CP/M 3

Ein Arbeitsbuch

Band 1

von Raoul O. Koerber

CP/M 3
Ein Arbeitsbuch in zwei Bänden

Band 1 — Zusammenstellung aller Befehle und Programme, die dazugehören,

aller BDOS-Aufrufe mit Erläuterungen,

Programmbeispiele

Band 2 — des BIOS mit Erläuterungen und Listing,

und allem was sonst noch dazugehört,

von

Raoul O. Koerber

Inhaltsverzeichnis

Vorwort

CP/M ist wohl das am meisten benutzte Betriebsystem für Mikrocomputer mit einer 8080- oder Z80-CPU. Es besticht einerseits durch die sehr einfache Handhabung, andererseits durch die große Menge vorhandener, meist preisgünstiger Programme, die ohne Änderung sofort einsetzbar sind.

Seit Gary Kildall es im Jahre 1974 der Öffentlichkeit vorstellte, hat CP/M immer mehr Freunde gewonnen, wenn auch nicht zu leugnen ist, daß für Viele (die meistens von Großrechnern kamen) die Oberfläche des Systemes zu dürftig, die Möglicheiten zu gering und die Flexibilität zu klein ist.

Schnell waren zwar Hilfsprogramme (z.B. SHELL) auf dem Markt die eine UNIX-ähnliche Umgebung schafften, doch alle Programme dieser Art verkleinerten nur die TPA, (den Arbeitsspeicher) die, bei CP/M 2 mit einer BIOS-Anpassung für doppelte Dichte, meist sowieso schon zu klein war.

Mit der Einführung von CP/M 3 (CP/M PLUS) wurden die meisten Wünsche der Benutzer erfüllt. Damit hat diese Betriebssystem sicher noch erheblich mehr an Bedeutung gewonnen, denn alles geht nun wesentlich komfortabler und schneller vonstatten und CP/M 2 Programme funktionieren noch nach wie vor, wenn auch mit kleinen, unbedeutenden Ausnahmen.

Zu CP/M 2 gibt es mittlerweile eine ganze Menge guter Handbücher in deutscher Sprache, zu CP/M 3 bisher noch nicht. Es hieße Eulen nach Athen tragen, das x-te Mal über CP/M 2 zu schreiben.

Aufgabe dieses Buches soll es daher sein vor allem CP/M 3 typisches herauszustellen, trotzdem soll es auch dem Anfänger als Einführung dienen können.

Es wird für den ernsthaften Benutzer nach wie vor notwendig sein, die (englisch-sprachigen) Original-Handbücher von Digital Research INC, dem Vertreiber von CP/M , zu haben, um das eine oder andere Detail, welches hier nicht ausführlich genug besprochen werden konnte, nachzuschlagen. Sicherlich können jedoch in vielen Fällen die Beispiel in den Listings die in diesem Buch abgedruckt sind (aus einem funktionierndem Gerät) oft weiter helfen.

Der Aufbau dieses Buches ist bewußt so gehalten, daß der Benutzer, je nach Wissensstand oder Notwendigkeit immer tiefer in die Materie eindringen kann, aber, selbst ohne Vorkenntnisse, schnell die wichtigsten Vorteile von CP/M 3 anzuwenden lernt.

Um den Text durch Querverweise nicht zu unübersichtlich zu machen, wurden viele, als allgemein bekannt geltende, Begriffe nicht erläutert oder, es werden Begriffe nur kurz angesprochen und in einem späteren Kapitel ausführlich erläutert. Im Glossar am Ende des Buches steht meist eine genauere Erläuterung oder ein Hinweis, wo in diesem Buch Näheres über den gesuchten Begriff zu finden ist.

Zur vollständigen Beschreibung von CP/M 3 gehört auch die Beschreibung des BIOS (Basic Input Output System), der Hardware-Anpassung an unterschiedliche Systemvoraussetzungen.

Es werden alle Notwendigkeiten eines CP/M 3 BIOS beschrieben und ein komplettes BIOS-Listing abgedruckt, wobei der NDR-Klein-Computer als Hardware dient. Zu diesem Computer steht reichlich Literatur zur Verfügung, sodaß eine Besprechnung der dort benutzten Hardware kein Thema in diesem Buch sein muß.

Wir beginnen in diesem Buch, wenn sich das Betriebssystem erstmals meldet mit:

```
CP/M V3.0 Loader
Copyright (C) 1982, Digital Research
```

Sollten Sie, verehrter Leser, (noch) kein funktionierendes Betriebssystem haben, hängt es evtl. (noch) an der Hardware-Anpassung, dann ist es vielleicht zweckmäßiger die nachfolgende Kapitel nur kurz zu 'überfliegen' und danach sofort zu Teil IV, (Band 2) der Hardware-Anpassung überzugehen.

Anmerkungen zur Schreibweise:

Dieses Manuskript wurde mit dem Text-Verarbeitungs-System WORD-STAR (R) auf einem ELZET-80 Mikrocomputer geschrieben.

Unter CP/M werden viele Schriftzeichen benutzt, die im Englischen eckige oder geschweifte Klammern darstellen, im Deutschen aber für die Umlaute Ä, Ö, Ü reserviert sind. Um trotzdem den Text sinnvoll ausdrucken zu können, wurde er zweigeteilt, wobei den einen Part teilweise der Computer selbst (mit dem Befehl PUT) übernahm, nämlich alles was 'ER' auf die Konsole von sich aus ausgeben kann.

Das eigentliche Manuskript ist mit Umlauten geschrieben, weil es sich so doch erheblich besser lesen läßt, alles was aber in den Bereich der Kommunikation mit dem Computer fällt, wurde Computergerecht (ohne Umlaute) geschrieben und ist in diesem Buch als Ausdruck mit einem Nadeldrucker dargestellt.

Und da wäre noch das Copyright...

Alle Listings werden (in unüblicher Weise) KOMPLETT veröffentlicht, um die Theorie durch Beispiele und Muster zu untermauern. Fehler, Irrtümmer und schlechter Programmierstil sind keineswegs ausgeschlossen.

Das Recht an allen Programmen, auch in veränderter Form, die in diesem Buch abgedruckt sind, soweit sie nicht zum Lieferumfang von CP/M (R) gehören oder, wie im Falle einfacher Treiberprogramme, Allgemeingut sind, liegen ausschließlich beim Verfasser. Gewerbliche Nutzung und Wiederveröffentlichung bedürfen der schriftlichen Abstimmung.

Kapitel 1 Der erste Schritt

Wenn ein CP/M 3 System erst einmal gebootet hat, steht der Anfänger meist etwas ratlos vor der lapidaren Meldung:

```
A>_
```

und einem daneben blinkenden (oder feststehenden) Cursor. Gibt man über die Tastatur einen beliebigen Text ein, so ist das zwar möglich, aber rühren tut sich normalerweise nichts. Erst wenn man die RETURN-Taste drückt kommt Leben in das System. Das könnte zum Beispiel wie folgt aussehen:

```
A>mach mal was
```

als Antwort folgt, nach kurzem Laufwerksgeklapper. in der nächsten Zeile:

```
MACH?

A>
```

Ganz offensichtlich hat CP/M nicht verstanden, was wir wollten.

Nun zunächst einmal eine Klarstellung: wer uns da 'geantwortet' hat, ist ein Teil von CP/M mit dem klangvollen Namen CCP (Console Command Prozessor - Konsolen Befehls Ausführer). Mit diesem Programmteil haben wir es immer dann zu tun, wenn CP/M kein eigenständiges Programm abarbeitet. Mehr zu diesem Thema später.

Hier stellt sich nun die Frage: "wie sag' ich's meinem Kinde?".

Die Beantwortung ist ganz einfach, der CCP kennt nur zweierlei Arten von Befehlen und zwar die 'eingebauten' Befehle und Befehle die als quasi eigenständige Programme auf der Diskette abgelegt sind. Programme dieser Art werden in diesem Zusammenhang als TPA-Befehle bezeichnet.

Die Hauptaufgabe des CCP's ist es, dem Benutzer Hilfestellung bei der Verwaltung von Disketten und Dateien zu geben, Hardware-Voraussetzungen zu ändern oder umzuschalten (z.B. Drucker. Bildschirmausgabe oder Stellen der Uhrzeit und des Datums usw.) und den Zugriff auf beliebige Programme zu ermöglichen.

Betrachten wir dazu erst einmal die 'eingebauten'

CCP-Befehle:

DIR	Ausgabe des Inhaltsverzeichnis ohne SYS-Dateien
DIRS	Ausgabe des Inhaltsverzeichnis der SYS-Dateien
ERA	Löschen einer Datei aus dem Inhaltsverzeichnis
REN	Umbenennen einer Datei im Inhaltsverzeichnis
TYPE	Ausgeben einer TEXT-Datei auf dem Bildschirm
USER	Umschalten in eine andere Benutzerebene

Meist können oder müssen dem Befehlswort noch nähere Angaben folgen, wie überhaupt eine strenge Syntax unter CP/M eingehalten werden muß; aber darauf kommen wir noch einmal ausführlich zurück.

Zuvor wollen wir die zweite Art von Befehlen vorstellen, die

TPA-Befehle:

Unter TPA (Transient Programm Area) versteht man einen Speicherbereich, in den Programme von der Diskette geladen und ausgeführt werden; verallgemeinert gesagt der Arbeitsspeicherbereich.

Unter CP/M 3 sind die eingebauten Befehle DIR, ERA, REN und TYPE zusätzlich als TPA-Programme vorhanden, tauchen in unserer Zusammenstellung also zweimal auf. Dies ist aber lediglich eine Option, wenn besondere 'Leistungen' von den eingebauten Befehlen verlangt werden. Ist der entsprechende TPA-Befehl nicht auf der Diskette, wird ganz einfach die 'besondere' Leistung' verweigert.

TPA-Befehle sind aus dem Inhaltsverzeichnis der jeweiligen Diskette zu ersehen, sie sind gekennzeichnet durch die Buchstaben COM als 'Anhängsel' an den Befehlsnamen. Sie unterscheiden sich im übrigen von 'normalen' Programmen nur durch die Tatsache, daß es sich dabei ausschließlich um Programme handelt, die in irgendeiner Art mit der Diskettenverwaltung zu tun haben.

Doch nun ist es an der Zeit einmal einen Befehl auszuprobieren, wir geben dazu ein:

```
A>dir

A: CPM3    SYS : CCP     COM : PIP     COM : SID     COM : ERASE   COM
A: TEST    Z80 : CHARIO  Z80 : DISKIO  Z80 : MBASIC  COM : TEST    BAS
A: FORMAT  COM : GENCPM  DAT : CHARIO  REL : BIOSKRNL REL : GEN     SUB
```

Ganz offensichtlich handelt es sich hier um die Ausgabe des Inhaltsverzeichnisses der gerade eingelegten Diskette.

Nicht alles was mit 'COM' endet sind TPA-Befehle, die meisten sind vielmehr sogenannte Programme, dazu jedoch später.

Eine zusätzliche Möglichkeit haben wir bisher völlig unerwähnt gelassen,

das Umschalten von Laufwerken:

Es ist trivial, muß aber erwähnt werden. Unter CP/M hat jedes Laufwerk 'seinen', wenn auch nicht gerade sehr phantasievollen Namen.

Das Laufwerk auf das gebootet wurde, hat üblicherweise den Namen 'A', weitere Laufwerksnamen sind 'B' ,'C' bis 'P'. Welche davon wirklich vorhanden sind, hängt vom Gerät ab. Dies sollte in der Gerätebeschreibung (des Computerherstellers) zu finden sein.

Nicht unbedingt kann dies durch Abzählen der sichtbaren Laufwerke geschehen, es kann sehr wohl eine RAM-Disk im Gerät sein oder

eine vorhandene Harddisk ist in zwei oder mehrere logische Laufwerke aufgeteilt. Dann müßte jeder dieser Einheiten ein eigener Laufwerksname zugeteilt werden.

Zurück zum Laufwerkswechsel:

Wird ein anderes Laufwerk gewünscht, wird einfach sein Name gefolgt von einem Doppelpunkt eingegeben und der CCP reagiert sofort:

```
A>b:
B>_
```

Ist ein Laufwerk (physikalische) nicht vorhanden, wird dies gemeldet mit:

```
A>b:
CP/M Error in B: invalid Drive
BDOS Function = 14
```

Leider 'spricht' CP/M (fast) nur Englisch. Gemeint ist hier ganz einfach, daß das Laufwerk 'C' ungültig, im Sinne von nicht vorhanden ist. Dies wurde mit der BDOS-Funktion 14 festgestellt. Fraglich in diesem Zusammenhang ist, was wohl eine BDOS-Funktion ist. Darauf wird sehr ausführlich in Teil III eingegangen, so sei hier also einfach einmal 'hingenommen'.

Soll in einem Befehl oder Programmaufruf ein anderes Laufwerk angesprochen werden, so kann dies auf zweierlei Art geschehen: entweder man spricht zuerst das Laufwerk wie oben gezeigt an oder spezifiziert in der Befehlszeile, welches Laufwerk gemeint ist:

```
B>dir a:

A: CPM3    SYS : CCP     COM : PIP     COM : SID     COM : ERASE   COM
A: TEST    Z80 : CHARIO  Z80 : DISKIO  Z80 : MBASIC  COM : TEST    BAS
A: FORMAT  COM : GENCPM  DAT : CHARIO  REL : BIOSKRNL REL : GEN     SUB
```

Ganz offensichtlich wird hier von Laufwerk 'B' aus das Inhaltsverzeichnis des Laufwerkes 'A' abgerufen.

Ehe wir weitere Befehle vorstellen, wollen wir uns nun etwas intensiver mit der Befehlszeile beschäftigen, also dem Schreibraum hinter dem Meldezeichen (dem sog. Prompter) 'A>'.

Dies ist die einzige Kommunikationsstelle zwischen Benutzer und dem CCP, hier erwartet der CCP 'seine' Befehle, jeweils abgeschlossen mit einem <CR>.

Kenner von CP/M 2 haben sich sicherlich oft genug geärgert, wenn sich in einer langen Befehlszeile ein Tippfehler eingeschlichen hatte, ...alles nochmal - dies hat unter CP/M 3 (leider nur in der gebräuchlichen Ausführung) ein Ende.

Digital Research hat CP/M 3 für die Befehlszeile einen Editiermodus beschert. Mit wenigen Wordstar-ähnlichen Kontrollsequenzen kann selbst ein bereits vom CCP als fehlerhaft zurückgewiesener Text editiert und neu verwendet werden. Die entsprechenden Kontrollsequenzen sind in Tabelle 1 zusammengefaßt.

Ein Hinweis:

Ein Kontrollzeichen z.B. CTRL-C wird durch gleichzeitiges Drücken der CTRL-Taste und der entsprechenden Zeichentaste erzeugt. Unbekannte oder nicht zum Zeileneditor gehörende Kontrollzeichen werden durch einen vorangestellten Aufwärtspfeil auf dem Bildschirm wiedergegeben.

CTRL A	bewegt den Cursor ein Zeichen nach links bis zum '>' ohne das entsprechende Zeichen zu löschen.
CTRL B	bewegt den Cursor an die Position direkt nach dem '>' ohne den Text zu verändern. Ist der Cursor bereits an dieser Stelle, wird er zum Textende bewegt.
CTRL E	erzwingt ein CR ohne dabei die Befehlszeile logisch abzuschliessen. Dies ist von Bedeutung, wenn man lange Befehlssequenzen eingeben muß, aber eine bessere Übersicht auf dem Bildschirm behalten möchte.
CTRL F	bewegt den Cursor ein Zeichen nach rechts, ohne das entsprechende Zeichen zu löschen.
CTRL G	löscht das Zeichen unter dem Cursor und rückt den Text rechts davon, nach links auf. Der Cursor bleibt an seiner momentanen Position.
CTRL H	oder BS (backspace) löscht ein Zeichen und bewegt den Cursor eine Position nach links. Die gleiche Funktion ist u.U. auch mit der DEL(ete)-Taste möglich.
CTRL I	oder TAB, bewegt den Cursor um eine TAB-Position nach rechts. TAB-Position ist jede 8. Spalte.
CTRL J	oder LF (linefeed) signalisiert das Befehlsende. Es hat den gleichen Effekt wie CNTRL-M. Dieses Zeichen kann an jeder beliebigen Cursorposition eingeben werden und veranlaßt dann, die Abarbeitung des kompletten Befehls in der Befehlszeile, d.h. auch der Text hinter diesem Zeichen wird abgearbeitet.
CTRL K	löscht die Restzeile ab der Cursorposition.
CTRL M	oder CR siehe CTRL-J.
CTRL R	schreibt den bisher editierte Text (der besseren Übersicht halber) mit allen vorgenommenen Korrekturen noch einmal.
CTRL U	löscht den Zeileninhalt und rückt eine Zeile tiefer.
CTRL W	bringt die 'alte' Befehlszeile zurück in den Zeilenpuffer zum Editieren. So können von CP/M reklamierte Fehleingaben einfach korrigiert werden.
CTRL X	löscht die Befehlszeile und setzt den Cursor direkt nach dem Prompter '>'

Tabelle 1 Editiersequenzen des Zeileneditors

Es müssen an dieser Stelle noch einige häufig benutzte Kontrollzeichen angeführt werden, die der Bildschirmausgabe und der Drukkersteuerung dienen. Sie sind in Tabelle 2 zusammengestellt.

Zur nachfolgenden Tabelle 2 noch eine Anmerkung für Umsteiger von CP/M 2. Unter CP/M 3 kann eine Textausgabe wie gewohnt mit CTRL S

angehalten werden, weiter geht es jedoch NUR mit CTRL Q. Ein Vorteil hat die Sache jedoch: nach CTRL S kann mit CTRL P der Drucker zu- oder abgeschaltet werden.

CTRL C	Zurücksetzen des Betriebssystems oder Abbruch einer Ausgabe. Wird nur an erster Stelle hinter dem '>' akzeptiert.
CTRL P	schaltet den Drucker ein (falls vorhanden) und gibt bis zu einem erneuten CTRL P alle Bildschirmausgaben auch auf den Drucker aus. Ist kein Drucker angeschlossen kann dies ein 'Hängenbleiben' verursachen
CTRL S	eine Textausgabe auf dem Bildschirm kann mit diesem Kontrollzeichen angehalten werden.
CTRL Q	mit diesem Zeichen läuft die Bildschirmausgabe wieder weiter.

Tabelle 2 Steuerzeichen für Bildschirm- und Druckerausgabe

Es ist nun an der Zeit einige immer wiederkehrende Begriffe vorzustellen und zu erläutern. Dazu noch einmal einen Auszug aus einem Inhaltsverzeichnis:

```
A>dir

A: CPM3     SYS : CCP      COM : PIP      COM : SID      COM : ERASE    COM
A: TEST     Z80 : CHARIO   Z80 : DISKIO   Z80 : MBASIC   COM : TEST     BAS
A: FORMAT   COM : GENCPM   DAT : CHARIO   REL : BIOSKRNL REL : GEN      SUB
```

Am Anfang jeder Zeile steht der Name des Laufwerkes, in welchem das Inhaltverzeichnis der dort eingelegten Diskette gelesen wurde. Dahinter kommt ein Name, meist gefolgt von einigen Leerzeichen, und dann noch mal ein Name. Nach einem Doppelpunkt folgt die gleiche Sequenz mit anderen Namen noch 4 mal, nur am Ende fehlt der Doppelpunkt.

Betrachten wir die Namen nun einmal genauer. Sie bestehen aus zwei Teilen:

1. dem eigentlichen Dateinamen

   Er darf bis zu 8 Zeichen enthalten, die eine sinnvolle Bezeichnung der entsprechenden Datei ergibt.

2. den Index (oder engl: extend)

   Er kann bis bis zu drei Zeichen enthalten und dient dazu, einen Dateinamen nach seiner Art zu spezifizieren. Der Index wird vom Dateinamen durch einen Punkt <.> getrennt.

Der Dateiname (wir wollen darunter ab sofort Namen und Index verstehen), darf nicht durch ein Leerzeichen getrennt sein. Alle darstellbaren Zeichen, außer den in Tabelle 3 zusammengestellten Sonderzeichen, die für bestimmte 'Regieanweisungen' benötigt werden, sind zugelassen.

Der Index kann u.U. auch entfallen, falls er nicht aus zwingenden Gründen (wie bei COM- oder SUB-Dateien) vorgeschrieben ist. Der Dateiname muß jedoch aus mindestens einem Zeichen bestehen.

< = , ! ¦ > [ ]	Zeichen die der Dateispezifikation dienen
TAB, SPACE	(=Leerzeichen) Beides beendet einen Dateinamen.
CR (Return)	Beendet eine Befehlzeile
:	Beendet eine Laufwerksangabe
.	Trennt Datei-Namen von Datei-Index
;	Trennt Datei-Namen von Passwort
* ?	gelten als JOKER (ersetzen jedes Zeichen)
< > & ! ¦ \ + -	Trennzeichen bei der Angabe von Optionen
[ ]	Trennzeichen fuer globale und lokale Optionen
( )	Trennzeichen fuer Mehrfachoptionen
/ &	Trennzeichen fuer Optionen in der (CCP-Befehlszeile
!	Trennzeichen fuer Mehrfachbefehle in der Befehlszeile
;	Trennzeichen am Beginn einer Befehls-Zeile fuer einen nachfolgenden Kommentar

Tabelle 3 Reservierte Zeichen
die nicht in Dateinamen verwendet werden dürfen.

Der CCP unterscheidet bei der Befehlseingabe nicht zwischen Groß- und Kleinbuchstaben, vielmehr werden alle Buchstaben (intern) in Großbuchstaben umgewandelt.

Als Namen für den Index sind einige Begriffe festgelegt, andere haben sich eingebürgert. Die meisten bleiben jedoch der Fantasie des Benutzers überlassen. Eine Zusammenfassung der unterschiedlichsten, jedoch häufig benutzten Indizes ist in Tabelle 4 zu finden.

COM	ein MUSS für alle ablauffähigen Programme
ASM	Index einer Assembler (Quellcode-) Datei
MAC	Index einer Assemblerdatei für den MAC-Assembler
TXT	Index für eine TeXT-Datei
DOC	für DOCumentation
SUB	ein MUSS für SUBmit-Dateien
REL	für linkbare Programmsegmente
HEX	für Dateien im (INTEL-) HEX-Format
BAK	von BAcKup abgeleitet, einer Sicherungsdatei wie sie von Editoren erstellt wird
BAS	Kennzeichnung einer BASIC-Datei
FOR	Kennzeichnung einer FORTAN-Datei
PAS	Kennzeichnung einer PASCAL-Datei
HLP	Overlay-Datei mit HILFE-Menüs
DAT	Eine Datei, die DATen (beliebiger Art) enthält
Z80	Kennzeichnung einer Assemblerdatei in Z80-Mnemonics

Tabelle 4 Gebräuchliche Abkürzungen des Dateinamen-Index

Es bleibt an dieser Stelle nur noch die Erläuterung des Begriffes DATEI. Prinzipiell ist darunter Alles zu verstehen, was unter einem Dateinamen auf einer Diskette abgepeichert werden kann. Es ist also ein Sammelbegriff für Daten ganz allgemein.

Für einige bestimmte Dateiarten sind jedoch feste Begriffe bekannt:

PROGRAMM ist eine Datei, die nach Aufruf eine festgelegte Arbeit übernimmt. Beispiele sind Editoren, Hochsprachen (BASIC, PASCAL usw), aber auch Assembler und Debugger, um nur einige zu nennen. Alle Programme sind durch den Index COM kenntlich gemacht.

