 CD 36 02 DA 31 02 E1 77 ..~...-..6..1..w
0220  23 C1 0C 05 CA 10 01 CD FB 02 C3 ED 01 E1 C3 20 #..............
0230  02 E1 C1 C3 10 01 23 7E CD 49 02 D8 17 17 17 17 ......#~.I......
0240  4F 23 7E CD 49 02 D8 81 C9 FE 30 DA 63 02 FE 47 O#~.I.....0.c..G
0250  D2 63 02 FE 41 D2 60 02 FE 3A D2 63 02 D6 30 C9 .c..A.`..:.c..0.
0260  D6 37 C9 37 C9 11 C3 03 0E 09 CD 05 00 06 80 21 .7.7...........!
0270  15 03 CD B8 02 16 10 7E CD D5 02 CD F5 02 23 05 .......~......#.

ADR>  -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F 0123456789ABCDEF
0280  C2 87 02 CD 91 02 C9 15 C2 77 02 CD 91 02 C3 72 .........w.....r
0290  02 C5 06 10 2B 05 C2 94 02 06 10 7E FE 20 DA B3 ....+......~. ..
02A0  02 FE 80 D2 B3 02 CD 06 03 23 05 C2 9B 02 CD FB .........#......
02B0  02 C1 C9 3E 2E C3 A6 02 E5 C5 2A 13 03 01 10 00 ...>......*.....
02C0  09 22 13 03 7C CD D5 02 7D CD D5 02 CD F5 02 CD ."..|...}.......
02D0  F5 02 C1 E1 C9 F5 0F 0F 0F 0F CD E2 02 F1 CD E2 ................
02E0  02 C9 E6 0F FE 0A D2 EF 02 C6 30 CD 06 03 C9 C6 ..........0.....
02F0  37 CD 06 03 C9 3E 20 CD 06 03 C9 3E 0A CD 06 03 7....> ....>....

ADR>  -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F 0123456789ABCDEF
0300  3E 0D CD 06 03 C9 C5 D5 E5 5F 0E 02 CD 05 00 E1 >........_......
0310  D1 C1 C9 00 00 CA 2D 02 CD 36 02 DA 31 02 E1 77 ......-..6..1..w
0320  23 C1 0C 05 CA 10 01 CD FB 02 C3 ED 01 E1 C3 20 #..............
0330  02 E1 C1 C3 10 01 23 7E CD 49 02 D8 17 17 17 17 ......#~.I......
0340  4F 23 7E CD 49 02 D8 81 C9 FE 30 DA 63 02 FE 47 O#~.I.....0.c..G
0350  D2 63 02 FE 41 D2 60 02 FE 3A D2 63 02 D6 30 C9 .c..A.`..:.c..0.
0360  D6 37 C9 37 C9 11 C3 03 0E 09 CD 05 00 06 80 21 .7.7...........!
0370  15 03 CD B8 02 16 10 7E CD D5 02 CD F5 02 23 05 .......~......#.

ADR>  -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F 0123456789ABCDEF
0380  C2 87 02 CD 91 02 C9 15 C2 77 02 CD 91 02 C3 72 .........w.....r
0390  02 C5 06 10 2B 05 00 94 02 06 10 7E 04 02 1C 00 ....+......~....
03A0  00 00 1C 01 1A 1A 1D 00 00 44 49 53 4B 2D 4D 4F .........DISK-MO
03B0  4E 49 54 4F 52 20 20 56 45 52 2E 20 31 2E 30 0A NITOR  VER. 1.0.
```

```
03C0  0D 0A 24 41 44 52 3E 20 20 2D 30 20 2D 31 20 2D ..$ADR>  -0 -1 -
03D0  32 20 2D 33 20 2D 34 20 2D 35 20 2D 36 20 2D 37 2 -3 -4 -5 -6 -7
03E0  20 2D 38 20 2D 39 20 2D 41 20 2D 42 20 2D 43 20  -8 -9 -A -B -C
03F0  2D 44 20 2D 45 20 2D 46 20 30 31 32 33 34 35 36 -D -E -F 0123456

ADR>  -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F 0123456789ABCDEF
0400  37 38 39 41 42 43 44 45 46 0A 0D 24 0A 0D 0A 2A 789ABCDEF..$...*
0410  2A 2A 2A 20 42 65 66 65 68 6C 73 75 65 62 65 72 *** Befehlsueber
0420  73 69 63 68 74 20 2A 2A 2A 2A 0A 0D 58 20 20 20 sicht ****..X
0430  20 3A 50 72 6F 67 72 61 6D 6D 65 6E 64 65 0A 0D  :Programmende..
0440  3F 20 20 20 20 3A 42 65 66 65 68 6C 73 75 65 62 ?    :Befehlsueb
0450  65 72 73 69 63 68 74 0A 0D 4D 20 20 20 20 3A 5A ersicht..M    :Z
0460  65 69 67 74 20 49 6E 68 61 6C 74 20 64 65 73 20 eigt Inhalt des
0470  53 65 6B 74 6F 72 73 20 28 44 4D 41 29 0A 0D 52 Sektors (DMA)..R

ADR>  -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F 0123456789ABCDEF
0480  26 26 26 26 3A 4C 69 65 73 74 20 54 72 61 63 6B &&&&:Liest Track
0490  2F 53 65 6B 74 6F 72 20 65 69 6E 0A 0D 57 26 26 /Sektor ein..W&&
04A0  26 26 3A 53 63 68 72 65 69 62 74 20 44 4D 41 20 &&:Schreibt DMA
04B0  69 6E 20 54 72 61 63 6B 2F 53 65 6B 74 6F 72 0A in Track/Sektor.
04C0  0D 41 26 26 26 26 3A 41 65 6E 64 65 72 74 20 44 .A&&&&:Aendert D
04D0  4D 41 20 76 6F 6E 2F 62 69 73 0A 0D 24 00 00 00 MA von/bis..$...
04E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
04F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................