Es gibt darüberhinaus eine bestimmte andere Programmart, das sogenannte

SUBMIT- Programm, gekennzeichnet durch den Index SUB. Mit seiner Hilfe ist es möglich verschieden CCP-Befehle in einer Datei zusammenzufassen und diese Befehlsfolge wie einen Befehl abzurufen. Als typisches Beispiel sei die folgende Submitdatei MAKESYS zur Generierung eines CP/M 3 Systemes vorgestellt:

```
Z80 SCB,BIOSKRNL,BOOT,MOVE,DISKIO,CHARIO
LINK BNKBIOS3[B]=SCB,BIOSKRNL,BOOT,MOVE,DISKIO,CHARIO
GENCPM AUTO
```

Anmerkung:

Z80.COM ist ein Assembler, der hier aufgerufen wird, um die Dateien SCB.Z80, BIOSKRNL.Z80, BOOT.Z80, MOVE.Z80 sowie DISKIO.Z80 und CHARIO.Z80 zu assemblieren und als 'REL'-Dateien abzulegen. Alle genannten Dateien werden danach durch das Programm LINK.COM in eine Datei BNKBIOS3.SPR ge'linkt'. Danach wird mit dem Programm GENCPM (AUTOmatisch) ein neues CPM3.SYS generiert. Alle 3 Zeilen könnten mit dem gleichen Ergebnis auch einzeln (hintereinander) in der Befehlszeile eingegeben werden. Da diese 'Operation' jedoch recht häufig vorkommt, wurden alle Befehle in einer Datei zusammen gefaßt - und können so mit einem Befehl, dem Dateinamen, aufgerufen werden.

Beide Programmarten werden durch einfache Namenseingabe ohne Angabe des Index aufgerufen.

Ein Programm kann u.U. auch nicht direkt ablauffähig sein, wie dies bei BASIC-Programmen der Fall ist, die erst in Zusammenarbeit mit einem BASIC-Interpreter funktionsfähig sind.

In diesem Falle wird erst die Hochsprache aufgerufen und dann erst aus dieser heraus das Programm. Häufig kann jedoch bereits in der Befehlszeile Hochsprachenname und Programmname eingegeben werden.

```
A>mbasic test
```

In diesem Beispiel wird das Programm MBASIC.COM aufgerufen, das wiederum das Programm TEST.BAS aufruft und ausführt.

Andere Dateien können nun beispielweise nur Text enthalten, man spricht dann von Textdateien (allgemein). Ist dieser Text z.B. ein Assembler-Programm, handelt es sich logischerweise um eine Assemblerdatei; bei einem BASIC-Programm um eine BASIC-Datei.

Es ist auf jeden Fall klar zu unterscheiden, ob es sich bei einer Datei um Daten im ASCII-Format handelt, also um direkt lesbare Zeichen (was auch für eine HEX-Datei gelten würde) oder um Daten in binärer Darstellung, also in einer Form wie sie direkt von der CPU interpretierbar sind. Letztere Form gilt im allgemeinen für alle Programme.

Diesen Unterschied zu (er)kennen ist wichtig zur Anwendung des Befehles TYPE und des Hilfsprogrammes DUMP. Mit TYPE werden Daten in ASCII ausgegeben - also Daten, die wie 'normaler' Text zu lesen sind. Mit DUMP dagegen werden Daten ausgegeben, die (von diesem Programm) erst in lesbare Form umgewandelt und dann auf den Bildschirm ausgegeben werden.

Dies sei an 2 Beispielen deutlich gemacht. (Alle Beispiele sind in der Ausgabe gekürzt, da sie nicht zur Auffüllung des Buches sondern zur Erläuterung benutzt werden sollen.)

```
A>dump ws.com

CP/M 3 DUMP - Version 3.0
0000: C3 08 2D C3 F8 2C C3 83 20 C3 19 2D 0B 00 60 00  ..-..,.. ..-...`.
0010: 18 00 00 00 00 00 00 20 20 43 4F 50 59 52 49 47  ....... COPYRIG
0020: 48 54 20 28 43 29 20 31 39 38 31 20 20 4D 69 63  HT (C) 1981  Mic
0030: 72 6F 50 72 6F 20 49 6E 74 65 72 6E 61 74 69 6F  roPro Internatio
0040: 6E 61 6C 20 43 6F 72 70 6F 72 61 74 69 6F 6E 20  nal Corporation
0050: 20 0F 00 20 20 4D 69 63 72 6F 50 72 6F 20 57 6F   .. MicroPro Wo
0060: 72 64 53 74 61 72 20 20 72 65 6C 65 61 73 65 20  rdStar  release
0070: 33 2E 30 30 20 20 73 65 72 69 61 6C 20 23 20 57  3.00  serial # W

A>type help.hlp

 ///Icommands

 CP/M 3 Command Format:

      A>COMMAND (command tail) <cr>

 A CP/M 3 command line is composed of a command, an optional
 command tail, and a carriage return. The command is the name or
 filename of a program to be executed. The optional command tail
 can consist of a drive specification, one or more file
 specifications, and some options or parameters.
```

Die Unterschiede sind recht deutlich zu erkennen. Bei DUMP erfolgt die Ausgabe immer in HEX auf der linken Bildschirmhälfte, in ASCII auf der rechten Bildschirmhälfte. Dabei ist gut zu erkennen, daß beim 'DUMP' zum Beginn ein Teil 'unleserlich' bleibt. Hier handelt es sich um die Darstellung in HEX-(adezimal) vom Maschinencode, dieser (nicht ausdruckbare) Code wird im ASCII-Teil durch einen Punkt <.> dargestellt.

Würde versucht, den Inhalt dieses Beispieles mit TYPE auf dem Bildschirm auszugeben, würden, je nach Art der Bildschirmausgabe,

die merkwürdigsten Dinge auf dem Bildschirm passieren. Es wäre nicht verwunderlich, wenn sich das ganze System 'aufhängen' würde.

Ganz links die Zahlen vor dem Doppelpunkt sind relative Adressangaben.

Bei den Programmen unterscheidet CP/M 3 selbst zwischen den 'normalen' Programmen und den sogenannten SYS(tem)-Programmen. Nirgends in den Unterlagen wird aber etwas darüber ausgesagt, was ein SYS-Programm ist und was nicht, es bleibt dies völlig dem Benutzer überlassen.

Die folgende Definition ist jedoch weit verbreitet und auch sinnvoll:

> Alle Hilfsprogramme auf einer Diskette, die das System an sprechen, (also die TPA-Befehle) und alle COM-Programme (z.B. Editoren Debugger und Assembler aber auch Hochsprachen) werden zu SYS-Dateien (mit dem TPA-Befehl SET.COM) erklärt. Sie können mit diesem Programm auch noch schreibgeschützt werden und 'verschwinden' damit aus dem 'normalen' Inhaltsverzeichnis.
>
> Ihr Vorhandensein kann mit dem CCP-Befehl DIRS überprüft werden.

Der Vorteil dieser Handhabung ist es, daß man im Inhaltsverzeichnis nur noch das Wesentliche hat. Es bleibt somit übersichtlich und auf den der Diskette zugewiesenen Arbeitsbereich beschränkt.

Werden Systemdateien eingeführt, meldet DIR und DIRS das Vorhandensein von Dateien aus dem 'anderen' Bereich mit der Meldung:

SYSTEM-FILES(S) EXISTS

Dateien, die als SYStem-files deklariert wurden, haben darüberhinaus noch einen besonderen Vorteil: sie können aus allen USER-Ebenen aufgerufen werden.

Zum weiteren Verständnis dieses Begiffes wenden wir uns jetzt dem Befehl USER zu:

Mit dem USER-Befehl kann der Speicherplatz einer Diskette oder Harddisk in bis zu 16 Benutzerebenen aufgeteilt werden. Darunter ist im Prinzip eine Aufteilung des Speicherraumes einer Diskette in Segmente mit einem 'eigenem' Inhaltsverzeichnis zu verstehen.

Diese Segmentierung einer Diskette ist eigentlich nur von Vorteil, wenn die Diskettenkapazität sehr groß ist oder wenn es sich um ein Mehr-Benutzersystem handelt - und das ist ein CP/M Gerät nun ja meist nicht.

Die Diskettenkapazität selbst verändert sich bei dieser Manipulation nicht. Auf 'normalen' Disketten bringt die Aufteilung in USER-Ebenen nur dann Vorteile, wenn z.B. zu Archiv-Zwecken Programme abgelegt werden, die u.U. gleiche Namen tragen. In diesem Falle würde sonst jedes Kopierprogramm die bereits vorhandene Datei gleichen Namens einfach überschreiben. Wird jedoch in

unterschiedliche USER-Ebenen kopiert, so weiß die eine Ebene nichts von der anderen Ebene.

Prinzipiell kann aus jeder USER-Ebene auf alle SYS-Programme zugegriffen werden, aber nicht (so ohne weiteres wenigstens) auf Programmteile, die in anderen USER-Ebenen liegen. D.h. allen Segmentbenutzern (den USERn) stehen alle Befehle und Programme mit gesetztem SYS-Attribut zur Verfügung, aber alle Dateien in 'seiner' Ebene nur dem jeweiligen USER. In diesem Zusammenhang muß der jeweilige USER natürlich nicht notgedrungen eine andere Person sein.

Ein sinnvolles Beispiel der Anwendung von USER-Ebenen ist der Fall, daß in USER-Ebene 1 alle notwendigen Dateien zur Generierung eines CP/M3 Systemes mit einer besonderen Hardware-Anpassung liegt, während in USER-Ebene 2 alle Dateien für eine völlig andere Anpassung liegen. Der Benutzer kann so allen Dateien ihren 'angestammten' Namen belassen und muß trotzdem kein verhängnisvolles Vertauschen befürchten.

Bei Harddisks mit großem Speicherbereich wird die Benutzung verschiedener USER-Ebenen erst so richtig sinnvoll. Dort kann z.B. in USER-Ebene 1 alles sein, was mit Textverarbeitung zu tun hat; in Ebene 2 alles für die Buchhaltung, in Ebene 3 alles was mit BASIC zu tun hat. So bleibt das Inhaltsverzeichnis immer übersichtlich und der Aufgabenbereich auf bestimmte Benutzerebenen beschränkt.

Nun noch zur Umschaltung in USER-Ebenen, dies ist genauso einfach wie die Laufwerksumschaltung:

```
A>2:
2A>_
```

Der CCP meldet sich nun zuerst mit der USER-Ebene (0..15) und dann mit dem Laufwerksnamen. Beim Ändern des Laufwerkes, wird die USER-Ebene 'mitgenommen':

```
2A>b:
2B>_
```

Es kann auch Laufwerk und USER-Ebene gleichzeitig gewechselt werden:

```
2B>8A
8A<_
```

Die USER-Ebene 0 wird nicht extra angezeigt. Alle eingebauten CCP-Befehle arbeiten in jeder USER-Ebene, aber alle sonstigen Programme und TPA-Befehle arbeiten nur wenn sie als SYS-Dateien deklariert wurden oder sich mit auf der USER-Ebene befinden. Ein Hilfsprogramm mit der Bezeichnung PIP.COM ermöglicht den Transfer zwischen den USER-Ebenen. Ein direkter Zugriff zwischen den USER-Ebenen ist nicht möglich.

Mit dem Befehl REN kann eine Datei im Inhaltsverzeichnis umbenannt werden (RENamed). Der Syntax hierzu ist ebenfalls recht einfach

```
A>ren test.asm=test.z80
```

In diesem Beispiel wird die (vorhandene) Datei TEST.Z80 in die Datei TEST.ASM umbenannt, von der Syntax her also:

Neuer Name=Alter Name

Nun noch ein 'gefährlicher' Befehl: ERA dieser Befehl löscht (ERAsed) eine Datei aus dem Inhaltsverzeichnis - ohne ganz spezielle Hilfsmittel unwiderruflich !!!

```
A>era test.asm
```

Es verbleibt für dieses einführende Kapitel nur noch eine Kleinigkeit, nämlich die Möglichkeit der Anwendung von JOKERN bei der Eingabe von Dateinamen:

1. Jeder Buchstabe kann durch ein Fragezeichen ersetzt werden. In diesem Falle wird jedes beliebige Zeichen an seiner Stelle akzeptiert.

   Beispiele:

   dir as??.??? zeigt z.B. Dateien wie

   ASM.COM, ASTI.TXT, AS.COM, AS.HEX, AS.ASM usw

2. Ein Namens-Rest kann durch einen Stern '*' ersetzt werden, in diesem Falle wird (intern) der gesamte Namensrest mit ????? aufgefüllt.

   Beispiele:

   dir *.COM zeigt alle Dateien mit dem Index COM
   dir A*.*   zeigt alle Dateien die mit A beginnen
   dir *.*   würde wiederum alle Dateien zeigen
              hätte also keinen besonderen Effekt

Jocker können bei allen CCP-Befehlen und bei den meisten Hilfsprogrammen eingegeben werden. Bei einigen Befehlen wird dazu der entsprechende TPA-Befehl benötigt. Ist er nicht vorhanden wird dies vom CCP gemeldet:

```
A>ren t*.asm=t*.z80

RENAME  COM required
```

Noch ein letzter Hinweis zur Befehlszeile:

Dateinamen werden von fast allen Befehlen und Programmen nur mit Name UND Index akzeptiert. Hiervon ausgenommen ist nur der Programmaufruf von COM und SUB Dateien.

Kapitel 2 Die Befehlserweiterungen des CCP

In diesem Kapitel beschäftigen wir uns mit den unter CP/M 3 zur Verfügung stehenden Erweiterungen des CCP-Befehlssatzes. Teilweise ist dazu die Voraussetzung mit anderen Hilfsprogrammen zu schaffen. Dies wird in jedem Falle erwähnt. Der Umgang mit den entsprechenden Programmen wird jedoch erst im übernächsten Kapitel behandelt. Es sei dem Benutzer jedoch überlassen für eigene Versuche die entsprechende Beschreibung bereits vorab zu lesen.

> Eines der wichtigsten Dinge bei einem Disketten-Betriebssystem ist es, daß unbedingt vor jedem Arbeiten mit unbekannten Dateien Sicherungskopien gemacht werden. Kein Software-Haus und kein Distributor wird durch Unachtsamkeit gelöschte oder sonst wie unbrauchbar gewordene Diskette zumindest ohne Zusatzkosten ersetzen.

Dem Einsteiger in CP/M soll daher an dieser Stelle dringend geraten sein, sich die vom Lieferanten seines Gerätes mitgelieferten Unterlagen noch einmal genau durchzulesen. Er sollte in der Lage sein, Disketten zu formatieren, die Systemspuren und alle Programme auf eine Arbeitsdiskette zu kopieren.

Die dazugehörenden Vorgänge werden zwar in diesem Buch prinzipiell besprochen, die entsprechenden Programme (FORMAT, COPYSYS oder wie immer der Distributor sie genannt hat) sind jedoch nicht einheitlich aufgebaut. Sie werden in den meisten Fällen vom Gerätehersteller, nicht von Digital Research, geschrieben und entziehen sich daher einer genauen Beschreibung (falls diese nicht mitgeliefert wurde). Die meisten Programme dieser Art sind jedoch Menü-geführt und 'narrensicher'. Diese Programme sind immer hardware-abhängig, also auch nur bedingt unter verschiedenartigen Systemen austauschbar.

Die nachfolgende Beschreibung des erweiterten Befehlssatzes erfolgt für jeden Befehl getrennt nach einer einheitlichen Syntax. Dieses Prinzip wird auch für die Beschreibung der Hilfsprogramme und 'Werkzeuge' beibehalten, um dem Benutzer einen schnellstmöglichen Überblick zu schaffen. Alle Beschreibungen beginnen am Anfang einer neuen Seite.

Eine wichtige Option sei an dieser Stelle ohne nähere Erläuterung bereits erwähnt. Viele Befehle und Hilfprogramme veranlassen Bildschirm-Ausgaben. Diese liefen unter CP/M 2 über den oberen Bildschirmrand hinaus falls sie nicht mit CTRL-S rechtzeitig angehalten wurden.

Unter CP/M 3 wird nur eine volle Seite ausgegeben, dann hält die Ausgabe an mit der Meldung:

Press RETURN to Continue

d.h. die nächste Seite wird nach drücken der RETURN (CR) Taste ausgegeben - meist tut es aber JEDE Taste außer CTRL-C, damit würde die Ausgabe abgebrochen. (Siehe hierzu auch Tabelle 2)

Bei manchen Gelegenheiten kann diese Option jedoch auch störend sein, wenn z.B. ein Text mit TYPE und CNTRL-P auf den Drucker ausgegeben werden soll. Hier bietet sich folgende Möglichkeit:

```
A)type test.asm[nopage]
```

Die Option NOPAGE (in eckiger Klammer)verhindert das Abfragen am Bildschirmende. Diese Option kann mit dem Hilfsprogramm SETDEF auch fest vorgegeben werden und explizit für bestimmte Ausgaben zurückgenommen werden.

```
A)type test.asm[page]
```

Noch etwas hat Digital Research seinem neuen CP/M (3) mitgegeben:

Es können üblicherweise mehrere Befehle in einer Zeile gegeben werden, dies ist einfach durch Trennen dieser Befehle mit einem Ausrufezeichen möglich.

```
A>dir!dirs
```

Es muß hier aber vermerkt werden:

bei manchen Programmen die CP/M nur als Vehikel für eigene Aufgaben verwenden ist dies nicht möglich, leider auch bei einem sehr bekannten, hervorragenden Assembler ...

Sogar Digital Research hat sich da selbst ein Bein gestellt, versuchen Sie es einmal mit der Sequenz:

```
A>dir!CNTRL-C
```

also: Aufruf des Inhaltsverzeichnis und danach CTRL-C zum Abbruch.

Sie sollten dies jedoch wirklich nur dann versuchen, wenn Sie wissen, wo bei Ihrem Gerät der RESET-Knopf ist ....

Anmerkung: in einem von DIGITAL RESEARCH genannten Patch ist dieser Fehler in der Zwischenzeit behoben - ist IHRE Version noch nicht geändert, fragen Sie Ihren Distributor !

****
* DIR(S) *
****

Der DIR(S)-Befehl

Syntax:

DIR (d:) (filename) ([Optionen])
DIRS(YS) (d:) (filename) ([Optionen])

Beschreibung:

Der DIR- oder DIRS-Befehl zeigt das Inhaltsverzeichnis einer Diskette mit den Diskettennamen, Attributen und sonstigen Informationen wie Dateigröße, Anzahl der logischen und physikalischen Blöcke, Anzahl der Dateien auf der Diskette und der von ihnen benutze Speicherraum usw.

In seiner einfachsten Anwendung ist DIR ein Befehlswort des CCP. Außer einem allgemeinen Inhaltsverzeichnis ist er in dieser Form nur in der Lage, Auszüge aus dem Inhaltsverzeichnis mit Hilfe von JOKERN auszugeben:

```
A>dir b:a*.*
A>dirs ?80.COM
```

Für den Benutzer sind u.U. aber weitere Informationen von Interesse. Der CCP-Befehl ruft bei weiterer Anforderung an sein 'Leistungsvermögen' die Datei DIR.COM auf. Findet er sie nicht, meldet er:

```
DIR      COM required
```

Mit dem Befehl DIR kann auf alle Dateien, die nicht als SYS-Dateien deklariert wurden, zugegriffen werden; mit DIRS oder DIRSYS entsprechend auf alle SYS-Dateien, ansonst unterscheiden sie sich in der Anwendung nicht.

Werden gesuchte Dateien nicht gefunden, wird dies gemeldet

```
NO File
```

Sind Dateien in der SYS- (bzw umgekehrt in der DIR-) Ebene vorhanden, die den Spezifikationen des gewünschten Dateinamen entsprechen, wird dies wie folgt gemeldet:

```
SYSTEM FILE(S) EXIST
NON-SYSTEM FILE(S) EXIST
```

In Tabelle 5 sind alle Optionen aufgeführt, die mit dem DIR(S)-Befehl möglich sind.

Auf der folgenden Seite jedoch zuerst noch einige Beispiele:

```
A>dir b:[full]

Scanning Directory...

Sorting  Directory...

Directory For Drive B:  User  0

    Name      Bytes   Recs   Attributes   Prot      Update          Create
------------ ------ ------ ------------ ------ ------------  -------------

BUCH           52k    405 Dir RW       None   12/07/85 16:39   12/07/85 16:10
BUCH     BAK   52k    405 Dir RW       None                    12/07/85 16:39
HELP     DAT   60k    477 Dir RW       None                    10/09/85 10:14
INS1           2k      1 Dir RW       None                    11/23/85 11:33
INSERT         4k     22 Dir RW       None   12/07/85 15:22   12/07/85 15:11
INSERT   BAK   4k     17 Dir RW       None                    12/07/85 15:22
TAB1     TXT   4k     17 Dir RW       None                    11/30/85  8:45
TABELLE  TXT   8k     50 Dir RW       None                    12/02/85 18:55
WS       COM   16k   124 Dir RW       None                    10/09/85 10:15
WSMSGS   OVR   26k   198 Dir RW       None                    10/09/85 10:15
WSOVLY1  OVR   34k   266 Dir RW       None                    10/09/85 10:16

Total Bytes     =    260k  Total Records =    1981  Files Found =   11
Total 1k Blocks =    253   Used/Max Dir Entries For Drive B:   32/ 256


A>dir b:[size]

Scanning Directory...

Sorting  Directory...

Directory For Drive B:  User  0

B: BUCH          52k : BUCH     BAK   52k : HELP     DAT   60k
B: INS1           2k : INSERT          4k : INSERT   BAK    4k
B: TAB1     TXT   4k : TABELLE  TXT   8k : WS       COM   16k
B: WSMSGS   OVR  26k : WSOVLY1  OVR  34k :

Total Bytes     =    260k  Total Records =    1981  Files Found =   11
Total 1k Blocks =    253   Used/Max Dir Entries For Drive B:   32/ 256
```

Das erste Beispiel zeigt ein volles (engl: full) Inhaltsverzeichnis.

Dazu wird der TPA-Befehl DIR.COM aufgerufen. Unter 'voll' ist in diesem Falle zu verstehen, daß jede Datei mit Angaben zu

1. Dateiname und Index (name)
2. Dateigröße in Kilobytes (Bytes)
3. Dateigröße in (log.) Sektoren (recs)
4. Attributen (DIR/SYS RW/RD usw) (Attributes)
5. Status des Paßwortschutzes (Prot)
6. Zeitpunkt der Überschreibung (update)
7. Zeitpunkt der Erstellung (create)

versehen wird. Darüberhinaus werden noch folgende Angaben gemacht, die den 'Belegungs'-Zustand der ganzen Diskette wiederspiegeln:

8. Belegter Speicherplatz (ges.) in Kilobytes (total Bytes)
9. dto in (log) Sektoren (total records)
10. Anzahl der Dateien im Inhaltsverzeichnis (Files found)
11. Belegung in 1k Blöcken (total 1k Blocks)
12. Belegte/Maximale Einträge (Used/max Dir Entries)

Im zweiten Beispiel werden mehr oder weniger die gleichen Angaben gemacht bis auf die Punkte 3..7.

ATT	Zeigt die (Benutzerdefinierten) Attribute F1, F2,F3 und F4.
DATE	Zeigt alle Dateien, die mit Datums- und Zeitmarken versehen sind. Sind diese Marken nicht aktiv, wird die Fehlermeldung: <Date and Time Stamping Inactive> ausgegeben.
DIR	Zeigt nur Dateien mit dem DIR-Attribut.
DRIVE=ALL	Zeigt das Inhaltsverzeichnis aller aktivierten Laufwerke.
DRIVE=(A,B,..)	Zeigt das Inhaltsverzeichnis der angegebenen Laufwerke.
EXCLUDE	Zeigt alle Dateien des Inhaltsverzeichnisses, die nicht der angegebenen Namensspezifikation entsprechen.
FF	gibt (falls der Drucker mit CNTRL-P zugeschaltet ist) ein FORM-FEED (Seitenvorschub) aus, bevor das Inhaltsverzeichnis ausgegeben wird. Ist die Option LENGTH mit angegeben, wird nach n Zeilen das FORM-FEED wiederholt.
FULL	Das Inhaltsverzeichnis wird vollständig ausgegeben, d.h. es wird neben den Dateinamen die Dateigröße, der benutzte und der Restspeicherplatz, die Anzahl der benutzten Einträge und evtl. vorhandene Datums- und Zeitmarken ausgegeben.
LENGTH=n	Gibt die Anzahl der Zeilen, die vor Ausgabe einer neuen Kopfüberschrift ausgegeben werden an. 'n' kann im Bereich 5...65536 Zeilen sein
NOPAGE	Das Inhaltsverzeichnis wird ohne anzuhalten ausgegeben.
NOSORT	Das Inhaltsverzeichnis wird nicht alphabetisch sortiert.
RO	Es werden nur Dateien mit gesetzem READ-ONLY Attribut ausgegeben.
RW	Es werden nur Dateien mit gesetztem READ-WRITE Attribut ausgegeben.
SIZE	Es werden alle spezifizierten Dateien mit der Dateigröße (in Kilobytes) ausgegeben.
SYS	Es werden alle Dateien mit gesetztem SYS-Attribut ausgegeben. (geht mit DIRS einfacher)
USER=ALL	Es werden alle Dateien, getrennt nach USER-Ebenen ausgegeben.
USER=n	Es werden alle Dateien in USER-Ebene n ausgegeben. (Innerhalb einer USER-Ebene genügt DIR alleine falls DIR.COM als SYS-Datei deklariert wurde.
USER=(0,1...)	Es werden alle Dateien in den genannten USER-ebenen ausgegeben.