End of dumping
```

8.0 Nachwort zur CPC-Praxis-Reihe

An dieser Stelle endet die 3-bändige Reihe CPC-464/664 Praxis. Ich hoffe, daß ich Ihnen mit dieser Reihe ein geschlossenes Bild über die relevanten Fähigkeiten der Schneider Computer CPC-464/664 gegeben habe.

Speziell dieser 3. Band zeigte, wie universell die Schneider Computer durch ihre CP/M-Fähigkeit eingesetzt werden können. Für viele unter Ihnen war sicherlich das CP/M-Betriebssystem Neuland. Ich hoffe aber, daß dieser 3. Band Sie leicht und umfangreich ins „professionelle Lager" der CP/M-Benutzer geführt hat.

Heute fällt immer mehr das Stichwort MS-DOS, als das (16-Bit-) Betriebssystem der Zukunft. Ohne Frage wird sich der 16-Bit-Prozessor innerhalb kurzer Zeit auch im Bereich der Billigcomputer (unter 3000.— DM) durchsetzen und damit wird auch das MS-DOS eine noch größere Verbreitung finden, als es heute schon der Fall ist. Heute aber ist und bleibt CP/M das relevante Betriebssystem für 8-Bit-Computer. Mit den hier erworbenen Kenntnissen ist ein ‚Umlernen' auf das Betriebssystem MS-DOS (oder andere) nur ein relativ ‚kleiner' Schritt, da viele Kommandos identisch oder ähnlich sind und viele der hier beschriebenen BDOS-Funktion auch unter MS-DOS aufrufbar sind. Mit dem CP/M-Betriebssystem haben Sie auf jeden Fall einen Schritt nach vorne getan.

Die 3 Bände enthalten 76 teilweise sehr umfangreiche Programme und unzählige Beispielprogramme. Diese Programme sollten Ihnen nicht nur zeigen, wie man etwas auf dem „Schneider" programmiert, sondern auch eine Reihe von (meines Erachtens) hervorragenden Anwendungsprogrammen zur Verfügung stellen. Ich habe versucht, eine gesunde Mischung zwischen ‚Lehrbuch und Programmsammlung' zu erzielen. Mein vorrangiges Ziel war es allerdings, Ihnen durch die Programme vor allem Anregungen für eigene Programme zu geben und Sie umfangreich über die Möglichkeiten der Schneider Computer zu informieren. Ich hoffe, daß mir dieses gelungen ist. Es würde mich freuen, Ihre Meinung darüber zu hören.

Noch ein Wort zu den Programmen: Der Dr. Alfred Hüthig Verlag hat sich entschlossen, ALLE Programme der Bände 1 bis 3 auf EINER Diskette anzubieten. Damit haben auch diejenigen unter Ihnen, die nicht die Zeit haben, alle Programme einzutippen, die Möglichkeit, sie zu benutzen. Die Programmdiskette ist mit fast 340 kByte auf beiden Seiten gefüllt und für 92,— DM über den

Dr.A.Hüthig Verlag
Postfach: 10 28 69
6900 Heidelberg

oder direkt bei Ihrem Buchhändler erhältlich.

Manfred Walter Thoma, Juni 1986

Register

Thomas Tai

Hüthig

Cracker, Hacker, Datensammler

Softwarepiraterie unter der Lupe

1986, 96 S., kart., DM 24,—
ISBN 3-7785-1236-6

Die zunehmende Softwarepiraterie im Homecomputer- und PC-Bereich hat die Hersteller von Computerprogrammen zu recht drastischen Schritten gegen die oft jugendlichen Kopierer gezwungen. Sie sehen in den Raubkopierern einen stark umsatzschädigenden Faktor, während die Beschuldigten ihr Hobby unter der Rubrik „Kavaliersdelikt" führen.

Das vorliegende Buch wird weder Tips und Tricks für Kopierer geben, noch ist es die Werbeschrift irgendeines großen Softwarehauses. Die recht unterschiedlichen Meinungen beider Parteien werden dargestellt, mit den entsprechenden Argumenten unterlegt und verglichen. Dabei erhält man Einblick in die Mühen einer Softwarefirma und in den „Computeruntergrund". Ein Kapitel über die Kampfmethoden gegen Kopierer rundet diese Bestandsaufnahme des Software(schwarz)marktes ab.

Weiterhin erfährt der Leser alles über die rechtliche Situation in Bezug auf den Urheberrechtsschutz von Software und bisherige Urteile zu dieser Problematik. Um Abmahnvereinen und manchmal recht dubiosen Vorgehensweisen von Rechtsanwälten vorzubeugen, werden dann noch Verhaltensregeln für den Fall einer Abmahnung oder Durchsuchung gegeben.

Dr. Alfred Hüthig Verlag
Im Weiher 10
6900 Heidelberg 1

Hüthig

Erika Hölscher

Logo auf dem Spectrum

1986, ca. 150 S., kart.,
ca. DM 35,—
ISBN 3-7785-1121-1

Am Beispiel des Sinclair-Logo wird in die Programmierung nach dem Top-Down-Prinzip eingeführt.

Abgesehen von den geringen Kenntnissen, die das Logo-Handbuch vermittelt, werden keine Grundlagen vorausgesetzt.

Logo ist ein sehr leistungsfähiges Programmierwerkzeug. Die vielfältigen Möglichkeiten dieser Sprache werden mit einfachen und dennoch interessanten Beispielen erläutert. Obwohl diese Sprache wegen ihrer Anschaulichkeit sehr für Kinder geeignet ist, findet auch der erfahrene Programmierer noch Neues. Das Sinclair-Logo unterscheidet sich nur unwesentlich von der ursprünglichen Logo-Idee. Zudem ist der ZX-Spectrum ein in seiner Klasse führender Computer bezüglich seiner Leistungsfähigkeit.

Dr. Alfred Hüthig Verlag
Im Weiher 10
6900 Heidelberg 1

Manfred Walter Thoma

Hüthig

CPC 464/664-Praxis

Band 2: Datenverwaltung