Tabelle 5 Optionen zum DIR(S) Befehl

****
* ERA(SE) *
****

Syntax:

ERA(SE) (filename)[[C(onfirm)]

Beschreibung:

Mit dem ERA-Befehl kann eine Datei aus dem Inhaltsverzeichnis gelöscht werden. Werden JOKER verwendet, wird automatisch das Hilfsprogramm ERASE.COM aufgerufen. Ist dieses nicht vorhanden, wird sein Fehlen gemeldet.

Werden JOKER benutzt, wird nachgefragt, ob der durch JOKER definierte Dateityp wirklich gelöscht werden soll.

Bitte bedenken Sie, daß ein 'Wiedererwecken' einer einmal gelöschten Datei nur mit ganz speziellen Hilfsmittel, (die nicht zum Lieferumfang von CP/M 3 gehören) möglich ist.

Schreibgeschützte Dateien und Dateien, die mit einem PASS-Wort versehen sind, können nur gelöscht werden, wenn mit SET.COM der Schreibschutz aufgehoben wird, bzw. wenn das PASS-Wort bekannt ist.

Durch Anhängen einer eckigen Klammer gefolgt von einem 'C' oder ausgeschrieben CONFIRM, wird jeder der Spezifikation entsprechende Dateiname auf dem Bildschirm ausgegeben, gefolgt von der Frage, ob die Datei gelöscht werden solle oder nicht. Die Frage ist mit Y (für JA) und N (für NEIN) zu beantworten. Aussteigen aus der Abfragekette ist mit der Sequenz CNTRL-S,CTRL-C möglich.

Wird eine Datei nicht gefunden, so wird dies gemeldet mit dem lapidaren Text:

ERASE COM required

Nachfolgend noch ein Beispiel der 'Serienlöschung' mit Nachfrage:

```
ERA *.*[c]

B: HELP    .DAT (Y/N)? n
B: WSOVLY1 .OVR (Y/N)? n
B: BUCH    .   (Y/N)? n
B: WSMSGS  .OVR (Y/N)? n
B: WS      .COM (Y/N)? n
B: INSERT  .   (Y/N)? n
B: TAB1    .TXT (Y/N)? n
B: TABELLE .BAK (Y/N)? y
B: INSERT  .BAK (Y/N)? y
B: BUCH    .BAK (Y/N)? y
B: INS2    .   (Y/N)? y
B: TABELLE .TXT (Y/N)? n

B>
```

****
* REN(AME) *
****

Der RENAME-Befehl

Syntax:

```
REN(AME) neuername=altername
```

Beschreibung:

Mit dem CCP-Befehl REN kann eine Datei des Inhaltsverzeichnisses umbenannt werden.

Wird die gewünschte Datei nicht gefunden oder ist bereits unter dem gewünschten Namen eine Datei im Inhaltsverzeichnis abgelegt, folgt eine entsprechende Fehlermeldung:

```
A>rename ins1.txt=ins2.txt
No File

A>rename ins2.txt=ins1.txt
ERROR: Not renamed INS2  TXT file already exists, delete (Y/N)
```

Ist eine Datei die umbenannt werden soll, nicht vorhanden, so folgt das lapidare No File = Datei nicht vorhanden.

Wird ein bereits vorhandener Dateiname gewählt, so wird dies angezeigt und angefragt, ob die alte Datei gleichen Namens gelöscht werden soll (Y=JA, N=Nein). Wird die Frage verneint, wird keine Umbenennung vorgenommen. Im gezeigten Beispiel wurde die Datei INS2.TXT als bereits vorhanden gemeldet.

Werden JOKER im Dateinamen angegeben, wird vom CCP die Hilfsdatei RENAME.COM aufgerufen.

RENAME.COM kann auch direkt, ohne weitere Angaben (jedoch dann mit vollem Namen,) aufgerufen werden. Das Programm fragt dann selbst nach weiteren Angaben:

```
A>rename
Enter New Name: text.new
Enter Old Name: text1.new
Error: no such file to rename
A>
```

Es wird zuerst nach dem neuen Dateinamen gefragt (New Name), dann nach dem alten (bisherigen) Dateinamen (Old Name). Wird der alte Dateinamen nicht gefunden, wird dies, mit <Fehler: keine derartige Datei zum Umbenennen (gefunden)>, gemeldet und zum CCP zurückgekehrt.

****
* TYPE *
****

Der TYPE-Befehl

Syntax:

TYPE filename ([PAGE!NOPAGE])

Beschreibung:

Mit dem TYPE-Befehl können Textdateien auf dem Bildschirm oder via CTRL-P auf Bildschirm und Drucker ausgegeben werden.

Nicht für TYPE geeignet sind Dateien, die nicht druckbare Zeichen enthalten. (also Programmteile o.Ä.). Unter nicht abdruckbaren Zeichen sind alle Zeichen zu verstehen, die nicht in der ASCII-Tabelle aufgeführt sind (Zeichen über 7FH), oder von der Bildschirm- bzw. Druckerausgabe als CNTRL-Zeichen evtl. falsch interpretiert werden könnten (z.B. 00H...1FH).

Als Erweiterung des CCP-Befehles wird von diesem bei Bedarf das Hilfsprogramm TYPE.COM aufgerufen. Mit der Erweiterung ist es möglich auch mehrere Textdateien nacheinander durch Eingabe von JOKERN passender Art mit einem Befehl auszugeben.

Soll eine Datei gleichzeitig auf den Drucker ausgegeben werden, ist die Option NOPAGE nach einer eckigen Klammer, gefolgt von CTRL-P anzuhängen, um den Druckvorgang nicht durch die Bitte nach 'press RETURN to continue' zu unterbrechen (dieser Text würde übrigens auch auf den Drucker ausgegeben werden).

Es ist zugelassen auch nur TYPE einzugeben, in diesem Falle wird nach dem zu 'typenden' Dateinamen gefragt.

Enter filename:

Beim 'typen' einer Datei kann durch CNTRL-S die Ausgabe der Datei angehalten werden. Weiter geht es nach CNTRL-Q. Nach jeder Eingabe von CNTRL-S kann mit CNTRL-P der Drucker zu oder abgeschaltet werden. Auch in diesem Falle geht es jedoch erst mit CNTRL-Q weiter. Abgeschaltet werden kann TYPE durch CNTRL-C. Der Drucker muß, nach Gebrauch, mit CNTRL-P wieder explizit abgeschaltet werden.

Ist die Option PAGE wirksam, wird jeweils am Bildschirmende angehalten. Auch mit dieser Option bleibt die Kombination CNTRL-S, Q, P wirksam, am Bildschirmende wird dann jedoch zusätzlich angehalten.

Kapitel 4 Mehr Ordnung und Übersicht mit Hilfsprogrammen

CP/M 3 stellt dem Anwender eine ganze Palette von Hilfsprogrammen zur Verfügung,um Dateien zu schützen oder zu markieren, Datum und Uhrzeit als Organisationshilfe einzubinden und Systemparameter bereits beim Booten einzugeben oder je nach Bedarf zu verändern.

Eine sehr nutzvolle Eigenschaft ist die Tatsache, daß nach jedem Kaltstart (also nach jedem Booten) vom CCP geprüft wird ob eine Datei mit dem festgelegten Namen PROFILE.SUB auf der BootDiskette vorhanden ist. Ist dies der Fall, wird vor jeder weiteren Aktivität ausgeführt, was in dieser Datei verlangt wird. Dies ist vorteilhaft, um die Suchordnung innerhalb der Laufwerke und von COM- und SUB-Dateien vorzugeben. Ebenfalls gesetzt werden kann die Option PAGE oder NOPAGE (Siehe hierzu SETDEF). Weiterhin kann erzwungen werden, daß Datum und Uhrzeit gesetzt werden.

Bei manchen CP/M Systemen ist es möglich die Zeichensätze (englisch-deutsch) für die Bildschirmausgabe umzuschalten oder es kann veranlaßt werden, daß bereits nach dem Booten ein ganz bestimmtes Programm automatisch aufgerufen wird.

Für den geübten CP/M 3 Kenner ist es so möglich, komplexe Vorgänge quasi vorausschauend für einen Ungeübten zu übernehmen und diesem nur 'sein' Programm auf 'seiner' Diskette zuzuweisen.

So ist es heute mit CP/M 3 sehr einfach, den Computer in Büros oder Kleinbetrieben nur für ganz bestimmte Aufgaben einzusetzen z.B. mit einem Text-Verarbeitungssystem für die Sekretärin, einem Buchhaltungsprogramm für die Fakturistin und einem Datenbank-System für den Chef. ohne daß der Benutzer CP/M überhaupt jemals bewußt zu Gesicht bekommt.

Der System-Programmierer hat es mit PROFILE.SUB einfach in der Hand jedem Anwender die für ihn notwendigen Systemparameter auf ein im übrigen für alle geigneten Gerät anzupassen, ohne den Einzelnen mit Dingen zu belasten, die außerhalb seines normalen Interessenrahmens liegen. Es genügt dem Benutzer also vollständig die 'Gebrauchsanweisung' eines bestimmten Programmes zu lesen und ein wenig zu Üben. Von CP/M braucht er dabei keinerlei Ahnung zu haben.

Alle nachfolgenden Programme haben in irgendeiner Weise mit der Organisation und dem Schutz von Disketten und Dateien zu tun. Es sind Programme die Digital Research seinem CP/M 3-Paket beigefügt hat, um eine möglichst große Flexibilität zu erreichen.

Viele dieser Hilfsprogramme werden nie in die Hände des 'einfachen' Anwenders gelangen und viele werden auch von routiniertesten Anwendern nicht benötigt. Es bleibt auf jedem Fall dem Anwender überlassen, was er auf seinen Arbeitsdisketten benötigt und was er nur als 'stille Reserve' beiseite legt.

Die Hilfsprogramme sind nachfolgend in alphabetischer Reihenfolge aufgeführt. Zur Vereinfachung der Syntaxangaben sind in Tabelle 6 alle Syntaxregeln angegeben, soweit diese von allgemeiner Bedeutung sind.

Da es sich bei allen Programmen von der Bedeutung her mehr um Befehls-Erweiterungen handelt, werden sie in der Einzelbeschreibung auch als Befehle bezeichnet.

DIR	DIRectory Attribut
n	eine beliebige Zahl innerhalb des zugelassenen Zahlenbereiches
o	Bezeichnet eine Option
RO	READ-ONLY (nur Lesen - nicht beschreiben)
RW	READ-WRITE (Lesen und Schreiben)
s	Textzeile (String)
SYS	SYStem Attribut (Datei kann aus allen USER-Ebenen ereicht werden)
(..)	Angaben innerhalb geschweifter Klammer sind optional,muessen also nicht angegeben werden
[..]	Angaben in eckigen Klammern sind Optionen, werden sie genutzt, muessen die eckigen Klammern mit angegeben werden. Die abschliessende Klammer kann (meist) entfallen
(...)	Angaben eines bestimmten Bereiches werden innerhalb runder Klammern angeben. Die Klammern muessen angegeben werden.
...	Gibt an, dass eine vorangegangene Angabe wiederholt werden kann.
^ oder CNTRL	Gibt an, dass der nachfolgende Buchstaben zusammen mit der CNTRL-Taste gedrueckt werden muss. CNTRL-Zeichen koennen auf dem Bildschirm nicht anders dargestellt werden - sie dienen nur zu Steuerzwecken der Bildschirm- (oder Drucker-) Ausgabe.
<cr>	Repraesentiert die RETURN-Taste
?	JOKER bei der Angabe von Dateinamen, ersetzt jeden beliebigen Buchstaben
*	JOKER er ersetzt alle nachfolgenden Buchstaben innerhalb eines Dateinamens oder Index durch Fragezeichen.
;	trennt verschiedene Moeglichkeiten in der Syntaxangabe. Es kann jedoch nur jeweils eine Moeglichkeit gewaehlt werden.
!	Trennt verschiedene Befehle in einer Befehlszeile. (Wird von manchen Programmen, die Befehlserweiterungen in der Befehlszeile zulassen, jedoch falsch interpretiert!!!)

Tabelle 6 Abkürzungen und Symbole in den Syntax-Angaben

****
* COPYSYS *
****

Der COPYSYS-Befehl

Syntax:

COPYSYS

Beschreibung:

Dieses Programm dient dazu, die vom Betriebssystem reservierten Systemspuren zu kopieren.

Auf den Systemspuren befindet sich bei CP/M 3 lediglich der CPM-Loader, ein Programm. welches seinerseits die Datei CPM3.SYS, das eigentliche, benutzerspezifisch konfigurierte Betriebs-System lädt und nicht wie bei CP/M2, das komplette Betriebssystem.

Dies gibt dem Benutzer die Möglichkeit, 'sein' CP/M sehr schnell beinahe beliebigen Anforderungen anzupassen ohne auf die Größe des Programmes (paßt es auf die Systemspuren ?) achten zu müssen. Der Loader kann, einmal auf die hardware eines Systemes abgestimmt, immer der gleiche bleiben. Er muß jedoch auf jeder Diskette, auf die gebootet werden soll. vorhanden sein !

Die Datei COPYSYS.ASM wird allen von Digital Research zugelassenen Distributoren zur Verfügung gestellt, um hardware-spezifisch angepasst zu werden. Sie wird dann als COM-Datei (vielleicht auch unter anderem Namen) an den Benutzer weiter gegeben.

Der Umgang damit ist sehr einfach wie das folgende Beispiel zeigt:

```
A>copysys

Copysys Ver 3.0

woher kopieren ( oder <cr> fuer login-Laufwerk ) :A
kopieren von A dann <cr> : <CR>
Funktion beendet
wohin kopieren ( oder <cr> zum Abbruch ) :B
kopieren nach B dann <cr> : <CR>
Funktion beendet
soll CPM3.SYS kopiert werden (J/N)? j
woher kopieren ( oder <cr> fuer login-Laufwerk ) : <CR>
kopieren von A dann <cr> : <CR>
Funktion beendet
A>
```

Anmerkung:

Nicht alle Distributoren haben sich die Mühe gemacht den Originaltext in deutsche Sprache zu übersetzen - es kann also sehr wohl möglich sein, daß Ihr COPYSYS englisch 'spricht', der sinngemäße Inhalt ist aber der gleiche.

Unter LOGIN-Laufwerk ist immer das Laufwerk zu verstehen, das vom Prompter des CCP's angezeigt wird.

****
* DATE *
****

Syntax:

DATE (c(ontinuous)
DATE set
DATE mm/dd/yy hh:mm:ss

Beschreibung:

Mit dem DATE-Befehl ist es möglich, Datum und Uhrzeit zu stellen oder abzurufen. Dies setzt vom Prinzip her natürlich die entsprechende Hardware voraus, kann aber auch ohne diese Bedeutung haben.

Es wird in jedem Falle das eingegebene Datum und die Uhrzeit abgelegt. Diese Daten bleiben im System erhalten bis zum Abschalten oder RESET.

Befehlen oder Programmen, die diese Daten verwenden, stehen die eingegebenen Werte zur Verfügung, auch wenn die Uhr nicht 'läuft' (oder jedesmal von Hand gesetzt werden muß). Als Ordnungselement sind die eingegebenen Daten aber auch in diesem Falle immer noch zu verwenden.

Die Eingabefolge ist leider 'amerikanisch' wie aus dem Syntax zu ersehen ist. Es sind jedoch CP/M3 Systeme auf dem Markt bei denen DATE.COM zu DATUM.COM wurde, mit 'deutschem' Syntax. Es handelt sich dabei um 'gepatchte' oder selbstgeschriebene Programme.

Nachfolgend einige Beispiele und mögliche Fehlermeldungen:

```
A>date

Fri 12/27/85  11:04:55

A>date set

Enter today's date (MM/DD/YY):12/27/85

Enter the time (HH:MM:SS): 11:15:00

Press any key to set time
```

Nach Eingabe des Datums in der Folge Monat, Tag, Jahr und der Uhrzeit in der Folge: Stunde, Minute, Sekunde (24-Stunden!) werden alle Daten erst nach Drücken einer weiteren, beliebigen Taste (... Press any key ..) Übernommen, die Uhr kann also sekundengenau gestellt werden.

Es steht noch eine weitere Option zur Verfügung, ein 'C' oder der Text 'CONTINUOUS'. Sie ist nur sinnvoll, wenn eine Sekundenanzeige zur Verfügung steht. Die Uhrzeit wird dann solange in der Befehlszeile ausgegeben, bis eine beliebige Taste gedrückt wird. Abgebrochen wird die Sekundenausgabe mit jeder Taste oder nach dem von Digital Research genannten Patch mit CNTRL-S - CNTRL-C.

```
**********************
* DEVICE             *
**********************
```

Der DEVICE-Befehl

Syntax:

```
DEVICE (names¦values¦physikalische Einheit¦logische Einheit)
DEVICE logische Einheit=physikalische Einheit(Option)(,phys-Einheit(o),...
DEVICE logische Einheit=NULL
DEVICE physikalische Einheit (Option)
DEVICE CONSOLE[page]COLUMES=n¦LINES=n
```

Beschreibung:

Mit DEVICE.COM ist es möglich die unter CP/M3 ansprechbaren fünf logischen EIN-/AUS-Gabe-Einheiten

CONIN	=Konsolen Eingabe
CONOUT	=Konsolen Ausgabe
AUXIN	=Zusatzkanal EIN
AUXOUT	=Zusatzkanal AUS
LST (LIST)	=Druckerkanal

mehr oder weniger beliebigen anderen physikalischen Kanälen zuzuordnen.

Welche logischen Einheiten und welche physikalischen Einheiten vorhanden sind und wie diese belegt sind, ist durch den Befehl DEVICE schnell feststellbar:

```
A>device

Physical Devices:
I=Input,O=Output,S=Serial,X=Xon-Xoff
CONSOL 9600  IOS    VIDEO  NONE  0      CON2   9600  IOS
PPRINT NONE  0      SPRINT 9600  OS     SIO2A  9600  IOS
SIO2B  300   IOS

Current Assignments:
CONIN:  = CONSOL
CONOUT: = CONSOL
AUXIN:  = CON2
AUXOUT: = CON2 .
LST:    = PPRINT

Enter new assignment or hit RETURN

CONOUT=CONSOL,VIDEO
```

Der Unterschied zwischen logischen Einheiten und physikalischen Einheiten ist recht einfach zu verstehen:

Logische Einheiten sind die 5 Einheiten, die CP/M zuläßt:

CONIN, CONOUT, AUXIN, AUXOUT und LST.

Unter physikalische Einheiten versteht man die Einheiten, welche auch wirklich (... eben physikalisch) vorhanden sind und mit einem entsprechenden Treiberprogramm (das die Ein/Ausgabe auch ermöglicht) in die Datei CHARIO eingebunden sind.

Es sind theorethisch 16 physikalische Einheiten möglich, praktisch sind jedoch nur 14 Einheiten zugelassen (zwei Zuweisungen sind als Reserve vorgesehen).

Die Namen der logischen Einheiten sind unveränderbar, während die Namen der physikalschen Einheiten dem Benutzer (zumindest jedoch dem Programmierer) überlassen bleiben. Die Namen der logischen Einheiten sind hierfür jedoch nicht zugelassen.

Meist werden Namen gewählt die (in Steckkarten-Systemen eine bestimme Systemkarte oder) einen bestimmten Baustein bezeichnen, z.B. SIO oder Namen, die einen bestimmten Zweck der Schnittstelle andeuten, z.B. MODEM.

Die nachfolgenden Beispiele zeigen die Anwendung des Befehls. Die möglichen Optionen sind in Tabelle 7 zusammen gefaßt.

Fehler, die eine unzulässige oder physikalisch unmögliche Zuweisung betreffen, werden angezeigt, wie im Beispiel der AUX-Zuweisung, wenn z.B. eine Baudraten-Umschaltung (software-mäßig) nicht möglich ist.

Die Umschaltmöglichkeiten hängen immer davon ab, ob das entsprechende Gerät auch physikalisch (also greifbar) vorhanden ist; die Umschaltung der Baudrate davon, ob diese über Software gesteuert werden kann. Auskünfte gibt normalerweise das Benutzerhandbuch des CP/M-Gerätes oder die bereits schon genannte Datei CHARIO.

```
A>DEVICE AUXIN:=SIO2B[XON,300]

Error at the '^'; Baud rate can not be set for this device

A>DEVICE console[columns=40,lines=16]

Console width set to 40 columns
Console page set to 16 lines

A>dir

A: STATLINE INC : DISKIO   REL
A: CPM3     SYS : BOOT     REL
A: BNKBIOS3 SPR : SCB      Z80
A: Z80ASM   COM : DISKIO   BAK
A: STIME    INC : RESBDOS3 SPR
A: BNKBDOS3 SPR : GENCPM   DAT
A: CHARIO   REL : BIOSKRNL Z80
A: SUB      SUB : CLOCK1   INC
A: CPM3     INC : RECALL   COM
A: HDISK    INC : BOOT     Z80
A: INS1         : FBIOS    REL
A: CLOCK2   INC : MOVE     REL

A>
```

Nach der Umschaltung auf 40 Spalten und 16 Zeilen, zeigt der DIR-Befehl deutlich. was mit dieser Umschaltung erreicht wurde: die DIR-Ausgabe ist nur noch halb so 'breit'.

logische EIN/AUSgabe-Einheiten:	
CONIN:	Konsolen Eingabe (Tastatur oder Terminal).
CONOUT:	Konsolen Ausgabe (Bildschirm oder Terminal).
AUXIN:	AUXiliare (allgemeine, zusätzliche) Schnitt-
AUXOUT:	stelle, z.B. zur Verbindung mehrerer Computer oder als MODEM-Schnittstelle gedacht.
LST:	Druckeranschluss.
Abrufbare Optionen:	
NAMES	Es werden die Namen aller belegeten physikalischer Einheiten und deren Charakteristik ausgegeben.
VALUES	Es wird die augenblickliche Zuweisung der logischen Einheiten ausgegeben.
phys Einheit	Es werden alle Attribute der genannten phys. Einheit ausgegeben. Mögliche Namen sind in der BIOS-Datei CHARIO deklariert.
log Einheit	Es werden alle Attribute und Zuweisungen der genannten logischen Einheit ausgegeben. Mögliche Namen siehe oben.
Einstellbare Optionen:	
XON	Für die benannte Einheit wird ein XON/XOFF-Protokoll benutzt.
NOXON	Für die benannte Einheit wird kein XON/XOFF-Protokoll benutzt.
nnnn	Für die benannte Einheit wird die Baudrate nnnn benutzt. Es sind folgende Baudraten zugelassen: 50,75,110,134,150,300,600,1200 1800,2400,3600,4800,7200,9600,19200 Die Baudrate kann nur geändert werden, wenn dies in der BIOS-Datei CHARIO ausdrücklich zugelassen wird, in diesem Fall muß ein entsprechender software-Treiber vorhanden sein

Tabelle 7 Zuweisungen und Optionen des DEVICE-Befehles

****
* DUMP *
****

Syntax:

DUMP filename.ext

Beschreibung:

Genaugenommen handelt es sich hier nicht um einem Befehl,sondern um ein Programm, das zur schnellen Überprüfung von Dateien, die nicht als Text-Dateien zur Verfügung stehen, dient.

'Ältere' Programme erlauben nur eine Darstellung im HEX-Format. Neuere Programme stellen die Daten in HEX- und ASCII dar. Die Option PAGE oder NOPAGE ist bei den meisten Programmen dieser Art wirkungslos, weiter geht es meist auch NUR mit der Return-Taste, falls diese Option überhaupt vorgesehen ist.

In Teil IV (Beispiele und Anwendungen) ist ein DUMP-Programm als Listing abgedruckt.

```
A>dump ws.com

CP/M 3 DUMP - Version 3.0
0000: C3 08 2D C3 F8 2C C3 B3 20 C3 19 2D 0B 00 60 00  ..-..,.. ..-..`.
0010: 18 00 00 00 00 00 00 20 20 43 4F 50 59 52 49 47  ....... COPYRIG
0020: 48 54 20 28 43 29 20 31 39 38 31 20 20 4D 69 63  HT (C) 1981  Mic
0030: 72 6F 50 72 6F 20 49 6E 74 65 72 6E 61 74 69 6F  roPro Internatio
0040: 6E 61 6C 20 43 6F 72 70 6F 72 61 74 69 6F 6E 20  nal Corporation
0050: 20 0F 00 20 20 4D 69 63 72 6F 50 72 6F 20 57 6F   .. MicroPro Wo
0060: 72 64 53 74 61 72 20 20 72 65 6C 65 61 73 65 20  rdStar  release
0070: 33 2E 30 30 20 20 73 65 72 69 61 6C 20 23 20 57  3.00  serial # W
```

****
* GENCOM *
****

Der GENCOM-Befehl

Syntax:

```
GENCOM (COM-Dateiname)(RSX-Dateiname)...([LOADER;NULL!SCB=(offset,wert)])
```

Beschreibung:

Mit diesem Befehl ist es möglich, RSX-Dateien an ein Programm anzuhängen oder wieder abzuhängen.

Die Funktion und der Zweck von RSX-Dateien kann sehr vielfältig sein. An dieser Stelle sei nur erwähnt, daß es sich bei RSX-Dateien (Resident System eXtensions) um eine CP/M 3 spezifische Möglichkeit handelt, ganz bestimmte anwendertypische Erweiterungen des BDOS oder des BIOS in das System zu integrieren.

RSX-Dateien werden vor den Beginn des residenten (ungebankten Teiles) des BDOS geladen, die BDOS-Einsprungadresse auf Adresse 5 (in der TPA) wird ebenfalls auf eine entsprechende Adresse in der RSX-Datei umgeändert. Vor jedem BDOS-Aufruf kann nun im RSX-Programm geprüft werden, ob evtl. vor (oder nach) dem Aufruf irgendetwas veranlaßt werden soll.

Die vielfältigen Möglichkeiten werden dem geübten Programmierer schnell ins 'Auge springen', den Anfänger jedoch nur völlig verwirren. Es sei daher auf Teil III verwiesen, in dem näher auf die Möglichkeiten von RSX-Dateien eingegangen wird.

Das 'Anhängen' einer RSX-Datei ist nur an COM-Dateien möglich (Also an Dateien, die prinzipiell selbstständig ablauffähig sind).

Das folgende Beispiel zeigt das Anhängen des RSX-Programmes RSX22 (einer CPM2 Emulation für bestimmte Anwendungen) an das Programm SUPERZAP.COM.

Soll der RSX-Teil wieder abgehängt werden, so wird lediglich das entsprechende Programm noch einmal über GENCOM aufgerufen, dann aber ohne Namensangabe des RSX-Programmes.

```
A>gencom superzap rsx22
A>gencom superzap
```

Es können auch mehrere RSX-Programme an eine COM-Datei angehängt werden, wie folgendes Beispiel zeigt:

```
A>gencom myfile test1 test2
```

Neben dem Anhängen eines RSX an eine COM-Datei, ist es auch möglich, aus einer RSX-Datei sofort eine COM-Datei zu generieren:

```
A>gencom rsx22 test2[null]
```

In diesem Falle wird eine Datei RSX22.COM generiert, die aus den RSX-Dateien RSX22.RSX und TEST2.RSX bestehen. Wird RSX22.COM danach aufgerufen, 'legt' es sich vor das BDOS und bleibt (je nach Konfigurierung) für alle später aufgerufenen Programme wirksam.

Es bestehen mit GENCOM noch zwei weitere Möglichkeiten Dateien umzu-'bauen', zum Beispiel mit:

```
A)gencom filename[loader]
```

Hier wird vor die Datei <filename> ein sogenannter Header (Kopf) eingebunden, während mit dem folgenden Beispiel:

```
A)gencom filename[scb=(1,5)]
```

nach Aufruf der Datei <filename> zuerst das Byte 1 im SCB auf den Wert 5 geändert wird.

Die Nutzanwendung beider Möglichkeiten soll ebenfalls in Teil III zur Sprache kommen.

****
* GET *
****

Der GET-Befehl.

Syntax:

```
GET (console input from) FILE filname (((ECHO¦NO ECHO)¦SYSTEM))
GET (console input from) CONSOLE
```

Beschreibung:

Der GET-Befehl ist in gewisser Weise verwandt mit dem SUBMIT-Befehl. Hier können Befehlsketten beliebiger Länge aus einer Datei auf Diskette abgerufen werden.

Wenn die Disketteneingabe beendet ist, zur Abarbeitung jedoch noch weitere Daten benötigt werden, so werden diese von der Konsole erwartet.

```
A>get file make
A>myprog
```

In diesem Falle wird die Datei MAKE als Befehlsdatei spezifiziert. Da jedoch die Option SYSTEM nicht angegeben wurde, wird zunächst auf eine weitere Konsolen-Eingabe gewartet. Alle von (im Beispiel) MYPROG.COM benötigten Daten werden nun von der Datei MAKE übernommen.

Nach Beendigung des Programmes MYPROG geht CP/M wieder zum 'normalen Tagesgeschehen' über, d.h. GET ist nicht mehr aktiv. Wird ein Programm jedoch wie folgt aufgerufen:

```
A>get file testit[system]
```

so wird jede weitere Eingabe von der Datei TESTIT übernommen. Es wird nur nach dem Befehl:

```
GET CONSOLE
```

in den Normalbetrieb zurückgekehrt. Dieser Befehl muß in der GET-Datei enthalten sein oder kann (erst) nach dem vollständigen Abarbeiten der GET-Datei über die Konsole eingegeben werden.

Optionen ECHO und NO ECHO geben an, ob alle Befehle aus der GET-Datei heraus auf der Konsole 'geechot' werden oder nicht.

****
* HELP *
****

Syntax:

HELP (Begriff)(Unterbegriff1 Unterbegriff2 ...)([NO] PAGE;PAGE])
HELP [extract|create]

Beschreibung:

Der HELP-Befehl ist als Gedächnisstütze und Hilfe gedacht, wenn der Syntax eines Befehles vergessen wurde.

Leider ist die in HELP von Digital Reseach mitgegebene Information nicht ausreichend und darüberhinaus auch noch in englischer Sprache.

Alle Texte stecken in einer Datei mit der Bezeichnung HELP.HLP und werden ihrerseits von der Datei HELP.COM entsprechend dem Suchbegriff und einem evtl. Unterbegriff ausgesucht und dargestellt.

Die Darstellung erfolgt seitenweise (mit der Option PAGE) oder durchgehend (mit der Option NO PAGE)..

Es besteht jedoch die Möglichkeit die Datei HELP.HLP nach Belieben abzuändern, zu kürzen oder zu ergänzen, (... auch in die deutsche Sprache zu übersetzen ...)

Hierzu sind zwei Optionen wichtig:

EXTRACT (oder einfach 'E') hiermit wird der Header (Kopf) der Datei HELP.DAT entfernt. Diese Datei kann nun mit jedem Editor wie eine normale Textdatei bearbeitet werden.

CREATE (oder einfach 'C') hiermit wird die Textdatei wieder in eine von HELP.COM direkt aufrufbare Datei zurück gewandelt.

Sollen die Möglichkeit der Bearbeitung, Veränderung oder Erweiterung der Datei HELP.DAT genutzt werden, so ist genau auf den vorgeschriebenen Syntax zu achten. Am besten betrachte man sich die HELP.HLP Datei als Beispiel.

In der HELP-Datei muß für jeden Begriff ein 'Kopf' eingeführt werden in der Form:

///nBegriff<CR>

Die Zeichen:

/// melden, daß nun ein neuer Begriff kommt

n symbolisiert den Rang des Begriffes.
N=1 ist immer der Oberbegriff, Zahlen im Bereich von 2..9 symbolisieren immer tiefer gestaffelte Unterbegriffe mit dem Namen 'Begriff'.

Folgende Aufrufe von HELP.COM sind möglich (wie von Digital Reseach geliefert):

```
A>help

A>help date

A>help dir options
```

wobei die

1.Eingabe	eine Reihe von Möglichkeiten im Menü anbietet, die
2.Eingabe	Informationen zum Befehl DATE gibt und die
3.Eingabe	alles zu den Optionen des Befehls DIR ausgibt.

nach jeder Ausgabe meldet sich dieses Programm mit

```
<HELP>
```

als Prompter in der Befehlszeile.

Es kann nun nach weiteren Befehlen, Optionen usw. gefragt werden, ohne daß die Datei HELP jedesmal neu geladen werden muß (sie ist ja schon geladen).

Ein <CR> beendet HELP-Befehl.

****
* HEXCOM *
****

Syntax:

```
HEXCOM (d:)filename
```

Beschreibung:

Mit diesem Hilfsprogramm werden aus Dateien im INTEL-HEX-Format ablauffähige 'COM'-Dateien generiert.

Derartige HEX-Dateien werden z.B. von dem mit CP/M 2 gelieferten Assembler ASM.COM oder dem mit CP/M 3 gelieferten (Macro)-Assembler MAC.COM erzeugt. Die assemblierten Programme müssen selbstverständlich wie unter CP/M üblich für die Startadresse 100H ausgelegt sein.

HEXCOM sucht NUR nach Dateien mit dem Index HEX. Dieser Index braucht nicht eingegeben werden.

Zur Information nachstehend ein Auszug aus einer HEX-Datei, um auch dies einmal darzustellen:

```
:10010000AF3216072178B02CD5602CD44A2211B03E7
:10011000230521EF02C3560ACD5602CD44A2D64135
:10012000FE052160A330F13214A721810BCD56021C
:10013000CD44A2219204FE3038F3FE3430EFE6B3E2
:10014000-60C3216072182040CD56023A14A7C64107
 ^^ ^  ^ ^                                 ^
 ;; ;  ; ;                                 ;_ Pruefsumme
 ;; ;  ; ;_ 16 (10H) Bytes Daten in ASCII-Format
 ;; ;  ;_ Kennzeichen (00) fuer 'normale' HEX-Datei
 ;; ;_ Startadresse des ersten Bytes in der Zeile (LOW-Byte-HIGH-Byte Format
 ;;_ Anzahl der Datenbytes in jeder Zeile
 ;_ Startmarke einer neuen Datenzeile

:00000001FF     Dies ist eine Abschlusszeile mit Angabe der Startadresse
                (100H in LOW-Byte HIGH-Byte Format) und der Pruefsumme.
                Die Pruefsumme ist nichts anderes als die fortlaufende
                Subtraktion (von 00 beginnend) aller Bytes (ausser dem
                Doppelpunkt) in einer Zeile. Das empfangende Programm
                addiert zu dieser Pruefsumme alle empfangenen Bytes. Das
                Ergebnis muss 00 sein, wenn der Empfang fehlerfrei war.
```

Alle HEX-Dateien beginnen mit einem Doppelpunkt <:> danach folgt eine Angabe (in Hexadezimal) über die Anzahl der Datenbytes in einer Zeile (10H=16dezimal).

Nach dieser Angabe folgt die Adresse, wohin das erste Byte einer Zeile geschrieben werden soll (ebenfalls in Hexadezimal).

Es folgt ein Kennbyte: 00=Normale Daten, 01=Sonderdatei o.Ä.

Nun folgen soviele Datenbytes wie in Position 2 angegeben wurden.

Als letztes wird ein Byte ausgegeben, das die Summe aller gesandten Zeichen in dieser Zeile (ohne dem Doppelpunkt) enthält.

Ein evtl. 'Überlauf' wird dabei unterschlagen. Diese Prüfsumme (Checksum) ist ebenfalls in Hexadezimal.

Alle Daten in einer HEX-Datei werden im ASCII-Format übergeben, da diese Dateienart ursprünglich vor allem für den Datenverkehr zwischen verschiedenen Computern gedacht war und die entsprechenden Computerschnittstellen nur im (7-Bit) ASCII-Format senden und empfangen konnten. Anstelle von handshake-Signalen wurde Textanfang und Textende durch entsprechende CNTRL-Zeichen erkannt. Das Textende in Dateien wird auch heute noch von den meisten Editoren mit 1AH (EOT=end of text=Textende) markiert.

HEX-Dateien können mit TYPE-Befehl auf der Konsole ausgegeben werden.

****
* INITDIR *
****

Der INITDIR-Befehl

Syntax:

```
INITDIR d:
```

Beschreibung:

Mit INITDIR kann das Inhaltsverzeichnis einer Diskette darauf vorbereitet werden, Datumsmarken und einen Diskettennamen aufzunehmen.

Dies geschieht wohlgemerkt NICHT mit INITDIR selbst, sondern mit dem Hilfsprogramm SET.COM. INITDIR hat lediglich eine ähnlich Aufgabe wie die Formatierung einer Diskette, nämlich die Formatierung des Platzes für das CP/M 3 typische, erweiterte Inhaltsverzeichnis vorzunehmen.

Bei INITDIR wird folgender Dialog ausgegeben:

```
A>initdir b:

INITDIR WILL ACTIVATE TIME STAMPS FOR SPECIFIED DRIVE.
Do you want to re-format the directory on drive: B  (Y/N)?  y

Do you want the existing date/time stamps cleared (Y/N) y
```

Übersetzung:

INITDIR AKTIVIERT DIE ZEITMARKEN FÜR DAS ANGEGEBENE LAUFWERK
Soll das Inhaltsverzeichnis in Laufwerk B um-formatiert werden (Ja/Nein)

hat die Diskette bereits ein erweitertes Inhaltsverzeichnis, so wird abgefragt:

Sollen die existierenden Zeitmarken gelöscht werden (Ja/Nein)?

An dieser Stelle soll der Vollständigkeit halber die Auswirkung des erweiterten Inhaltsverzeichnises erwähnt werden.

Jeweils für drei Einträge im Inhaltsverzeichnis wird ein zusätzlicher Eintrag reserviert. In diesem Eintrag werden nun alle entsprechenden Daten wie Zugriffsdatum und Uhrzeit, aber auch ein mögliches Paßwort (natürlich verschlüsselt) vermerkt.

Um keine Probleme mit dieser Art des Eintrages unter CP/M 2 zu haben, ist das erste Byte des zusätzlichen Eintrages ein 21H, während ein (möglicher) Diskettenname (ein Disketten-Label) durch ein vorangestelltes 20H erkennbar ist (näheres hierzu in TEIL III und TEIL V). CP/M 2 interpretiert übrigens DIR-Einträge mit vorangestelltem 20H oder 21H als 'leer'-Datei.

Es ist dabei besonders wichtig zu wissen, daß die Anzahl der effektiven Einträge in das Inhaltsverzeichnis um ein Viertel (und dem DIR-Label) der möglichen Einträge reduziert werden, wenn ein

erweitertes Inhaltsverzeichnis benutzt wird.
Da bei den heutigen Diskettenformaten mit doppelter Dichte und auf Harddisks genügend Speicherraum zur Verfügung steht, um das Inhaltsverzeichnis ohne Probleme auf 1024 Einträge zu erweitern (dies entspricht einem belegten Speicherplatz von 32k), dürfte dies kein Problem darstellen.

****
* PATCH *
****

Syntax:

```
PATCH filename.{ext} (n)
```

Beschreibung:

Dieser Befehl dient eigentlich nur der 'Buchhaltung', ob Fehlermeldungen, die Digital Research von Zeit zu Zeit veröffentlicht, bereits korrigiert wurden. Soll z.B. festgestellt werden, ob in dem Hilfsprogramm SHOW.COM ein 'patch' und wenn ja welcher, installiert wurde, so wird dies einfach wie folgt abgefragt:

```
A>patch show

Current patch for SHOW.COM
  None

A>
```

Wurde ein neuer 'patch' installiert, so kann dieser wie folgt übertragen werden:

```
A>patch show 1

Do you want to indicate that Patch # 1
has been installed for SHOW.COM? y

Patch installed

A>
```

Patch wird 'nur' mit maximal 32 Fehlermeldungen fertig.

Anmerkung:

Die eigentliche Korrektur wird dabei nicht von PATCH.COM übernommen, sondern muß nach Anweisung von DIGITAL RESEARCH oder dem zuständigen Distributor vorgenommen werden, zumeist mit Hilfe von SID.COM.

PATCH ist nur für COM-Dateien verwendbar.

****
* PUT *
****

Der PUT-Befehl

Syntax:

PUT CONSOLE!PRINTER (output to) CONSOLE!PRINTER!FILE filename([o])

Beschreibung:

Mit dem PUT-Befehl ist es möglich,alle Ausgaben, die normalerweise auf dem Bildschirm erscheinen, auf eine Datei oder den Drucker umzulenken.

Dieser Befehl ist vor allem für Protokolle geeignet z.B. bei längeren Assemblier- und Linkaufgaben, bei welchen festgehalten werden muß, wie das generierte Programm zusammengebunden wurde.

Es können zwei Quellen spezifiziert werden:

a) die Konsole oder allgemein CONOUT

b) der Drucker oder allgemein LST

Als Ziel sind 3 Einheiten möglich:

a) Eine Datei (spezifiziert als <filename>)

b) Der Drucker (wenn Ausgabe über CONOUT erfolgt)

c) Die Konsole (wenn Ausgabe über LST erfolgt)

Dazu stehen 5 Optionen zur Verfügung:

a) ECHO oder NO ECHO steuert die zusätzliche Wiedergabe auf dem Bildschirm.

b) FILTER oder NO FILTER. Wird FILTER angegeben, werden alle Steuerzeichen mit einem vorangestellten '^' ausgegeben, bei NO FILTER werden sie ausgegeben wie sie sind.

c) SYSTEM veranlaßt, daß ALLE folgenden Daten (auch Echo) an das angegebene Ziel ausgegeben wird.

Nach der Option SYSTEM kann nur mit einer Rücküberweisung in den Normalbetrieb zurückgeschaltet werden, wie nachfolgendes Beispiel zeigt:

```
A>put console output to file holdit[echo,system]


A>put console output to console
```

Nach Eingabe der 1. Zeile, werden alle weiteren Ausgaben, die der gewünschte Programmablauf mit sich bringt (wegen ECHO), auf dem Bildschirm dargestellt. Wird jedoch die Option NO ECHO gewählt, so muß der Befehl zum Zurückschalten 'blind' eingegeben werden.

****
* SAVE *
****

Syntax:

```
SAVE
```

Beschreibung:

Unter CP/M 2 war der SAVE-Befehl noch im CCP eingebaut. Da unter CP/M 3 der CCP ein Programm im üblichen Sinne ist, wird der Bereich ab Adresse 100H bei jedem Warmstart vom CCP überladen, bevor eine dort befindliche Datei 'gerettet' werden kann.

Die SAVE-Funktion wird normalerweise nur bei Fehlerkorrekturen von Dateien benutzt; sie hat also einen recht eingeschränkten Zweck. Normalerweise wird unter CP/M 3 mit dem mitgelieferten Programm SID.COM gearbeitet, das eine SAVE-Funktion enthält. SID hat für Besitzer eines Z80-Computers jedoch einen entscheidenden Nachteil - er 'versteht' nur die INTEL (8080) Mnemonics.

Benutzer, die mit ZSID.COM oder einem anderen CP/M 2 kompatiblen Debugger arbeiten, können Dateien zwar korrigieren, entwanzen oder was immer, aber danach nicht auf Diskette zurückschreiben. Mit dem SAVE-Befehl wurde jedoch ein entsprechendes Hilfsmittel zur Verfügung gestellt.

Das folgende Beispiel zeigt seine Einsatzmöglichkeiten:

```
A>save
A>zsid myprog.com

        ;
        ; jetzt koennen Aenderungen vorgenommen werden
        ; ... dann zurueck ins Betriebssystem mit ..
        ;

#g0
SAVE Ver 3.0
File (or RETURN to exit)?myprog1.com
Delete myprog1.com? y
From?100
To?500

A>
```

Der SAVE-Befehl kann auch direkt aus der Befehlszeile heraus verwendet werden, (durch zweimaliges Aufrufen) In diesem Falle muß jedoch berücksichtigt werden, daß zumindest die unteren 4K (also bis Adresse 1100H) ungültig, d.h. überschrieben sind. Es kann jedoch jede beliebige höhere Startadresse (in HEX!!!) angegeben werden, natürlich auch der Datenblock ab Adresse 0, falls dortige Daten in gewissen (Test) Fällen von Bedeutung sind, diese Daten können dann mit z.B. DUMP betrachtet werden.

Anmerkung: das Public-Domaine Programm DDTZ entspricht etwa ZSID kann jedoch auch Dateien 'saven'.

****
* SET *
****

Der SET-Befehl

Syntax:

SET (d:) (filename)[optionen]

Beschreibung:

Mit dem SET-Befehl steht eine reiche Auswahl von Möglichkeiten der Disketten- und Dateienorganisation und des Disketten- und Dateienschutz zur Verfügung. Alle Optionen sind in Tabelle 8 zusammengefaßt.

1. Setzen von Datei-Attributen.

Mit diesen Optionen ist es möglich Dateien als DIR- oder SYS-Datei zu erklären. Weiterhin kann eine Datei als RO- (read-only - nur lesen) Datei gesetzt werden. Ein unbeabsichtigtes Löschen einer derartigen Datei ist also recht wirkungsvoll verhindert.

Neben einem Archiv-Attribut, können noch 4 weitere Attribute gesetzt werden. Das Archiv-Attribut ermöglicht es mit dem Programm PIP.COM nur solche Dateien zu kopieren, die dieses Attribut noch nicht gesetzt haben. Die Attribute F1..F4 können persönlichen Zwecken dienen (siehe hierzu auch die Optionen bei DIR). Die letztgenannten Optionen sind vor allem für Großsysteme von Interresse, wo viele Dateien gleichzeitig auf einer Harddisk vorhanden sind und die Übersicht sonst leicht verloren gehen kann.

```
A>set *.com[sys]

A:FORMAT  .COM  set to system (SYS), Read Write (RW)
A:CCP     .COM  set to system (SYS), Read Write (RW)
A:WM      .COM  set to system (SYS), Read Write (RW)
A:SID     .COM  set to system (SYS), Read Write (RW)
A:DATE    .COM  set to system (SYS), Read Write (RW)
A:SETDEF  .COM  set to system (SYS), Read Write (RW)
A:INITDIR .COM  set to system (SYS), Read Write (RW)
A:TYPE    .COM  set to system (SYS), Read Write (RW)
A:DIR     .COM  set to system (SYS), Read Write (RW)
A:GENCPM  .COM  set to system (SYS), Read Write (RW)
A:SET     .COM  set to system (SYS), Read Write (RW)
A:ZSID    .COM  set to system (SYS), Read Write (RW)

A>
```

Alle folgenden Optionen bedürfen der 'vorherigen Behandlung' der Diskette mit INITDIR.

Es soll auch an dieser Stelle nicht unerwähnt bleiben, daß die Anzahl der möglichen Einträge in das Inhaltsverzeichnis durch die Option INITDIR um ein Viertel verringert wird.