1985, ca. 150 S., kart.,
DM 34,—
ISBN 3-7785-1209-9

Anhand von fünf sehr umfangreichen und leistungsstarken Programmen findet im Band 2 der CPC-464/664 Praxis-Reihe eine grundlegende Darstellung der wichtigsten Methoden zur Datenverarbeitung der wichtigsten Methoden zur Datenverwaltung statt. Neben der Behandlung von Konstanten, der sequentiellen Datei und der RAM-Floppy (relative Adressierung) darf natürlich der Direktzugriff auf Daten nicht fehlen. Mit einem sehr komfortablen Diskettenmonitor wird der Aufbau der Diskette genauer untersucht. Doch auch Such- und Sortierverfahren sowie die Handhabung und Programmierung von Index-Dateien finden hier ihren Platz.

Die erarbeiteten Programme gehen weit über das übliche Niveau hinaus (sie sind, bis auf den Direktzugriff auch mit dem Datenrecorder nutzbar) und lassen sich in der alltäglichen Praxis hervorragend anwenden. Die Programme brauchen durch ihre Qualität einen Vergleich mit kommerziellen Programmen nicht zu scheuen.

Dr. Alfred Hüthig Ve
Im Weiher 10
6900 Heidelberg 1

Hüthig

Manfred Walter Thoma

CPC 464/664-Praxis

Band 1: Graphik

1985, 188 S., zahlr. Abb., kart.,
DM 34,—
ISBN 3-7785-1149-1

Der erste Band des dreibändigen Werkes beschäftigt sich schwerpunktmäßig mit den Graphikmöglichkeiten des Schneider CPC 464/664. Es ist besonders darauf geachtet worden, daß alle behandelten Probleme und Programmiertechniken einen konkreten Bezug haben. Alle Programme sind aus einem konkreten Bereich entwickelt und mit den behandelten Befehlen realisiert worden.
Neben unzähligen Beispielen findet der Benutzer viele Programme und Unterprogramme, die seine Arbeit mit dem Schneider CPC 464/664 erheblich erleichert. Gezeigt werden Programme von der Multicolorgraphik bis hin zum kompletten CAD-System, vom Tortendiagramm bis zum 3D-Funktionspilot, von der 3-dimensionalen Wertdarstellung bis zur Analoguhr. Alle Programme sind so ausgelegt, daß sie ohne große Schwierigkeiten auch speziellen Wünschen angepaßt werden können. Trotzdem handelt es sich bei diesem Buch nicht um eine Programmsammlung im eigentlichen Sinne. Es ist vielmehr ein Arbeits- und Handbuch, welches durch praktische Anwendungen (Programme) ergänzt ist, die den professionellen Charakter des Schneider CPC 464/664 deutlich unterstreichen.
Neben der Graphik ist weiterhin die Centronics-Schnittstelle mit mehreren Hardcopy-Programmen, der Joystick, die Tastatur und die Interruptsteuerung ausführlich behandelt. Die Kassetten- und Diskettenbedienung findet ebenso Platz wie alle Besonderheiten des BASIC-Interpreters.

Dr. Alfred Hüthig Verlag
Im Weiher 10
6900 Heidelberg 1

Thomas Tai

Hüthig

Computer-Abtenteuer

Das Adventurebuch für den CPC 464

1986, ca. 150 S., zahlr. Abb.,
kart., ca. DM 35,—
ISBN 3-7785-1235-8

Adventures erfreuten sich immer schon großer Beliebtheit und auch der CPC 464/664 Besitzer kann aus einem unüberschaubaren Angebot solcher Spiele wählen.

Dieses Buch zeigt dem Leser den Weg, wie er Adventures schneller lösen kann und auftretende Probleme überwindet. Es gibt Tips für immer wiederkehrende Situationen und gewährt einen Blick in die Karten der Adventureprofis, die ein Spiel oft schon nach einer Woche gelöst haben. Die abgebildeten und kopierbaren Entwurfsblätter dienen zur leichten Erstellung eines Lageplans und zur übersichtlichen Aufzeichnung aller wichtigen Spieldaten.

Verzweifelte Spieler werden die vollständigen Lösungen von drei populären Grafikadventures begrüßen, wobei die Anwendung der Entwurfsblätter noch einmal ganz deutlich wird. Zu den sehr komplexen Spielen DUN DURACH und LORDS OF MIDNIGHT werden Tips und Erläuterungen gegeben, so daß der Reiz dieser für die Ewigkeit bestimmten Programme nicht verloren geht.

Den Schluß bildet eine Anleitung zur Erstellung eigener Computer-Abenteuer mit dem berühmten QUILL ADVENTUREWRITER.

Dr. Alfred Hüthig Verla
Im Weiher 10
6900 Heidelberg 1

Durch seine CP/M-Fähigkeit eröffnet der Schneider CPC-464/664 dem Benutzer eine neue Welt vielschichtiger Möglichkeiten. Schritt für Schritt leitet dieses Buch den Leser hin zur selbstverständlichen und problemlosen Benutzung des Betriebssystems CP/M 2.2. Es ist dabei so konzipiert worden, daß es auch wie ein Nachschlagewerk benutzt werden kann.

Der erste Teil des Buches beschäftigt sich umfassend mit den CP/M-Kommandos sowie mit allen 22 CP/M-Dienstprogrammen (SETUB, SUBMIT etc.).

Im zweiten Teil des Buches findet eine grundlegende Einführung in die interne Arbeitsweise des Betriebssystems und in die Programmierung unter CP/M 2.2 statt. Sämtliche BDOS-Routinen werden detailliert an vielen Beispielprogrammen erklärt. An einigen umfangreicheren Programmen (DISKMON, HELPER, DUMP etc.) wird die praktische Anwendung der BDOS-Routinen vertieft.

AMSTRAD CPC

MÉMOIRE ÉCRITE
MEMORY ENGRAVED
MEMORIA ESCRITA

https://acpc.me/