Dateien die mit INITDIR und den nachfolgenden Optionen behandelt wurden, werden im Inhaltsverzeichnis mit Uhrzeit und Datum abgelegt, Angaben darüber sind mit dem DIR-Befehl zu erhalten.

2. Zuweisung eines Namens für eine Diskette

Dieser Namen kann aus dem Inhaltsverzeichnis heraus (durch eine vorangestellte 20H) erkannt werden. Er dient hauptsächlich Ordnungszwecken.

```
A>set b:[name=texte.001]

Label for drive B:

Directory      Passwds  Stamp   Stamp   Stamp
Label          Reqd     Create  Access  Update
-------------  -------  ------  ------  ------
B:TEXTE   .001 off      off     off     off
```

3. Zuweisung von Zeitmarken für eine Diskette

Zeitmarken sind sinnvoll, um aus dem Inhaltsverzeichnis den Termin des letzten Zugriffes auf eine Datei, seine Veränderung oder Entstehung erkennen zu können.

Die beiden Optionen CREATE (Entstehung) und ACCESS (Zugriff) können wahlweise, jedoch nicht gemeinsam verwendet werden.

CREATE ist sinnvoll für Programmierer, um zu wissen wann eine Datei entstanden ist, während ACCESS sinnvoll ist, wenn nachgehalten werden muß, wann auf eine Datei zuletzt zugegriffen wurde.

Ob und welche Option sinnvoll ist muß, der Benutzer selbst entscheiden.

Als zusätzliche weiter Option steht UPDATE (Erneuerung) zur Verfügung.

```
A>set b:[create=on,update=on]

Label for drive B:

Directory      Passwds  Stamp   Stamp   Stamp
Label          Reqd     Create  Access  Update
-------------  -------  ------  ------  ------
B:TEXTE   .001 off      on      off     on
```

Ein Hinweis an dieser Stelle:

Textverarbeitungsprogramme legen normalerweise eine BAK Datei (quasi als Rücklage für den Fehlerfall) an. Hier liegt eine Fehlerquelle besonderer Art. Im Inhaltsverzeichnis dürfen (laut Vereinbarung) Dateien keine gleichen Dateinamen und gleichen Index haben. Im obigen Falle wird dies damit umgangen, daß (in den Textverarbeitungsprogrammen) die vorhandene Datei vor dem Ablegen der neu bearbeiteten Datei in <datei>.BAK umbenannt wird. Erst dann wird eine neue Datei (mit dem alten Namen) angelegt. CP/M interpretiert dies jedoch IMMER als das, was es ist, das Ablegen einer neuen Datei - mit einer neuen Zeitmarke. Da aber die BAK-Datei ebenfalls neu angelegt wird (eben durch die Umbenennung), wird sie folgerichtig ebenfalls mit einer neuen Zeitmarke versehen.

4. Zuweisung eines Paßwortes (Kennwort)

Mit einem Paßwort kann eine Diskette vor 'fremdem' Zugriff in verschiedenen Ebenen geschützt werden. Um einer Diskette oder einer Datei den Schutz durch ein Paßwort zu geben, sind verschiedene Vorgänge notwendig:

das Paßwort muß mit der Option PASSWORD zugewiesen werden.

Die Ebene des Schutzes (READ-WRITE-DELETE) muß definiert werden.

Der Schutz muß freigegeben werden.

Dieser sicherlich etwas umständliche Vorgang ist nachfolgend wieder an Hand einiger Beispiele dargestellt. Die Option DEFAULT erlaubt dem Kenner des benutzten Paßwortes einen gewissermaßen unbeschränkten Zugriff auf eine Diskette, solange diese nicht gewechselt oder das System neu gebootet wird.

```
B>set[password=geheim]

Label for drive B:

Directory      Passwds  Stamp    Stamp    Stamp
Label          Reqd     Create   Access   Update
-------------  -------  -------  -------  -------
B:TEXTE   .001   off      on       off      on

Password = GEHEIM          < GEHEIM ist das neue Passwort

B>set[default=geheim]

Default password = GEHEIM  < wurde als Global gesetzt


B>set[protect=on]

Label for drive B:

Directory      Passwds  Stamp    Stamp    Stamp
Label          Reqd     Create   Access   Update
-------------  -------  -------  -------  -------
B:TEXTE   .001   on       on       off      on
                 :
                 !_ der Passwortschutz ist eingeschaltet

B>set[protect=off]

Label for drive B:

Directory      Passwds  Stamp    Stamp    Stamp
Label          Reqd     Create   Access   Update
-------------  -------  -------  -------  -------
B:TEXTE   .001   off      on       off      on
                 :
                 !_ der Passwortschutz ist ohne Nachfrage ausgeschaltet
```

```
B>set [default=none]

Default password = NONE  < DEFAULT wird ausgeschaltet ...

B>set [protect=on]       < und der Passwortschutzt wieder eingeschaltet


Directory Label
Password ? geheim        <dazu wird bereits das (frueher zugewiesene)
                         <Passwort benoetigt. Die Eingabe erfolgt OHNE ECHO!

Label for drive B:

Directory       Passwds  Stamp    Stamp    Stamp
Label           Reqd     Create   Access   Update
--------------  -------  -------  -------  -------
B:TEXTE   .001    on       on       off      on
                  |
                  |_ der Passwortschutz ist wieder eingeschaltet

B>set[protect=off]       < und soll nun wieder abgeschaltet werden


Directory Label          < dieses mal wird abgefragt
Password ? hugo          < ein falsches Passwort ('blind' eingegeben)
ERROR: Wrong Password    < wird auch gemeldet
Directory Label
B>set[protect=off]       < nochmal


Directory Label
Password ? geheim        < mit dem richtigen Password den Schutz aufheben

Label for drive B:

Directory       Passwds  Stamp    Stamp    Stamp
Label           Reqd     Create   Access   Update
--------------  -------  -------  -------  -------
B:TEXTE   .001    off      off      off      off
                  |
                  |_ dieses mal ist der Schutz aufgehoben worden

B>set [password=none]


Directory Label
Password ? geheim        < auch zum Zuruecknehmen eines Passwortes
                         < wird es, einmal gesetzt, nocheinmal benoetigt
Label for drive B:

Directory       Passwds  Stamp    Stamp    Stamp
Label           Reqd     Create   Access   Update
--------------  -------  -------  -------  -------
B:TEXTE   .001    off      off      off      off

Password = NONE          < NONE bedeutet 'kein Password' ...
```

WICHTIG:

Paßwort nicht vergessen - sonst kann man Datei oder Diskette 'vergessen', nur noch sehr intime Kentnisse und geeignete Hilfsprogramme können dann noch helfen. Es gibt hier allerdings einen recht 'hinterhältigen' Trick, falls ein CP/M 2 System zur Verfügung steht.

CP/M 2 kennt keine Paßworte, also ist es möglich, alle Dateien unter CP/M 2 auf eine frisch formatierte Diskette umzukopieren. Der Datenschutz ist dann dahin, alle Datums- und Zeitmarken allerdings auch.

Das Abschalten einer Option hat dieselbe Syntax wie das Einschalten einer Option, nach dem Zuweisungszeichen '=' folgt jedoch lediglich ein RETURN (<cr>) oder im Falle von Namen, die Zuweisung 'NONE' (=Keiner).

DIR	Setzt das DIR-Attribut. Alle Dateien mit diesem Attribut werden von DIR erfasst.
SYS	Setzt das SYS-Attribut. Alle Datein mit diesem Attribut koennen nur von DIRS erfasst werden. Auf COM-Dateien mit dem SYS-Attribut kann aus allen Benutzer(USER)-Ebenen zugegriffen werden.
RO	Setzt das R(ead)-O(nly) Attribut. Auf Dateien mit diesem Attribut kann nur zum Lesen zugegriffen werden. Diese Dateien koennen nicht ueberschrieben werden.
RW	Setzt das R(ead)-W(rite) Attribut. Dateien mit diesem Attribut koennen gelesen und beschrieben werden.
ARCHIVE=ON!OFF	Setzt das Archiv-Attribut (ON) oder setzt es zurueck (OFF). Ist dieses Bit OFF, bedeutet dies, es wurde noch keine Sicherungskopie der enstprechenden Datei gemacht. Dies ist z.B. bei dem Kopierprogramm PIP.COM von Bedeutung wenn Sicherungskopien mit JOKERN gemacht werden. PIP setzt das Archiv-Attibut auf ON, wenn es eine Datei kopiert hat. Die Option OFF muss vom Benutzer gesetzt werden!
n=ON!OFF	Die vom Benutzer definierbaren Attribute (1-4) koennen mit dieser Option gesetzt (ON) oder zurueckgesetzt werden. Diese Optionen sind z.Z. nur fuer den eigenen Gebrauch von Interesse.
NAME=nnn.xxx	Zuweisung des Label nnn.xxx an eine Diskette
PASSWORD=password	Zuweisung eines Passwortes (Kennwort) an eine Diskette.
PROTECT=ON!OFF	Passwortschutz einschalten (ON) oder ausschalten. (OFF) Im eingeschalteten Zustand wird zum Abschalten das Passwort benoetigt.
=READ	Datei kann ohne Passwort nicht gelesen werden.
=WRITE	Datei kann ohne Password nicht beschrieben werden.
=DELETE	Datei kann ohne Passwort nicht geloescht werden
=NONE	Passwortschutz wird zurueckgenommen wenn das bisheriges Passwort bekannt ist.

Tabelle 8 Optionen des SET-Attributes

DEFAULT=password	Hiermit kann der berechtigte Benutzer sein Passwort dem Computer mitteilen. Bis zum Diskettenwechsel oder Abschalten des Systemes wird anstelle der Passwortabfrage bei jedem Zugriff nur auf das DEFAULT-Password zugegriffen. Dieses Passwort muss natuerlich dem zuvor angegebenen Passwort entsprechen.
CREATE=ON!OFF	Alle Dateien erhalten bei der Erstgenerierung eine Zeit- und Datumsmarke. (Kann nicht mit ACCESS zusammen definiert werden!) ON=EIN
ACCESS=ON!OFF	Alle Dateien erhalten bei jedem Zugriff eine neue Zeit- und Datumsmarke. (Nicht zusammen mit CREATE definierbar) ON=EIN OFF=AUS
UPDATE=ON!OFF	Alle Dateien erhalten nach jeder Aenderung eine neue Zeit- und Datumsmarke. ON=EIN

noch **Tabelle 8 Optionen des SET-Attributes**

****
* SETDEF *
****

Der SETDEF-Befehl

Syntax:

```
SETDEF [d:[,d:[,d:[,d:]]]][[TEMPORARY=d:]][ORDER=(typ[,typ])]]
SETDEF [DISPLAY[NO] DISPLAY][PAGE[NO] PAGE]
```

Beschreibung:

Mit dem SETDEF-Befehl können bestimmte Vorgabewerte vorbestimmt werden.

Die nachfolgende Beispiele sollen die Anwendung verdeutlichen.

Eine der wichtigsten Optionen für ein reibungsloses Arbeiten unter CP/M 3 ist sicher das Einstellen der Suchfolge auf den vorhandenen Laufwerken

```
A>setdef a:,*,e:
```

Dies bedeutet: eine Datei, gleichgültig auf welchem Laufwerk aufgerufen, wird zuerst auf Laufwerk A: gesucht, danach auf dem Laufwerk, das im Prompter (X>) gezeigt wird. Wird sie auch da nicht gefunden, wird in Laufwerk E: nachgesehen, dies könnte z.B. eine Harddisk sein.

Welche Suchordnung eingestellt wird, bleibt dem Benutzer überlassen. Es muß lediglich die gewünschte Such-Reihenfolge (bis zu drei Laufwerke) eingeben.

Nach jedem eingegebenen Laufwerksnamen folgt ein Doppelpunkt. Soll innerhalb der gewünschten Reihenfolge auf das momentane Laufwerk zugegriffen werden so wird dieses mit einem Stern <*> bezeichnet, ohne folgendem Doppelpunkt. Unter 'momentan' versteht man in diesem Zusammenhang das Laufwerk, von welchem aus ein zu suchendes Programm aufgerufen wurde.

Manche Programme legen während des Abarbeitens einer vorgegebenen Aufgabe eine oder mehrere Zwischendateien an, die anschließend wieder aus dem Inhaltsverzeichnis gelöscht werden. Diese Dateien benötigen Platz, der u.U. auf einer vorgegebenen Diskette 'eng' werden kann. Dem kann mit folgendem Beispiel abgeholfen werden:

```
A>setdef [temporaty=c:]
```

In diesem Falle werden alle Zwischendateien in Laufwerk C: abgelegt.

Eine weitere wesentliche Vereinfachung bei der täglichen Arbeit ist die mögliche Zuweisung der Suchordnung von COM- und SUB-Dateien.

Der wirkliche Effekt dabei es weniger, daß je nach Wunsch zuerst nach einer SUB- oder zuerst nach einer COM-Datei gesucht wird, sondern vielmehr die nicht ausgesprochene Tatsache, daß SUB(mit)-Dateien wie COM-Dateien einfach beim Namen aufgerufen werden können und nicht, wie noch beim 'alten' CP/M 2 mit vorangestelltem SUBMIT.

Die Einstellung erfolgt wie folgt:

```
A)setdef[order=(com,sub)]
```

Es wird danach immer zuerst eine COM-Datei mit dem aufgerufenen Namen gesucht (und zwar auf allen in der Suchordnung eingebundenen Laufwerken). Erst wenn eine solche nicht gefunden werden kann, wird nach einer SUB-Datei gleichen Namens gesucht.

Selbstverständlich kann die Suchreihenfolge auch umgekehrt sein. Dies bringt Vorteile mit sich, wenn z.B. vor dem eigentlichen Aufruf eines Programmes (der COM-Datei) erst noch verschiedene Optionen gesetzt werden sollen. Um den Benutzer nicht zu verwirren, erhält also eine SUBMIT-Datei den 'gewohnten' Dateinamen. Die SUBMIT-Datei kann nun zuerst seinerseits ein Hilfsprogramm aufrufen, welches z.B. verschiedene Optionen im Menü abfragt und erst dann die entsprechende COM-Datei; entweder auf einem anderen Laufwerk oder passend umbenannt.

Zwei Optionen ermöglichen es die bei der Systemgenerierung vorgegebene Optionen zu umgehen:

```
A)setdef[page]

A)setdef[display]
```

Im ersten Fall wird z.B. bei TYPE oder DIR nicht mehr am Bildschirmende angehalten, sondern der Text läuft bis zum Textende durch. Angehalten werden kann nur noch mit CNTRL-S, weiter gehts mit CNTRL-Q.

Im zweiten Falle werden vor jeder Programmausführung noch die sogenannten Systemparameter (falls vorhanden) mit auf dem Bildschirm ausgegeben. Darunter sind Angaben wie gültige Laufwerksnummern, Dateiname und Dateityp sowie die eingeschaltete USER-Nummer zu verstehen.

Beide Optionen können mit einem vorangestellten 'NO' wieder abgeschaltet werden.

****
* SHOW *
****

Der SHOW-Befehl

Syntax:

```
SHOW (d:)([SPACE|LABEL|USERS|DIR|DRI(ve)])
```

Beschreibung:

SHOW hat unter CP/M 3 etwa die gleiche Funktion wie STAT.COM unter CP/M 2.

Mit dem SHOW-Befehl und seinen Optionen kann der Benutzer Informationen über Restspeichergröße der Disketten, Laufwerkscharakteristiken, Diskettenname und die restliche Kapazität des Inhaltsverzeichnis erhalten.

Die folgenden Beispiele zeigen die Anwendung und das Ergebnis:

```
A>show                          < ohne weitere Angaben ...

A: RW, Space:      252k         < zeigt Restkapazitaet aller
B: RW, Space:      930k         < eingeloggten Laufwerke


A>show b: [label]               < Frage nach dem Label (Namen)

Label for drive B:              < wird mit allen Einzelheiten beantwortet

Directory     Passwds  Stamp  Stamp
Label         Reqd     Create Update  Label Created   Label Updated
-----------   -------  -----  ------  -------------   -------------
TEXTE   .0v1    off     on     on      06/24/85 09:19  12/28/85 13:20


A>show [users]                  < die Frage nach den (benutzten) USER-Ebenen

A: Active User :    0           < 0=keine USER-Ebene benutzt
A: Active Files:    0           < ... dann natuerlich auch keine Dateien
A: # of files  :   13           < dies sind die Dateien in USER-0 oder SYS

A: Number of time/date directory entries:   64   < belegt von Zeitmarken
A: Number of free directory entries:       167   < freie Eintrage


A>show [dir]                    < hier wird nur nach den Eintraegen gefragt


A: Number of time/date directory entries:   64
A: Number of free directory entries:       167
```

```
A>show [drive]              < hier wird nach der Diskettenspezifikation
                            < getragt ...

     A: Drive Characteristics       < Charakteristik fuer Laufwerk A:
 9,600: 128 Byte Record Capacity    < Anzahl der (logichen) Sektoren
 1,200: Kilobyte Drive  Capacity    < ges.Laufwerkskapazitaet (in K)
   256: 32 Byte  Directory Entries  < Anzahl der DIR-Eintraege
   256: Checked  Directory Entries  < soviele werden gepruefT
   128: Records / Directory Entry   < (log) Sektoren pro Eintrag
    16: Records / Block             < (log) Sektoren pro Block
   128: Sectors / Track             < (log) Sektoren pro Track (Zylinder)
     2: Reserved Tracks             < reservierte Tracks
 1,024: Bytes / Physical Record     < (phys) Sektorgroesse
```

Anmerkung:

Unter logischen Sektoren versteht man die kleinste physikalisch mögliche Sektorgrösse von 128 Bytes. Diese Grössenangabe rührt sicherlich auch heute noch aus den 'alten' CP/M-Zeiten her, als nur diese Größe vom BDOS verarbeitet werden konnte. Heute kann das BDOS Sektorgrößen bis 1024 Bytes (1k) selbst verwalten.

Unter Eintrag versteht man die Anzahl von Blöcken, die in einem DIR-Eintrag verwaltet werden können. Große Dateien benötigen demzufolge mehrere DIR-Einträge, obwohl jeder Eintrag im Inhaltsverzeichnis nur einmal erscheint.

Dies klärt auch den scheinbaren Widerspruch in obigem Beispiel, daß (unter USERS zu erkennen) nur 13 Dateien vorhanden sind, aber diese 25 DIR-Einträge belegen.

(Zum Nachrechnen: 256-(64+167)=25)

Unter Block versteht man die Mindestgröße eines einzelnen DIR-Eintrages. Er ist bei Standard-CP/M-Disketten oder bei Disketten mit weniger als 256k Speicherplatz üblicherweise 1k groß. Bei allen anderen Disketten ist die Blockgröße meist 2k (16*128Byte im Beispiel).

Das wichtigste Kriterium bei der Blockgröße (sie wird im BIOS bestimmt!) ist es, daß ihre Größe multipliziert mit der Anzahl der DIR-Einträge die maximal mögliche, belegbare Speichergröße einer Diskette bestimmt. Eine Harddisk mit 64MB Speicherkapazität nützt nichts, wenn dafür nur 128 DIR-Einträge bei 2k Blockgröße vorgesehen werden. (Dies würde maximal 4MB und minimal 256k möglicher Belegung zulassen - danach würde das System lakonisch DIRECTORY FULL melden)

****
* SUBMIT *
****

Der SUBMIT-Befehl

Syntax:

SUBMIT (filename) (argument1) ... (argument10)

Beschreibung:

SUBMIT gehört eigentlich mehr zu den in Kapitel 5 besprochenen Programmen und Werkzeugen. Unter CP/M 3 ist es (siehe hierzu SETDEF) jedoch möglich, eine SUB(mit) Datei durch einfachen Aufruf wie einen Befehl ablaufen lassen zu können.

Es kommt bei der Programmentwicklung recht häufig vor, daß bis zum endgültigen Ergebnis viele Einzelbefehle an den CCP gegeben werden müssen. Teilweise sind diese Befehle darüberhinaus lang und kompliziert, dazu hin muß der Programmierer 'am Gerät' bleiben, auch wenn manche Sequenzen lange dauern.

Hier greift der SUBMIT-Befehl ein. Mit einem Editor werden alle Befehle in einzelne Zeilen geschrieben und dann nur noch diese Datei aufgerufen. Diese Datei MUSS den Index SUB haben, damit SUBMIT sie erkennt.

Es können auch Parameter in der Befehlszeile übergeben werden. Es sind bis zu zehn Parameter zuläßig, jedem Parameter ist ein '$' gefolgt von einer Zahl 1..10 voranzustellen.

Werden Programme wie PIP oder SUB aufgerufen, die einen gewissen Dialog verlangen, so können die dort erwarteten Eingaben mit einem vorangestellten '<' bereits in der SUB-Datei eingegeben werden. Ein '<' ohne nachfolgendem Befehl entspricht einem RETURN (<cr>)..

Empfängt SUBMIT noch Daten, nachdem dies eigentlich nicht mehr erwartet wird, wird die Warnung

Warning: Program input ignored

ausgegeben, bevor das Programm abgebrochen wird oder ein weiterer SUBMIT-Lauf erfolgt. Fehlen dagegen Daten, die von der SUB-Datei nicht mehr geliefert werden, wird auf eine Eingabe über die Tastatur gewartet.

Die SUB(mit)-Datei PROFILE.SUB ist in diesem Zusammenhang von ganz besonderer Bedeutung. Nach jedem Kaltstart (Boot) prüft der CCP nach, ob auf der Boot-Diskette diese Datei vorhanden ist. Wenn dies der Fall ist, wird sie automatisch über SUBMIT aufgerufen. Mit dieser Datei können nun nach jedem Kaltstart beliebige Systemparameter gesetzt werden und, falls dies gewünscht wird, am Ende noch der automatische Start eines beliebigen Programmes veranlaßt werden.

Ein SUBMIT-Lauf bedeutet immer die Ablage eines Zwischenprogrammes auf der jeweiligen Diskette. Aus diesem Grund darf die Diskette nicht schreibgeschützt sein. Im Falle von PROFILE.SUB muß natürlich auch die Datei SUBMIT.COM auf der Boot-Diskette sein. Ist eines von Beidem nicht der Fall, wird PROFILE.SUB schlichtweg ignoriert, das System bootet ohne der gewünschten

Voreinstellung von Optionen --- und das ohne jede Warnung !!!!!!

Eine übliche PROFILE-Datei zeigt das folgende Beispiel, in welchem zuerst die Suchordnung der Laufwerke und Programme vorgegeben und danach die Eingabe des Datums und der Uhrzeit erzwungen wird.

```
SETDEF *,A:[ORDER=(COM,SUB)]
DATE SET
```

Ein kleines Spiel-Beispiel zeigt noch die Anwendung der Parameterübergabe in einer SUBMIT-Datei. Im Beispiel werden die Quellbeginn- und Endadresse sowie die Zieladresse eines move-Befehles an SID übergeben.

```
sid
<f100,120,55
<f8000,801f,AA
<d100,11f
<s8000
<"dies ist ein test"
<.
<m$1,$2,$3
<d100,11f
<g0


A>test 8000 801f 100

CP/M 3 SID - Version 3.0
#f100,120,55
#f8000,801f,AA
#d100,11f
0100: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUUUUUUUUUU
0110: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 UUUUUUUUUUUUUUUU
#s8000
8000 AA "dies ist ein test"
8012 AA .
#m8000,801F,100
#d100,11f
0100: 64 69 65 73 20 69 73 74 20 65 69 6E 20 74 65 73 dies ist ein tes
0110: 74 22 AA AA AA AA AA AA AA AA AA AA AA AA AA AA t"..............
#g0

A>
```

Kapitel 5 Werkzeuge und Programme von CP/M 3

Digital Reseach hat seinem CP/M 3 eine ganze Reihe hervorragende Werkzeuge (Utilities) und Programme beigefügt. Sie sind fast ausschließlich für den erfahrenen Programmierer gedacht, und sollen es diesem ermöglichen, sich Hardware-Anpassungen an CP/M 3 nach eigenen Wünschen zu schreiben und diese Programmteile zu einem maßgeschneiderten System zusammenzubinden.

Mit diesen Werkzeugen können selbstverständlich auch eigenständige Programme entwickelt und 'entwanzt' werden.

Leider bietet Digital Research unter CP/M 3 keinen Debugger mit Z80 Mnemonics, aber der 'alte' ZSID ist ohne weiteres funktionsfähig, geeigneter ist vielleicht noch das Public-Domaine Programm DDTZ, da es eine SAVE-Funktion zur Verfügung stellt, die ZSID nicht hat.

Auch die an sich recht guten Assembler sind voll auf die INTEL-(8080)-Mnemonics ausgelegt. Es können damit zwar mit Hilfe einer MACRO-Bibliothek, die mitgeliefert wird, auch Z80-Programme geschrieben werden, aber eben nicht mit Zilog(Z80)-Mnemonics sondern einem Kauderwelsch von 8080 und Z80. Der Z80-Benutzer wird daher nicht umhin können, einen guten Z80-Assembler zusätzlich zu erwerben.

Ein weiteres 'Problem' ist der von Digital Research mitgelieferte Editor ED.COM. Es handelt sich hierbei um einen zeilenorientierten Editor, d.h. es kann immer nur innerhalb einer Zeile korrigiert (editiert) werden und nicht wie heute gewohnt, durch einfache CURSOR-Bewegungen über den ganzen Bildschirm hinweg.

Der Vorteil von ED liegt eigentlich allein darin, daß gleichgültig wie 'exotisch' die angeschlossene Konsole auch angesteuert werden muß, mit ED kann immer gearbeitet, da nur die Steuerfunktionen CR und LF benötigt werden (die genormt sind). Im Übrigen wird nur der Standard-ASCII-Zeichensatz benötigt.

Es würde den Rahmen dieser Zusammenstellung bei weitem sprengen wenn der Umgang mit den nachfolgend besprochenen Programmen detailliert ausgeführt würde. In diesem Kapitel sollen lediglich die Aufrufkonventionen und die entsprechenden Steuerbefehle zusammengestellt werden. Die entsprechenden Handbücher von DIGITAL RESEARCH geben genauere Auskünfte.

Das Arbeiten mit dem Programm GENCPM.COM wird in Band I (Hardware-Anpassung) ausführlich besprochen.

ED Beispiel.BAT
Program aufrufen FnA danach B*T

****
* ED *
****

der Editor

Syntax:

ED (d:) (dateiname.ext)

Beschreibung:

ED ist ein zeilenorientierter Editor,geeignet für relativ einfache Textverarbeitung, vor allem aber gedacht zum Schreiben von (Assembler-) Programmen zur Systemanpassung.

ED kann bereits mit dem zu bearbeitenden Dateinamen aufgerufen werden oder falls dies nicht der Fall ist, fragt ED nach weiteren Informationen wie das folgende Beispiel zeigt:

```
Enter Input File: test.txt

Enter Output File:

NEW FILE

*
```

Unter Inputfile ist der Name zu verstehen, unter welchem die Datei auf der Diskette abgelegt wurde oder der Name einer neu zu eröffnenden Datei.

Wird eine bereits vorhandene Datei aufgerufen, so kann diese unter einem anderen Namen abgelegt werden, wenn die zweite Frage (Output File) entsprechend beantwortet wird.

Die zweite Frage kann auch allein oder zusätzlich mit einer Laufwerksbezeichnung (gefolgt von einem Doppelpunkt) beantwortet werden. In diesem Falle wird die Datei nach erfolgter Bearbeitung auf der Diskette in dem genannten Laufwerk abgelegt. Wird die Frage nur mit <cr> beantwortet, wird die einzulesende Datei in filename.BAK umbenannt und die neu abzulegende Datei erhält den 'alten' Dateinamen.

Nach diesen 'Vorbereitungen' meldet sich ED mit seinem Befehlsprompter '*'. Mit den in Tabelle 9 genannten Befehlen kann nun ein beliebiger Text editiert werden.

Zum 'Start' soviel:

In den Schreibmodus kommt man mit dem I(nsert)-Befehl. Wurde eine Datei benannt, die bereits vorhanden ist, muß diese erst mit dem A(ppend)-Befehl in den Editier (Schreib) -Puffer geladen werden.

Ein recht umfangreicher Befehlssatz macht es dann möglich, die 'Eintritts'-Stelle zu suchen und dort im dem I-Befehl in den Text hineinzukommen oder mit entsprechenden anderen Befehlen sonstwie zu editieren.

Befehl	Auswirkung
nA	fügt n Zeilen aus der aufgerufenen Datei an den Speicherpuffer an (Append).
ØA	fügt soviele Zeilen aus der aufgerufenen Datei an den Speicherpuffer an, bis dieser halb voll ist.
#A	fügt soviele Zeilen aus der aufgerufenen Datei an den Speicherpuffer an, bis dieser voll oder die Datei zu Ende ist.
B, -B	setzt den Zeilenzeiger auf den Beginn oder das Ende (-B) des Textes im Speicherpuffer (Beginn).
nC,-nC	setzt den Zeilenzeiger n Zeichen vorwärts oder rückwärts (-nC) (Columne).
nD, -nD	löscht n Zeichen nach (nD) oder vor (-nD) dem aktuellen Zeichen (Delete).
E	legt die editierte Datei ab und kehrt zum CCP zurück (Exit).
Ftext^Z	finde 'text' ab Zeilenposition (Find).
H	Zwischenspeichern der editierten Datei ohne den Editor zu verlassen (Hold).
I	Start des Einfuegungsmodus (Insert). Es können nach I beliebige Zeichen eingefügt werden. Aussprung aus dem I-Modus mit CNTRL-Z oder ESC-ape. ACHTUNG: Groß-I fügt nur Großbuchstaben ein, Klein-i beides.
nK, -nK	löschen von n Zeichen vor (-n) oder ab der aktuellen Position. (Kill)
nL,-nL	n Zeilen vor oder (-n) zurück. (Line)
nMbefehl	den Befehl <befehl> n-mal ausführen.
n,-n	n Zeilen vor oder zurück (-n) und diese Zeile ausgeben. (Make)
n:	Zeiger auf Zeile n stellen.
n:befehl	führe Befehl <befehl> zeilenweise bis Zeile n aus.
Ntext^Z	suche den Text <text> über den ganzen Speicherpufferbereich (Next).

Tabelle 9 Zusammenfassung der ED-Befehle

Befehl	Auswirkung
0	mache alle Änderungen seit Beginn der Editierphase rückgängig. (Original)
nP,-nP	es werden die nächsten (n) Zeilen ausgegeben oder die letzten (-n) Zeilen. (Print)
Q	Abbrechen der momentanen Editerarbeit ohne Abspeichern der Datei (Quit).
Rfilename.ext^Z	Datei filename.ext in den Datenpuffer lesen. A-Befehl zum 'anhängen' benutzen ! (Read)
Stext1^Ztext2^Z	löschen von text1 dafür einfügen von text2 (Substitute).
nT,-nT	Ausgeben von n Zeilen Text (Type).
U,-U	Textumwandlung von Kleinbuchstaben in Großbuchstaben. (Uppercase)
V,-V	Zeilennummerierung AN/AB-Schalten.
0V	Ausgabe des restlichen Speicherplatzes.
nW	Ausgabe von n Zeilen auf Diskette an bearbeitete Datei (Write).
nXfilename.ext^Z	Ausgabe von n Zeilen an die Datei <filename.ext>. Die Zeilen werden an eine evtl vorhandene Datei angehängt.
0Xfilename.ext^Z	lösche Datei <filename.ext>.

noch **Tabelle 9 Zusammenfassung der ED-Befehle**

frei für Notizen zum 'eigenen' Editor

```
**********************
* LIB                *
**********************
```

der LIBrary-manager

Syntax:

```
LIB filename([I¦M¦P¦D])
LIB filename([I¦M¦P])=filename(modifier)(,filename(modifier)...)
```

Beschreibung:

Mit LIB werden sogenannte library-Dateien (Programm-Bibliotheken) zusammengebunden.

Es handelt sich dabei um (REL)-Dateien immer wieder benutzter Unterprogramme, die mit LINK zu ablauffähigen Programmen zusammengebunden werden. Entsprechende REL-Dateien werden von RMAC erzeugt aber auch von anderen Assemblern wie M80 oder Z80ASM.

In jedem Falle wird das sog. Microsoft(R)-REL-Format verwendet. LIB-Dateien sind vor allem für den geübten Assemblerprogrammierer von Nutzen aber auch dem Programmierer von Hochsprachen wie PL1 oder FORTAN, die derartige LIB-Dateien in ihre Programme einbinden können.

Es stehen folgende Optionen zur Verfügung:

- I es wird eine indizierte Datei ....IRL generiert, auf die LINK schneller zugreifen kann.
- M Modul-Option, die Namen der einzelnen Module werden hier auf den Bildschirm ausgegeben
- P Die Public-Option schließt die M-Option ein und gibt darüberhinaus noch die Namen aller public-Variablen an.
- D Diese Option gibt den Inhalt der Module in ASCII aus.

Zur Erweiterung, Veränderung und Auswahl von Modulen stehen folgende 'modifier' (siehe Syntax) zur Verfügung:

delete	<module=>
replace	<module=filename.REL> oder <filename> wenn Modulname und 'filename' gleich sind.
select	(anfangsmodul-endmodul,modl,mod2,...modn)

Dies sei durch nachfolgende Beispiele verdeutlicht:

```
A>LIB test[p]=test1,test2,test4,test5,test6
```

Hier wird eine neue LIB-Datei TEST.REL erzeugt, welche die Dateien test1.REL bis test6.REL enthält. Alle Modulnamen (TEST1 bis TEST6) werden auf den Bildschirm ausgegeben, darüberhinaus noch alle public-Namen (d.h. die Labels der eingebunden Unterprogramme). Der Sinn dieser Ausgabe ist es, festzustellen (und evtl. mit dem Drucker zu protokollieren) welche Programmteile eine LIB-Datei enthält.

```
A>LIB mylib=test(test1,test3),console(auxin-lstout,clock)
```

Hier wird die LIB-Datei MYLIB.REL aus der LIB-Datei TEST.REL generiert. Die neue Datei enthält jedoch nur die Module test1 und test3. Darüberhinaus wird die LIB-Datei CONSOLE.REL eingebunden jedoch nur mit den Programmteilen AUXIN ... LSTOUT und CLOCK.

****
* LINK *
****

der Linker

Syntax:

```
LINK (d:)(filename)([option])=filename1([o])(,.....
```

Beschreibung:

Mit LINK.COM werden REL(ative)-Dateien zu einem ablauffähigen COM-Programm ge'linkt' (gebunden).

LINK ermöglicht es, zumindest während der Entwicklungsphase, Programme in kleine Module aufzuspalten, die wesentlich einfacher zu bearbeiten und zu 'debuggen' sind. Die einzelnen Module werden dann erst am Schluß, wenn alles fehlerfrei ist, zu einem Programm zusammengebunden (gelinkt).

Wird genau genug definiert, was jedes Modul 'können' soll, so ist es auch absolut möglich, daß die Programmentwicklung von mehreren Programmieren einzeln geschehen kann.

REL(ative)-Programme, sind Programmsegmente, die keine absoluten Adressangaben enthalten, vielmehr werden alle Adressen erst beim Linken definiert. (Alle Adressen sind RELativ zu einer noch zu definierenden Startadresse).

Ein typisches Beispiel des Linkens zeigt das Zusammenbinden des BIOS-Teiles von CP/M 3, das seinerseits aus mehreren Modulen besteht:

```
A>link bnkbios3[b]=scb,bioskrnl,chario,move,diskio,boot

LINK 1.31

@A1VEC   FE26  @AOVEC   FE28  @BFLGS   FE57  @BNKBF   FE35
@CIVEC   FE22  @COVEC   FE24  @CRDMA   FE3C  @CRDSK   FE3E
@DATE    FE58  @ERDSK   FE51  @ERMDE   FE4B  @FX      FE43
@HOUR    FE5A  @LOVEC   FE2A  @MEDIA   FE54  @MIN     FE5B
@MLTIO   FE4A  @MXTPA   FE62  @RESEL   FE41  @SEC     FE5C
@USRCD   FE44  @VINFO   FE3F  @CBNK    01EA  @CNT     01E8
@CTBL    05U7  @DBNK    01E9  @DMA     01E6  @DRV     01E1
@DTBL    04E6  @SECT    01E4  @TRK     01E2  DSKSTA   0ED1
GETADR   0BA1  GETDPH   0E91  IPCHL   0E90  MOTOFF   0EE4
SNDSTL   02AA  TABLE1   017B  CLKGET   0E63  CLKSET   0E35
DSKNAM   02CC  I2CINI   0E91  CURDPH   14EE

ABSOLUTE     0000
CODE SIZE    0A6D (0000-0A6C)
DATA SIZE    0DFD (0B00-18FC)
COMMON SIZE  0000
USE FACTOR     3F
```

Im Beispiel wird eine Datei BNKBIOS3.SPR generiert. Alle als 'public' oder Global definierten Adressen werden ausgegeben. Unter Globalen Adressen versteht man Labels (Unterprogramm-Marken), die in einem Modul als Programm vorhanden sind, in einem anderen Modul aber aufgerufen werden.

Zur Definition stehen verschiedene Assembler-Anweisungen zur Verfügung. Nach den Label-Angaben mit den zugehörigen Adressen folgen Angaben über das gebundene Programm:

ABSOLUT — das heißt unveränderlich sind 0000 Programmsegmente.

CODE SIZE — in diesem Fall der Anteil an Programmsegmenten im gemeinsamen Speicherbereich sind 0A6DH Bytes, beginnend bei der (immer noch relativen) Adresse 0000H.

DATA SIZE — in diesem Falle der Anteil im gebankten Speicherbereich sind 0DFDH Bytes, beginnend bei Adresse 0B00H.

COMMON — ist bei der benutzten Definition nicht vorgesehen.

Der USE FACTOR gibt an wieviel Prozent des vorhandenen Speicherplatzes während des Linkens benutzt wurde.

A	Zwischendateien werden auf Diskette angelegt, notwendig wenn der Speicherplatz im Arbeitsspeicher zu klein wird.
B	generiert eine SPR-Datei.(notwendig z.B. zum generieren von CPM3.SYS)
Dnnnn	Beginn des Datensegmentes ab Adresse nnnn (Hex).
Glabel	Startadresse des Programmes wenn dieser nicht am Programmanfang. Es wird von LINK dann eine zusätzliche Programmzeile (JP label = vielmehr C3 labeladresse) generiert.
Lnnnn	Startadresse des Programmes (Vorgabe ist 100H). Diese Adresse ist genauergenommen die Adresse des ersten Programmbytes. Im Unterschied hierzu geben die Adressen Pnnn und Dnnnn nur den Beginn eines entsprechenden Segmentes an.
NL	Unterdrückt Ausgabe der SYM(bol)-Tabelle.
NR	legt keine SYM(bol)-Datei an.
Phhhh	Beginn des Programmsegmentes (Vorgabe ist 100H).
Q	Alle Symbole die mit ? beginnen werden ausgegeben.
S	vorangestelle Datei ist als LIB-Datei zu behandeln
$Cd	Konsolenausgaben werden auf Ausgabeeinheit 'd' umgelenkt. Möglich ist außer den Laufwerksbezeichnungen A..P noch zusätzlich Y=LST (Ausgabe auf den Drucker), X=CONOUT (Ausgabe auf die Konsole) und Z=NULL (keine Ausgabe). Vorgegeben ist X
$Id	evtl. notwendige Zwischendateien werden auf Laufwerk 'd' ausgegeben. Vorgabe ist das Laufwerk von dem aus LINK aufgerufen wurde.
$Od	Zieladresse der COM-Datei. 'd' kann Laufwerk A-P sein oder Z=NULL (siehe hierzu die $I-Option).
$Sd	Zieladresse der Symboltabelle (SYM-Datei). 'd' kann Laufwerk A..P sein oder Z=NULL

Tabelle 10 Optionen von LINK.COM

****
* (R)MAC *
****

die macro-Assembler

Syntax:

```
MAC filename ($options)
RMAC filename ($rd:$sd:$pd)
```

Beschreibung:

Assembler dienen dazu aus sogenannten Assemblerprogrammen (Quell-Programmen), die in verhältnismäßig einfach zu merkenden Mnemonics (Kürzel von Programmieranweisungen) geschrieben sind, einen code zu erzeugen, den die entsprechende CPU direkt verstehen und abarbeiten kann.

Bei Assemblerprogrammen wird im Gegensatz zu Hochsprachenprogrammen jedes Register und jedes Flag der CPU direkt angesprochen. Assemblerprogramme sind meist schlechter zu 'durchschauen' und wesentlich aufwendiger zu schreiben als Hochsprachen-Programme, haben dafür jedoch den durch nichts zu ersetzenden Vorteil kompakt und schnell zu sein. Keine Hochsprache ist (zumindest bisher) in der Lage auch nur eine dieser Bedingungen zu erfüllen.

Es kann in diesem Zusammenhang auf die vielfältigen Möglichkeiten der Assemblerprogrammierung nicht eingegangen werden. Beispiel sind jedoch im Anhang zu finden, gute Fachbücher zu diesem Thema darüberhinaus leicht erhältlich.

Der Unterschied zwischen MAC.COM und RMAC.COM liegt allein in der Tatsache, daß nur RMAC.COM sogenannte REL-Dateien erzeugen kann, während MAC nur die macro-Version des 'alten' (CP/M 2) ASM.COM ist. Die Optionen beider Assembler sind in Tabelle 11 zusammengefaßt.

MAC und RMAC erwarten eine Eingabedatei mit dem Index filename.ASM. Als Ausgabedateien können die Dateien filename.PRN und filename.SYM erstellt werden, wobei die PRN-Datei das sogenannte Listing, d.h. das komplette Programm mit allen Adressangaben und dem Befehlscode in HEX enthält, während die SYM-Datei die u.U. für (Z)SID benötigten Labeladressen enthält. Von MAC.COM wird darüberhinaus noch eine Datei filename.HEX während RMAC.COM die Datei filename.REL generiert.

Die beiden zuletzt genannten Dateien sind Vorstufen der eigentlichen Programme. HEX-Dateien werden mit HEXCOM in COM-Dateien umgewandelt oder mit (Z)SID und anderen HEX-Dateien zu einem Gesamtprogramm zusammengebunden, während REL-Dateien mit LINK in COM-Dateien gewandelt werden. Nachfolgend Beispiel der Aufrufkonventionen:

```
A>mac change
CP/M MACRO ASSEM 2.0
04B4
003H USE FACTOR
END OF ASSEMBLY

A>rmac change
CP/M RMAC ASSEM 1.1
04B4
003H USE FACTOR
END OF ASSEMBLY
```

Ad	Laufwerk auf welchem sich die ASM-Datei befindet möglich ist Laufwerk A..O.
Hd	Laufwerk auf welchem die HEX-Datei abgelegt wird, möglich ist Laufwerk A..O und Z=keine Ausgabe.
Ld	Laufwerk auf welchem sich LIB-Dateien (Aufgerufen von MACLIB-Befehlen) befinden.
Pd	Laufwerk auf welchem die PRN-Datei abgelegt wird, Möglich ist Laufwerk A..O, Z=keine Ausgabe, P=LST und X=CONOUT.
Sd	Laufwerk auf welchem die SYM-Datei abgelegt wird, möglich ist Laufwerk A..O, Z=keine Ausgabe, P=LST und X=CONOUT.

Für RMAC sind nur die Optionen Pd und Sd gültig, anstelle von Hd tritt Rd als Ziellaufwerk auf welchem die REL-Datei abgelegt wird.

Werden keine Optionen angegeben, so ist immer das Laufwerk von welchem aus das Programm aufgerufen wurde das Quell- und Ziel-Laufwerk.

Nur für MAC - zusätzliche Optionen:

+L	Eingabezeilen aus MACRO-Buecherei werden aufgelistet.
-L	Eingabezeilen aus MACRO's werden unterdrückt. (dies ist die Vorgabe)
+M	Alle MACRO's werden aufgelistet wie sie während des Assemblerdurchlaufes auftreten.
-M	MACRO's werden nicht aufgelistet (Vorgabe).
+Q	Alle (Macro-)LOCALS werden in die SYM-Liste übernommen.
-Q	LOCALS werden nicht in die SYM-Liste übernommen (Vorgabe).
+Sd	SYM(bol)-Tabelle wird an PRN-Listing angehängt mit Ziellaufwerk (siehe oben).
-S	Es wird keine SYM(bol)-Tabelle generiert.

Tabelle 11 Optionen der Assembler MAC und RMAC

****
* PIP *
****

Das Kopierprogramm

Syntax:

PIP zielname[d:][[Gn]]=quellname.ext[[Optionen][,..]][d:][[Optionen]]

Beschreibung:

PIP (Peripheral Interchange Programm) ermöglicht es, Kopien von einer oder mehreren Dateien von dem einem auf das andere Laufwerk oder von der einen auf die andere USER-Ebene zu kopieren. Die Dateien können dabei umbenannt, aneinandergehängt oder nur teilweise kopiert werden. Sie können in Groß- oder Kleinbuchstaben gewandelt oder mit Zeilennummern versehen werden, wenn es sich um Textdateien handelt. Kopierziel kann neben einem Laufwerk auch eine andere logische Einheit wie z.B. ein Drucker oder der AUX-Kanal sein.

PIP kopiert alle Systemattribute und fragt bei Kennwort-geschützten Dateien nach dem Paßwort.

1. Kopieren von Dateien.

Es können Dateien einzeln oder mit Joker zum Kopieren aufgerufen werden.

Als erste Angabe muß das Kopierziel angegeben werden (gefolgt von einem Doppelpunkt), danach folgt der Name unter welchem die zu kopierende Datei im Kopierziel abgelegt werden soll. Bleibt der Name der gleiche, kann er bei der Eingabe entfallen. Die Angabe der Kopierquelle folgt als nächstes, getrennt durch ein vorangestelles '='-Zeichen. Ist die Kopierquelle das Laufwerk, von welchem aus PIP aufgerufen wurde, kann eine entsprechende Angabe entfallen. Als letztes folgt der Name der zu kopierenden Datei, der Joker enthalten darf. (*.* bedeutet dann ALLES kopieren).

Es sind dann nur noch einige Optionen möglich, vom Dateinamen getrennt durch eckige Klammern.

Die nachfolgenden Beispiele sollen dies verdeutlichen, alle Optionen sind in Tabelle 12 zusammengefaßt.

```
A>pip b:=test.com
A>pip b:=*.*
A>pip b:=*.*[vor]
A>pip b:myfile.asm[G1]=a:lib.mac[1]
A>pip A:=B:*.mac
```

2. Zusammenfassen von Dateien

Mit PIP kann (auf dem gleichen Laufwerk) auch eine Datei so kopiert werden, daß eine zweite Datei mit anderem Dateinamen entsteht.

Diese Möglichkeit ist dann von Nutzen, wenn eine vorhandene Datei unverändert bestehen bleiben, sie andererseits aber als 'Grundstock' für ein neues Programm dienen soll. In diesem Falle ist es auch möglich, aus mehreren Dateien eine neue Datei zu generieren.

Als Zieldatei kann natürlich in diesem Falle auch ein anderes Laufwerk dienen.

Werden als Quelldatei mehrere Dateien (durch Komma getrennt) angegeben, so bedeutet dies immer, daß die angegebenen Dateien in EINER Zieldatei zusammengefaßt werden.

Diese Möglichkeit der Programm'vervielfachung' ist heute weniger gefragt, sie kann mit modernen Editoren einfacher und übersichtlicher geschehen.

```
A>PIP newfile.mac=newfile.asm
A>PIP newfile=file1,file2,file3
A>PIP b:test.bas=master.bas,b:final.bas
```

3. Kopieren von oder nach logischen Einheiten.

Dieser Sonderfall des Kopierens erlaubt das Ausgeben oder Einlesen von Dateien auf bzw von den logischen Einheiten AUX, CON und LST.

Die zugelassenen Ziel- bzw Quell-Einheiten zeigt folgende Aufstellung:

als Ziel	als Quelle	
AUX:	AUX:([o])	AUX-Eingabe oder Ausgabekanal
CON:	CON:([o])	Konsolen Eingabe oder Ausgabekanal
PRN:		Druckerausgabe jedoch mit TAB-Expansion und Seitenvorschub nach 60 Zeile
LST:		Druckerausgabe
	NUL:	Eine (imaginaere)Quelldatei die 40 mal 00H ausgibt
	EOF:	Eine (imaginaere) Quelldatei die 1x CNTRL-Z (1AH), ausgibt. Diese Zeichen ist die (Text)Endemarkierung

Die folgendenden Beispiele zeigen einige Anwendungen:

Im ersten Falle wird zuerst jede Konsoleneingabe, d.h. alles was über die Tastatur eingegeben wird, auf den Drucker ausgegeben. Nach Eingabe eines CNTRL-Z werden hintereinander die Dateien myfile und yourfile auf den Drucker ausgegeben.

Im zweiten Falle wird jegliche Tastatureingabe (die ja auf CON wiedergegeben wird) in einer Datei reserve.all abgespeichert.

Beendet wird dies durch ein CNTRL-Z.

```
A>pip prn:=con:,myfile,yourfile
A>pip b:reserve.all=CON:
```

PIP kann auch ohne jede weitere Angabe aufgerufen werden. Es meldet sich dann zuerst mit Namen und dann mit einem Stern '*' als Prompter.

Es kann nun wie zuvor gezeigt gearbeitet werden. Der Nutzen liegt hier darin, daß PIP nicht mit jedem Aufruf neu von Diskette geladen werden muß, sondern alle weiteren Befehle aus dem Arbeitsspeicher heraus bearbeitet.

Muß PIP Dateien kopieren, die größer als der freie Arbeitsspeicher sind, so wird die Kopierarbeit in mehreren Schritten erledigt.

Wird PIP mit den Optionen U oder L belegt (die sich natürlich gegenseitig ausschließen) kann bei größeren Dateien schon einmal eine 'Ruhepause' eintreten, die bis zu einer Minute betragen kann. In jedem Fall meldet sich PIP mit dem CCP-Prompter oder bei Direktaufruf mit dem *-Prompter zurück.

```
A>PIP
CP/M 3 PIP Version 3.0
*b:=*.com[vor]
*b:=*.mac[vo]
*b:=c:*.*[vor]
*<cr>
A>
```

A	Es werden nur die Dateien kopiert, bei welchen das ARCHIVE-Attribut zurückgesetzt ist (OFF).
C	Bei Vielfachkopien mit JOKERN wird bei jeder Datei vor dem Kopieren nachgefragt ob kopiert werden soll oder nicht (Y=Ja, N=Nein).
Dn	Lösche alle Zeichen nach dem n-ten Zeichen bis <cr> oder <lf> folgt. Diese Option dient vor allem zur Unterdrückung von Zeichen, wenn z.B. ein Drucker eine Spaltenbreite von 80 Zeichen hat, der Quelltext aber häufig länger ist, sonst würde der Text auf die nächste Zeile überlaufen.
E	Echo beim Transfer von Text-Dateien auf den Bildschirm.
F	Alle Form-Feed (Blattvorschub beim Drucker oder Bildschirm löschen bei einigen Terminals) werden unterdrückt.
Gn	Wenn n einem Quellnamen (woher) folgt, wird die entsprechende Datei aus der USER-Ebene n gelesen. Wenn n einem Zielnamen (wohin) folgt, wird die Datei in die entsprechende USER-Ebene geschrieben.
H	Transfer von HEX-Daten. PIP überprüft die Daten auf Korrektheit, im Fehlerfalle wird dies gemeldet
I	:00 Dateien im HEX-Transfer werden ignoriert mit 'I' wird automatisch auch die H-Option gesetzt.
L	Mit dieser Option werden beim Kopieren von Text-Dateien alle Großbuchstaben in Kleinbuchstaben gewandelt. Diese Option muß der Angabe der Zieldatei folgen.
N	In eine Textdatei werden Zeilennummern eingefügt. Nach jeder Zeilennummer folgt ein TAB.
O	Notwendig beim Transfer von Binär-Dateien. Es werden alle CNTRL-Z und EOF-Marken ignoriert.
Pn	Textdateien werden nach jeweils n Zeilen mit einem Form-Feed unterbrochen (Seitenvorschub). Wird die Option mit n=1 oder garnicht spezifiziert erfolgt ein Form-Feed nach jeweils 60 Zeilen.

Tabelle 12 Zusammenstellung der Optionen von PIP

Qtext^Z	ASCII-Datei vom Quellaufwerk kopieren bis zum Beginn der Zeichenkette <text>. Kann mit der Option S benutzt werden, um Textauszuege aus einer Datei zu kopieren.
R	Einzige Möglichkeit um SYS-Dateien zu kopieren
Stext^Z	ASCII-Dateien von Quellaufwerk kopieren ab Beginn der Zeichenkette <text>.
Tn	Beim Kopieren von Textdateien werden mit dieser Option TAB-s (CNTRL-I) durch soviele Leerzeichen (Blanks) ersetzt, daß das nächste Zeichen in einer Spalte beginnt, die ein Vielfaches von n ist
U	Alle Kleinbuchstaben der Quelldatei werden in Großbuchstaben umgewandelt.
V	Wenn die Zieladresse ein Laufwerk ist, wird die kopierte Datei mit der Originaldatei noch einmal verglichen, um Kopierfehler zu vermeiden.
W	Möglichkeit um R(ead)-O(nly)-Dateien zu überschreiben. Vorsicht !!!
Z	Mit dieser Option ist es möglich eine ASCII-Datei zu kopieren, bei der alle Parity-Bits (Bit 7) auf NULL gesetzt werden.

noch **Tabelle 12 Zusammenstellung der Optionen von PIP**

****
* SID *
****

Syntax:

SID (filename.ext)(,filenme.sym)

Beschreibung:

SID (Symbolic Instruction Debugger) ermöglicht es dem erfahrenen Programmierer, Fehler (Bugs=Wanzen) in einem neu geschriebenen Programm relativ schnell zu finden.

Hierzu stehen eine Vielzahl von Hilfsmitteln zur Verfügung. Sie reichen von der einfachen Darstellung des Speicherinhaltes (in HEX und ASCII) über die Möglichkeit des Disassemblierens bis zum Verfolgen (tracen) eines Programmes in Einzelschritten mit fortlaufender Darstellung der Registerinhalte.

Nachdem sich SID mit Namen und Versionsnummer gemeldet hat, werden folgende Angaben gemacht:

```
NEXT   MSZE    PC   END
xxxx   xxxx   xxxx  xxxx

#
```

Damit erhält der Benutzer Angaben über die derzeitige Größe des Arbeitsspeichers (MSZE), die Adresse des ersten freien Speicherplatzes hinter einem evtl. geladenen Programme (NEXT), der Endadresse im Arbeitsspeicher (END) und die Einsprungadresse des geladenen Programmes (PC=Program-Counter).

In der nächsten Zeile meldet sich dann der SID-Prompter '#' und wartet auf entsprechende Befehle. Diese sind in Tabelle 13 zusammen gefaßt.

SID hat für Benutzer eines Z80-Computer den Nachteil, daß er 'nur' die Register und den Befehlssatz (und die Mnememonics) des 8080-Prozessors kennt.

Da heute überwiegend Z80-Prozessoren mit ihrem wesentlich erweiterten Befehlssatz auf dem Markt sind, wird der Benutzer in den meisten Fällen auf einen anderen Debugger wie z.B. TRACE-80 oder DSD zurückgreifen oder seinen 'alten' ZSID, der Z80-Version von SID unter CP/M 2, weiterbenutzen. In letzterem Falle sei auf den TPA-Befehl SAVE.COM verwiesen, wenn Dateien wieder auf Diskette zurückgeschrieben werden sollen.

Zu SID werden noch zwei Hilfsprogramme geliefert und zwar HIST.UTL und TRACE.UTL. Beide Programme werden wie 'normale' Programme nachgeladen und ermöglichen es Histogramme (HIST) zu erstellen oder die Daten eines Tracevorganges zu sammeln. Nähere Angaben hierzu sind im Handbuch von Digital Research SID (TM) Symbolic Instruction Debugger User's Guide (in englischer Sprache) zu finden.

Syntax	Befehlsbedeutung
Annnn	Programmeingabe in Assembler-Mnemonic (INTEL) ab Adresse nnnn.
Cnnnn[a[,b]]	Aufruf (Call) eines Programmes mit Start auf Adresse nnnn. Beim Aufruf koennen folgende Daten uebergeben werden: a=Inhalt des BC-Registers und b=Inhalt des DE-Registers. (Beide Register sind die (CP/M-) ueblichen Uebergaberegister.
D[W][nnnn][,eeee]	Dump (in HEX und ASCII) eines Speicherbereiches. Die Option W veranlasst eine Ausgabe im Wortformat (jeweils 2 Byte) nnnn gibt die Start Adresse an, eeee die Endadresse. Wird keine End-Adresse spezifiziert, werden nur die naechsten 128 Bytes ausgegeben. Wird keine Startadresse angegeben, wird ab der zuletzt dargestellten Adresse (beginnend bei 100H) ausgegeben.
Efilename.ext[(filename.SYM)	Laden eines Programmes (mit optional, seiner Symboltabelle).
E*filename	Laden der SYM(bol)-Tabelle filename.SYM im Nachgang zu Efilename.ext.
Fnnnn,eeee,bb	Fuellen des Speicherbereiches nnnn bis eeee mit dem Bytewert bb.
G[nnnn][,b[,b]]	Ansprung (Go) der Adresse nnnn. Es koennen bis zu 2 Haltepunkte (b) gesetzt werden. Erreicht das Programm den Haltepunkt, wird dessen Adr. ausgegeben und in den Befehlsmodus zurueckgekehrt. Die Registerinhalte und der Stack (siehe X-Befehl) entsprechen dem vorgefundenen Zustand. Wird nur ein G eingegeben, wird ab dem letzten Haltepunkt weiter gemacht, bis ein neuer Haltepunkt gefunden wird.
H[aaaa[,bbbb]]	Dieser Befehl allein gibt eine Auflistung aller Symbole (die mit dem E-Befehl geladen wurden) aus. Wird die Option aaaa benutzt,wird der entsprechende Wert in HEX(adezimal) Dezimal und ASCII ausgegeben Werden beide Optionen (aaaa und bbbb) benutzt, wird die Summe und die Differenz beider Werte ausgegeben.
Ibefehl	Es kann mit diesem Befehl ein kompletter CCP-emuliert werden. Dieser wird entsprechend aufbereitet, d.H. in die entsprechenden FCB-Puffer geschrieben, wie dies fuer jeden Befehlsaufruf aus dem CCP heraus typisch ist.
L[aaaa[,eeee]]	Disassembler-Listing (in INTEL-8080 Mnemonic) ab Adresse aaaa bis Adresse eeee. Wird nur die Start-Adresse angegeben, so werden nur die naechsten 8 Zeilen ausgegeben. Wird keine Adresse angegeben, erfolgt die Ausgabe ab der zuletzt ausgegebenen Adresse beginnend bei 100H.
Maaaa,eeee,dddd	Transfer (move) des Speicherinhaltes von aaaa bis eeee nach Startadresse dddd.
P[nnnn[,cccc]]	Setzen eines permanenten (festen) Haltepunktes auf Adresse nnnn. Dieser Haltepunkt wird (falls Option gesetzt) cccc mal durchlaufen, bis er angezeigt wird. Wird nnnn nicht spezifiziert, werden alle gesetzen Haltepunkte zurueckgenommen.

Tabelle 13 Zusammenfassung der SID-Befehle

Syntax	Befehlsbedeutung
Rfilename.ext(,nnnn)	Einlesen einer Datei <filename.ext> nach Adresse 100H. Die Option gibt einen Offset zur Adresse 100H an, an welche die Datei dann geschrieben wird.
S(W)nnnn	Setzen von Speicherstellen ab Adresse nnnn. Mit Option W werden Speicherstellen Wortweise (2-Byte) gesetzt, sonst Byteweise.
T(W)(nnnn(,cccc))	Trace (Einzelschritt-Verfolgung) eines Programmes ab Adresse P (Programmzaehler siehe X-Befehl). Ist die Option nnnn gesetzt, wird nnnn-mal ge'traced'.Die Option cccc gibt eine andere Startadresse als in P vorgesehen an. Mit der Option W wird die Verfolgung von Unterprogrammaufrufen unterbunden.
U(W)(nnnn(,cccc))	Entspricht dem T-Befehl, es erfolgt jedoch keine Ausgabe der Registerinhalte.
V	Gibt die erste freie Speicherstelle (NEXT) an, die naechste Speicherstelle hinter der groessten eingelesenen Datei (MSZE), den Inhalt des Programmzaehlers (PC) und die letzte verfuegbare Speicherstelle (END).
Wfilename.ext(aaaa,eeee)	Schreiben der Datei <filename.ext> von Adresse aaaa bis eeee auf Diskette. Wird keine Startadresse angegeben wird 100H vorgegeben und NEXT-1 als Endadresse verwendet.
X(f!r)	Es werden die Inhalte aller CPU (8080) Register dargestellt wenn keine Option eingegeben wurde. mit Option f koennen die Flagbits C,E,I,M oder Z gesetzt (1) oder zurueckgesetzt (0) werden. Mit Option R koennen die Register A,B,D,H,P und S veraendert werden. Nach Eingabe der Option wird der derzeitige Zustand bzw Wert ausgegeben, erst dann wird nach dem neuen Wert gefragt.
#	Wird dieses Zeichen einer Zahl vorhangestellt, wird die Zahl als dezimal behandelt.
'a'	Zeichen oder Zeichenketten in Hochkommas <'> werden als ASCII-Werte behandelt.
^C	oder G0 veranlassen einen Wiedereintritt in den CCP (Warmstart). Es gelten alle Kontrollzeichen des CCP-Befehlszeileneditors. (Siehe Tabelle 1)
ACHTUNG:	Diese Befehle sind gegenueber dem Befehlssatz von (Z)SID in CP/M 2 erheblich erweitert und zu diesem nur bedingt kompatibel.

noch **Tabelle 13 Zusammenfassung der SID-Befehle**

****
* XREF *
****

Crossreferenz-Erzeugung

Syntax:

XREF (d:) filename ($P)

Beschreibung:

Das Programm XREF.COM ermöglicht es, eine Zusammenstellung aller Labels in einem Programm zu erstellen, mit Querverweisen, an welcher Stelle in einem Programm diese Labels aufgerufen werden.

Unter Labels sei in diesem Falle der Name eines Programmsegmentes in einem Gesamtprogramm verstanden. Diese Zusammenstellung ist nur für Assemblerprogramme, die mit MAC oder RMAC assembliert wurden, möglich. Es werden dazu die PRN und SYM-Dateien benötigt. Beide Dateien müssen den gleichen Dateinamen haben.

Es wird eine neue Datei filename.XRF generiert, die alle Querverweise enthält. Die Option $P veranlaßt, daß die XRF-Datei nicht auf Diskette abgelegt, sondern auf den Drucker ausgegeben wird.

TEIL II Aufbau und Organisation von CP/M 3

Kapitel 1 Eine Vorbemerkung

Im Gegensatz zum weit verbreiteten CP/M 2.2 (und dem noch etwas 'älteren' CP/M 1.4) ist CP/M 3 nicht mehr auf einen Arbeitsspeicher von 64K beschränkt. Es können in der gebankten Version, und nur die ist eigentlich von Interesse, Speicher bis zu 1MB verwalten werden. Leider bezieht sich dies aber nur auf Daten- und DIR-ectory-Puffer und nicht auf die Größe des eigentlichen Arbeitsspeichers (der TPA).

Da die bei CP/M 3 (CP/M Plus) benutzten Prozessoren nach wie vor nur 16 Bit adressieren können, ist der direkt ansprechbare Speicherbereich nach wie vor 64K groß. Um dies zu umgehen, wurde der größere Speicherbereich einfach in Speicherbanken (banks) zu jeweils 64K aufgeteilt. Zum Datentransfer wird nun im obersten Speicherbereich einer Bank ein Speicher-Segment zur Verfügung gestellt, auf welches von allen Banken zugegriffen werden kann.

Dieses gemeinsame Speichersegment ist üblicherweise zwischen 4K und 8K groß und enthält neben dem genannten Datenpuffer auch den ungebankten Teil des CP/M's.

Die Puffergrößen für Daten und DIR-ectory-Einträge können bei der Generierung eines Systemes relativ freizügig (nach Vorhandensein von Speicherplatz) bestimmt werden, mit beispielsweise nur 2 Laufwerken wird man jedoch schnell feststellen, daß mehr als 2 Banks nicht sinnvoll belegt werden können.

Da heute Speicherkarten mit 1MB billiger sind als vor wenigen Jahren noch Speicherkarten mit 8,16 oder gar 32K, ist die Ausnutzung von 'verbleibendem' Speicherplatz schon ein wichtiges Thema.

Der Aufbau von CP/M 3 macht es hier bereits sehr einfach sog. RAMDISK's einzusetzen. Hierunter sind Pseudo-Laufwerke zu verstehen, die, mit Speicherkapazitäten bis zu rd. 800k, als sehr schnelle Arbeitsdisketten verwendet werden können. Gegenüber CP/M verhalten sie sich wie ein 'normales' Laufwerk, Ihr einziger Nachteil ist es, daß sie alle Daten beim Abschalten (oder Ausfall) des Computers verlieren - also erst Daten retten, dann Gerät abschalten. Mehr zu diesem Thema in Teil V.

Jedes CP/M System besteht prinzipiell aus 3 Programmsegmenten:

1. Dem BIOS (Basic-Input-Output-System). Hierunter ist die Hardware-Anpassung an ein vorgegebenes Computersystem zu verstehen.

   Hier werden alle notwendigen Schnittstellen zur Zeichen EIN- und AUS-Gabe initialisiert und bearbeitet, ebenso wie gewisse einfache Diskettenoperationen wie Lesen oder Schreiben eines Sektors und Laufwerk, Track und Seite ansprechen. Dieser Programmteil ist für die meisten Computer sehr unterschiedlich, aber zuständig für die Funktionsfähigkeit und Zuverlässigkeit eines Systemes. Beispiele und Hinweise zum BIOS sind in Teil IV zu finden.

2. Das BDOS (Basic Disk Operating System). Dieses Programmsegment ist das, was eigentlich unter CP/M zu verstehen ist.

Hier werden alle Diskettenoperationen veranlaßt und überwacht, ebenso wie die Zeicheneingabe und -ausgabe. Dieser Programmteil, zusammen mit dem BIOS muß als Mindestkonfiguration vorhanden sein, um CP/M-Programme funktionsfähig zu erhalten.

3. Der CCP (Console Command Prozessor) ermöglicht es bestimmte Dateien (zumindest erstmals) aufzurufen sowie Dateien zu kopieren und Disketten zu formatieren.

   Ein Teil des CCP's ist der sogenannte LOADER (Lader), der sich sofort nach dem Einsprung in den CCP automatisch vor das BDOS legt. Es ist der Loader, der die gewünschten Programme lädt und aus diesen evtl. vorhandene RSX-Elemente übernimmt.

Vor den Loader können, wie bereits angedeutet, sogenannte RSX-Dateien geladen werden (Resident System eXtension). Dies sind Programmteile, die in gewisser Weise eine Befehlserweiterung des BDOS-Segmentes zulassen.

Der Arbeitsbereich, die sogenannte TPA (Transient Program Area), ist der Arbeitsspeicher ab Adresse 100H bis zum Beginn des BDOS bzw. einem vorgelagerten RSX.

Kapitel 2 Die Speicherorganisation

CP/M 3 kann in seiner gebankten (segmentierten) Version RAM-Speicher bis zu 1MB verwalten. Notwendig zum Betrieb sind jedoch nur 2 Banks als Minimum mit jeweils 64K und einem gemeinsamen Speicherbereich von 4K bis 8K, der sog. COMMON-Area.

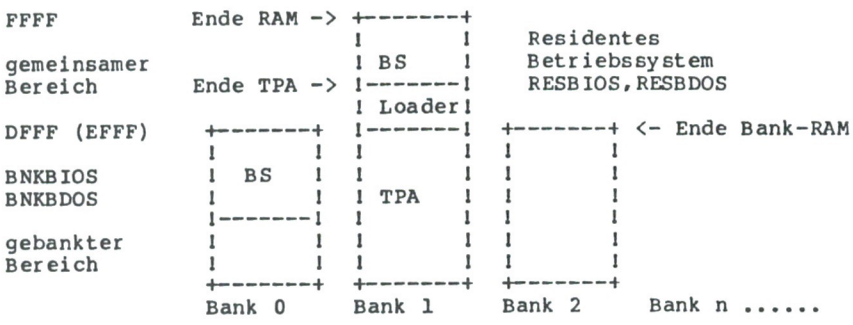

Bild 1. logische Speicheraufteilung unter CP/M 3

Es ist aus Bild 1 zu erkennen, daß Teile des Betriebssystemes (BS) auf Bank 0 und im gemeinsamen Speicherbereich zu finden sind. Der Loader ist direkt vor dem Betriebssystem im gemeinsamen Speicherbereich angesiedelt. Er kann in den gebankten Bereich der TPA (auf Bank 1) hineinreichen.

Auf Bank 0 befindet sich vor allem die (CP/M-) interne 'Verwaltung' und die Disketten- Ein/Ausgabe, während auf Bank 1 der BDOS-Einsprung und die BIOS-Sprungtabelle mit den zugehörenden Vektoren auf den Adressen 0000H und 0005H liegt, wie dies von CP/M 2 gewohnt ist.

Die TPA, d.h. der Arbeitsspeicher-Bereich in welchem die Programme ablaufen, ist in Bank 1 angesiedelt. In dieser Bank sind auch alle CP/M typischen Speicherbelegungen im Speicherbereich 0..100H.

In Bank 0 werden darüberhinaus die Puffer für das Inhaltsverzeichnis (DIR-ectory) abgelegt und, falls gewünscht, eine sog. Hash-Tabelle. (diese hat nichts mit 'Rauchen' zu tun, sondern ermöglicht es dem Betriebssystem über die in der Tabelle abgelegten Zeiger (Vektoren) schneller einen Eintrag im Inhaltsverzeichnis zu finden.)

Die Größe der Puffer für das Inhaltsverzeichnis kann (solange der Speicherplatz ausreicht) frei gewählt werden. Ideal ist die Zuweisung einer Puffergröße, die das gesamte Inhaltsverzeichnis aufnimmt. (z.B für je 32 Einträge * 32 Bytes=1024Bytes=1K). Bei einer Puffergröße von 1K und 128 Dir-Einträge entspräche dies einer Zuweisung von 128/32 = 4 Puffern).

Zuweisungen, die größer als der vom Inhaltsverzeichnis benötigte Raum ist, sind natürlich möglich aber nicht sinnvoll. Steht nicht

genügend Speicherkapazität zur Verfügung oder müssen Puffer für mehrere Laufwerke zugewiesen werden, können natürlich auch kleinere Puffer verwendet werden. Es werden dann eben nur Teile des Inhaltsverzeichnis abgelegt, was einen häufigeren Diskettenzugriff zur Folge hat. Die Mindestpuffergröße ist '1'.

Ist noch Platz in Bank 0, kann dieser mit Datenpuffer für jedes Laufwerk belegt werden.

In diesen Pufferbereich werden alle Daten, die von Diskette gelesen oder dorthin geschrieben werden, transferiert, um von hier aus schneller auf die Laufwerke zugreifen zu können, ohne erst kleine Segmente hin und her zu schieben. Die Größe dieser Puffer sollte möglichst der Blockgröße entsprechen, zumindest der Sektorgröße. Stehen mehr als 2 Banken zur Verfügung, werden Datenpuffer, die nicht mehr in Bank 0 passen, auf diese verteilt.

Die Datenpuffer dienen dem (internen) blocking- deblocking, das von CP/M 2 noch vom BIOS übernommen werden musste.

Zugegebenermaßen klingt dies alles etwas kompliziert, vor allem wenn man bedenkt, daß noch andere Puffer zugewiesen werden müssen, aber keine Angst: die ganze Zuweisung übernimmt das Generierprogramm GENCPM. Näheres hierzu in einem späteren Kapitel (Hardware-Anpassung).

Aus der Speicheraufteilung und dem bisher beschriebenen, kann erkannt werden, daß die BIOS-Einsprungtabelle zwar im gemeinsamen Speicherteil liegt, aber alle Unterprogramme die mit dem Zugriff auf Disketten zu tun haben liegen in Bank 0.

Nun zeigt zwar die BIOS-Sprungtabelle auf die richtige Adresse, aus einem TPA-Programm heraus gesehen ist aber ein Zugriff auf die Adresse der einzelnen Unterprogramme nicht (so ohne weiteres) möglich, da die TPA ja in Bank 1 liegt.

Hierin ist der Grund zu suchen warum einige Hilfsprogramme, die für CP/M 2 geschrieben wurden und gewisse Disketten-Operationen ausführen konnten, unter CP/M 3 nicht mehr funktionieren; diese Programme wurden alle mit direkten Aufrufen in das BIOS geschrieben. Zu diesen Programmen gehören Disketten'reparierer' wie DU, DISKDOCTOR, POWER und SUPERZAP. Fast alle diese Programme sind heute in neuen Versionen erhältlich, die entweder nur unter CP/M3 oder auch unter den alten Versionen funktionsfähig sind.

CP/M 3 besteht wie CP/M 2 aus mehreren Programmteilen, die nach einem ganz bestimmten Schema in Bank 1, der TPA-Bank zusammengesetzt werden. Immer resident sind die Programmteile BIOS und BDOS. Sie bilden den Kern des Betriebssystemes. Zugriff auf diese beiden Programmteile erhält der Benutzer über Einsprungadressen auf Adresse 0000H, auf die Warmstartadresse im BIOS und über die Einsprungadresse 0005H auf das BDOS, jeweils aus dem TPA-Bereich heraus.

Das BIOS (Basic Input/Output System) ist der hardware-abhängige Teil des Betriebssystemes, während das BDOS (Basic Disk Operating System) die gesamte Diskettenverwaltung (unveränderbar) übernimmt.

Das BIOS wird normalerweise vom Systemlieferanten angepasst und kann vom Kunden modifiziert und erweitert werden. Das BDOS ist

der invariable Teil des Betriebsystemes wie er von Digital Research INC, USA geliefert wird. Dieser Teil des Betriebssystemes wird in Form der Dateien BNKBDOS3.SPR und RESBDOS3.SPR zur Verfügung gestellt, im Falle eines ungebankten Systemes als Datei BDOS.SPR. (Siehe hierzu Bild 2).

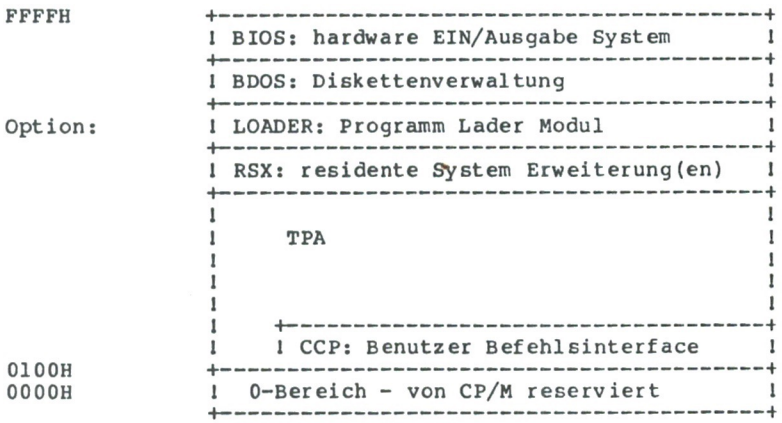

Bild 2 Belegung von Bank 1

Der LOADER und die/das RSX-Modul(e) sind CP/M3 typische Erweiterungen des Betriebssystemes. Mit diesen Modulen kann CP/M auf Benutzerebene ergänzt, erweitert oder verändert werden.

Aufgabe des Loaders ist es, TPA-Programme und RSX-Module zu laden, wobei TPA-Programme immer nach Adresse 100H geladen werden, während dies bei RSX-Programmen etwas komplizierter wird.

Bei RSX-Programmen wird der RSX-Teil direkt vor den Loader 'geladen'; der BDOS-Einsprung unter Adresse 0005H wird auf die RSX-Startadresse umgeändert und das zum RSX gehörende Programm wird 'ganz normal' nach 100H geladen. Jeder BDOS-Aufruf wird danach zuerst über das RSX-Modul geleitet bevor er den BDOS-Einsprung erreicht, er kann also vorher in jeder beliebigen Weise 'behandelt' werden.

Der Dialog Benutzer-CP/M erfolgt über das Programm-Modul CCP. Dieses Modul ist unter CP/M 3 nicht mehr Teil des Betriebssystemes, sondern ein TPA-Programm mit Startadresse 100H wie dies für TPA-Programme üblich ist.

In der gebankten Version wird bei jedem Warmstart der CCP-Programmteil jedoch nicht von der Diskette nachgeladen, sondern aus einer Bank, in welche der CCP vom BIOS bereits beim Kaltstart transferiert wurde. Dies hat den Vorteil, das nach dem Booten auch Disketten ohne Betriebssystem bearbeitet werden können.

Auf welcher Bank und Adresse der CCP zwischengespeichert wird, bestimmt der Systemprogrammierer. Näheres ist normalerweise aus der BIOS-Datei BOOT zu entnehmen. Im Übrigen kann dies dem Benutzer auch recht einerlei sein, da er auf den entsprechenden

Speicherbereich nicht so ohne weiters Zugriff hat. Für ihn von Bedeutung ist nur der schnelle Zugriff auf den CCP ohne Laufwerksaktivitäten.

Vergleicht man, auf das Banking bezogen CP/M 3 mit CP/M 2, so muß man klar erkennen, daß sich alles 'Wesentliche' bei CP/M 3 auf Bank 1 abspielt, denn dort ist die TPA, wärend bei CP/M 2 die TPA in Bank 0 war.

Dies spielt normalerweise für den Anwender keine Rolle, wenn aber ein neues BIOS installiert werden soll, so ist dieser Punkt doch zu berücksichtigen, wenn z.B. Unterprogramme aus dem Monitor heraus getestet (entwanzt) werden müssen.

Sollte dennoch einmal das Problem auftreten, Programme oder Programmteile von Bank 0 nach Bank 1 zu transferieren, so kann hier ein gutes Monitorprogramm sicher sehr hilfreich sein. Für den Transfer bietet sich, falls das Monitorprogramm dies nicht direkt übernehmen kann, immer der gemeinsame Speicherbereich an.

Kapitel 3 Disketten- und Laufwerksorganisation

CP/M3 kann bis zu 16 Laufwerke mit bis zu 512M-Byte Kapazität unterstützen. Jedes Laufwerk wird durch einen Buchstaben im Bereich A..P identifiziert. 'Laufwerk' kann prinzipiell jedes Floppy-Disk Laufwerk sein, jedoch auch Harddisks und Ramdisks und notfalls auch Kassettengeräte, wobei hier der Zugriff recht langsam werden kann.

Die Aufteilung auf einer Diskette ist im allgemeinen von aussen nach innen.

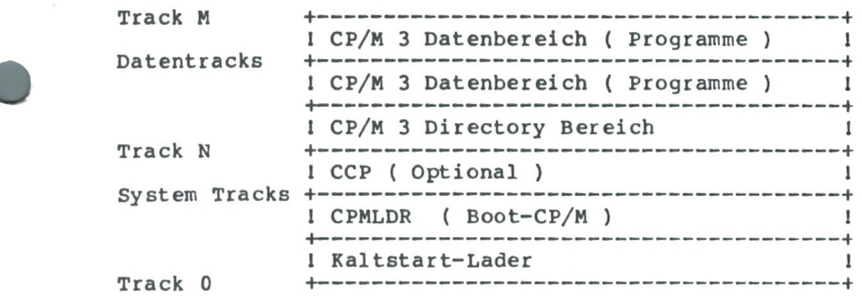

Bild 3 Aufteilung von Disketten

Wie in Bild 3 gezeigt, stellen die ersten N Tracks die sogenannten Systemspuren dar. Sie werden nur auf Disketten benötigt, welche CP/M 3 booten sollen. Hierunter ist das Laden des CP/M-Betriebssystemes zu verstehen. Die Anzahl der belegten Systemspuren ist unter CP/M 3 recht unterschiedlich, meist sind sie überbelegt (genauer gesagt leer), um mit dem Format der älteren CP/M 2 Disketten kompatibel zu bleiben.

Der Loader benötigt kaum mehr als 4..5K Speicherplatz, die restlichen Sektoren bleiben leer.

Das Laden des Betriebssystemes selbt geschieht in mehreren Schritten:

1. Nach dem Einschalten (oder nach Reset) meldet sich das Monitorprogramm als unterste Ebene.

2. Manuell kann nun gebootet werden (Siehe hierzu Monitor-Handbuch oder sonstige 'Kaltstartvorschriften' des benutzen Gerätes).

3. Der Monitorboot lädt seinerseits den Kaltstart-Lader von Track 0, Sektor 1 auf der boot-Diskette.Der Kaltstart-Lader wird nach Adresse 0, Bank 0 geladen.

4. Der Kaltstart-Lader lädt nun den CPMLDR. Dies ist ein 'abgemagertes' CP/M mit Sonderfunktionen und einem nur auf 'Lesen' und Zeichenausgabe geschrumpften BIOS .

5. Der CPMLDR lädt nun seinerseits die Programmteile CPM3.SYS - dies ist das eigentliche Betriebssytem - und verteilt

dieses auf die Systembank 0 und den gemeinsamen Speicherbereich, aus welchem auf alle Banken zugegriffen werden kann.

6. Ist die Verteilung erfolgt, wird der an das Betriebssystem angelinkte Programmteil BIOS angesprungen.

7. Das Programmsegment BOOT (im BIOS) lädt nun das TPA-Programm CCP.COM (auch nach Bank 0 zur Wiederverwendung)

8. Nachdem BIOS alle Parameter initialisiert hat, wird CCP exekutiert --- CP/M 3 ist aktiv .

9. Ist auf der Diskette eine Datei PROFILE.SUB vorhanden, so wird deren Befehlsinhalt vom CCP ebenfalls noch abgearbeitet, bevor sich das System mit dem gewohnten Prompter 'A>' meldet.

Dieser Boot-Vorgang ist üblicherweise nur nach dem Einschalten des Gerätes notwendig und nur beim Booten müssen die Systemtracks den Kaltstart-Lader und CPMLDR und die Datentracks die Programme CPM3.SYS (mit angepaßtem BIOS) und CCP.COM enthalten.

Mit PROFILE.SUB können beim booten verschiedene Systemparameter gesetzt werden. Typische Befehle sind Datumabfrage, Zuweisungen von Zusatzgeräten und die Reihenfolge in der die Laufwerke nach Programmen abgefragt werden sollen. (Also ein Aufruf der TPA-Befehle SETDEV, SET und DATE)

Nach dem boot-Vorgang kann die Boot-Diskette entfernt werden und eine andere (formatierte) Diskette eingefügt werden.

Das von CP/M 2 'gewohnte' CNTRL-C hat nach wie vor die gleiche Funktion, es erfolgt jedoch kein Diskettenzugriff - lediglich der Befehlsinterpreter CCP.COM wird aus einer Bank nachgeladen.

Kapitel 4 Zeiger im TPA-Bereich

Ebenso wie unter CP/M 2 sind verschiedene Speicherbereiche für das Betriebssystem reserviert, über die CP/M mit dem CCP und den von diesem aufgerufenen Programmen verkehrt. Solche 'Software-Schnittstellen' sind notwendig, da jedes CP/M-Betriebssystem je nach Konfiguration unterschiedliche Startadressen haben kann.

Unter CP/M 3 gibt es vor allem 2 Bereiche, die der gegenseitigen Verständigung dienen.

Allen voran - und CP/M 2 kompatibel - der Speicherbereich von 0...100H in Bank 1, der TPA-Bank. Die einzelnen Adressen haben dabei folgende Bedeutung:

0000H Einsprungadresse in die Warmstartroutine des BIOS.

Hier ist ein direkt ausführbarer Sprung implementiert, der aus jedem Programm heraus aufgerufen werden kann. Wird der Sprung ausgeführt, wird der CCP nachgeladen und angesprungen. Das System meldet sich mit dem bekannten Prompter '>'.

0001H Hier steht der Zeiger auf die Startadresse des BIOS (genauer gesagt auf den Einsprung WBOOT)

Diese Adresse wird vor allem zur Berechnung von direkten BIOS-Einsprüngen benutzt.

0005H Einsprungadresse in das BDOS oder einer vorgelagerten RSX-Datei, aber auch SID & Co.

Dies ist die wichtigste Adresse in jedem CP/M-System, denn über diese Adresse laufen ALLE Funktionen, selbst die Ein- oder Ausgabe der Konsole.

0006H Hier steht die Startadresse des BDOS oder einer vorgelagerten RSX-Datei.

Diese Adresse ist gleichbedeutend mit der obersten, frei zur Verfügung stehenden Speicherstelle der TPA.

0038H Diese Adresse (RST8) wird unter gewissen Voraussetzungen von SID & Co benötigt,wenn Haltepunkte beim Debuggen verwendet werden.

Wird in der Befehlszeile des CCP (oder mit dem I-Befehl in SID) ein Programm mit Optionen oder Zusatzprogrammnamen aufgerufen, so werden diese Daten auf die nachstehend genannten Adressen, wie dort angegeben, verteilt. Jedes Nachfolgeprogramm kann diese Daten aus dem entsprechenden Speicherbereich nutzen. (Siehe hierzu auch Teil V)

0050H Laufwerksbezeichnung von welchem die Datei geladen wurde. (01=A,02=B usw. 00 bedeutet, daß keine besondere Laufwerksangabe gemacht wurde, in diesem Falle ist immer das benutzte Laufwerk gemeint. (Das Laufwerk von dem aus ein Programm aufgerufen wurde).

0051H Paßwortadresse der 1. Datei in der Befehlsfolge.

0053H Paßwortlänge der 1. Datei.

0054H Paßwortadresse der 2. Datei in der Befehlsfolge.

0056H Paßwortlänge der 2. Datei in der Befehlsfolge.

005CH FCB (Dateikontrollblock) der 1. Datei, beginnend mit der Laufwerksangabe (00, 01 usw).

005DH Dateiname in Großbuchstaben auf 8 Stellen aufgefüllt mit Leerzeichen.

0065H Index des Dateinamen in Großbuchstaben, auf 3 Stellen aufgefüllt mit Leerzeichen.

006CH FCB der 2. Datei, beginnend mit der Laufwerksangabe.

006DH Dateiname der 2. Datei in Großbuchstaben, auf 8 Stellen aufgefüllt mit Leerzeichen.

0075H Index des Dateinamen in Großbuchstaben, auf 3 Stellen aufgefüllt mit Leerzeichen.

Adresse 80H wird vom CCP als DMA-Puffer (mit 128 Byte Größe) vorgegeben.

0080H Hier steht die Länge der eingegeben Befehlszeile.

0081H Hier beginnt der komplette Text der Befehlszeile jedoch in Groß- und Kleinschreibung, so wie vom Benutzer eingegeben.

Unter CP/M 3 steht ein weiterer Block mit Daten zur Verfügung, auf welchen alle Programme direkt zugreifen können: der SCB (System Controll Block). Der Zugriff kann jedoch nur über einen BDOS-Aufruf erfolgen. Die Daten, die im SCB abgelegt sind, können aus dem in Teil IV abgedruckten Listing entnommen werden.

Der SCB liegt im gemeinsamen Speicherbereich und dient dem Datenaustausch von CCP, BDOS und BIOS, kann aber auch nützliche Daten für ein TPA-Programm enthalten.

Kapitel 5 Die Dateiorganisation

Alle TPA-Programme leiten ihre Dateienbearbeitung und ihre Ein-Ausgabe über sogenannte BDOS-Calls. Die Konventionen hierzu sind die gleichen wie sie bereits von CP/M 1 und CP/M 2 her bekannt sind.

Programme, die nicht alle Ein/Ausgabe-Funktionen von BDOS-Calls ableiten, sondern Ports zur Diskettenverwaltung oder Ein/Ausgabe direkt ansprechen, sind nicht CP/M kompatibel. Hier ist unter kompatibel zu verstehen, daß ein Programm, falls es auf ein passendes Diskettenformat überspielt ist, auf jedem CP/M-Computer funktionieren soll(te). Einzige Ausname kann u.U. die Konsolenausgabe sein, soweit es sich um z.B. Cursoradressierung handelt oder um Attribute wie Blinken, Halbhell usw. Dies Vorbedingungen müssen jedoch spezifiziert sein und u.U. anpaßbar an unterschiedliche Ausgabegeräte.

Es ist gerade diese verlangte Kompatibilität, die es nicht gestattet spezielle Grafiktreiber einzubinden und damit 'Bildschirmspielchen', wie diese bei anderen Computern möglich sind, allgemeingültig einzubinden.

Die Verwendung von BIOS-Aufrufen unter Umgehung des BDOS kann, wie der Übergang von CP/M 2 auf CP/M 3 zeigt, ebenfalls Inkompatibilitäten mit sich bringen, wobei unter CP/M 3 hierzu keinerlei Notwendigkeit mehr besteht diesen Weg zu beschreiten. (Siehe hierzu BDOS-Funktion 50 in Teil III).

Alle Daten, die mit der Dateienverwaltung auf einer Diskette zu tun haben, werden über einen sogenannten File Control Block (FCB) verwaltet. Diese Datenstruktur erlaubt es CP/M, in einem 32 Bytes großen Datenblock alle Informationen über eine vorhandene oder neu zu eröffnende Datei zu erhalten. Der Aufbau dieses Datenblocks ist wie folgt:

```
  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
!dr!f1!f2!f3!f4!f5!f6!f7!f8!t1!t2!t3!ex!s1!s2!rc!
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
!d0!d1!d2!d3!d4!d5!d6!d7!d8!d9!dA!dB!dC!sD!dE!dF!
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

+--+--+--+--+
!cr!r0!r1!r2!
+--+--+--+--+
 32 33 34 35
```

Mit folgender Bedeutung:

dr — Laufwerkscode 0..16 entsprechend Laufwerk A..P bzw. 0 für das eingeloggte (default) Laufwerk,

f1..f8 — enthält den Dateinamen in ASCII mit Bit 7=0. Der Dateinamen muß in f1 beginnen, verbleibender Raum, falls der Dateinamen kürzer als 8 Zeichen ist muß mit Blanks (20H) gefüllt werden. Das 8. Bit jedes Zeichens kann von CP/M gesetzt sein und enthält dann Datei-Attribute. (siehe BDOS-Funktion 30)

t1..t3 Enthält den Index (TYP-Name) einer Datei in ASCII mit Bit 7=0. Ist Bit 7 gesetzt zeigt dies auf Datei-Attribute dabei gilt:

t1	READ-ONLY Datei (nur lesen)
t2	System-Datei (nur mit DIRS aufzeigen)
t3	Archiv-flag

ex Enthält das EXtend-Flag. Vom aufrufenden Programm üblicherweise auf 0 gesetzt. Der EXtend wird vom BDOS gesetzt, wenn eine Datei nicht in einen einzelnen FCB paßt. Jeder zusätzliche Extent erhält dann einen eiggenen, im übrigen völlig gleichen FCB zugewiesen.

s1,s2 CP/M reserviert zum Systemgebrauch, wird üblicherweise auf 0 gesetzt.

rc Zähler der 128-Bytes Segmente, in welchen CP/M die Dateien unterteilt. RC gibt also die Dateilänge in logischen Sektoren an.

d0..dF CP/M reserviert zum Systemgebrauch. Hier werden verschlüsselt die Sektor- und Tracknummern abgelegt,unter welchen die Datei abgelegt ist.

Der Teil wie bisher beschrieben wird im Inhaltsverzeichnis von CP/M (der DIRectory) abgelegt. Die weiteren FCB-Plätze sind nur innerhalb des Disketten-Bearbeitungsprogramm von Interesse.

cr Gibt die z.Z. zu lesenende- oder schreibende Segmentnummer an. Muß vom Programm auf 0 gesetzt werden, wenn eine Datei eröffnet wird.

r0..r2 optionale RANDOM-Segment Nummer im Bereich 0-3FFFFFH (0..262143). Dies entspricht einem 18 Bit Wert wobei r0 das LOW-Byte und r2 das HIGH-Byte repräsentiert.

Dieser FCB ist im übrigen für jede dort aufgenommene Datei identisch mit dem DIR-ectory-Eintrag, soweit es die Bytes 0..15 betrifft. In Teil V ist zu diesem Thema noch mehr zu finden.

TEIL III Programme und Programmieren unter CP/M 3

Kapitel 1 Einführung in die Vereinbarungen

Alle Programme kommunizieren mit CP/M über zwei Software-Schnittstellen, den Adressen 0000H und 0005H in der TPA. Über Adresse 0 wird ein Rücksprung auf Systemebene veranlasst (hier wird lediglich der CCP nachgeladen und als Programm ausgeführt), während über Adresse 0005H alle Funktionen der Kommunikation, wie sie ein Programm normalerweise benötigt, erledigt werden.

Jeder Aufruf der Adresse 0005h, ein sog. BDOS-call oder BDOS-Aufruf, benötigt bestimmte Konventionen. Hierfür sind zwei CPU-Register zur Datenübergabe vorgesehen.

Vor jedem BDOS-Aufruf müssen die nachfolgenden Register mit den genannten Daten geladen werden:

In Register C	wird eine Funktionsnummer übergeben. Entsprechend dieser Funktion weiß das BDOS was zu tun ist.
In Register DE	werden je nach Funktion Zeiger oder Daten übergeben, die das BDOS u.U. benötigt. Byte-Werte werden in Register <E> übergeben, Adressen und Vektoren im Doppelregister <DE>.

Hat das BDOS die gewünschte Funktion abgearbeitet, werden, falls dies notwendig ist, entsprechende Daten in den folgenden Registern zurückgegeben:

In Register A	Bytes aus Eingabefunktionen oder Fehlermeldungen.
In Register HL	Zeiger auf Datenfelder oder Fehlermeldungen

Die Register C und DE enthalten bei der Rückkehr ungültige Daten (außer bei Funktion 50). Auf diese Inhaltsveränderung der Register A(F), HL, DE und BC ist in jedem Falle zu achten, falls dies entsprechenden Registerinhalte bei einem BDOS-Aufruf nicht verändert werden dürfen.

Ein BDOS-Aufruf könnte innerhalb eines (Assembler-)Programmes also etwa wie folgt aussehen:

```
 1 0001' 0E 0F              ld      c,15          ; Funktion Datei eroeffnen
 2 0003' 11 000F'           ld      de,fcb        ; File-control-Block
 3 0006' CD 0005            call    0x05H         ; BDOS Aufruf
 4 0009' B7                 or      a             ; Fehler ?
 5 000A' 28 03              jr      z,next        ; wenn nicht
 6 000C' 00                 nop                   ; Fehlerbehandlung
 8 000D' 00           next: nop                   ; hier gehts weiter
 9 000E' 00                 ret
10 000F  00 00 00 00 fdc:   ds      32
```

Diese Vereinbarung macht es jedem Programm, welches Register direkt bearbeiten kann, einfach mit CP/M zu verkehren. Hochspachen-Programme wie BASIC, PASCAL oder FORTRAN haben entsprechende Treiber eingebunden. Der Benutzer braucht sich dort nicht mehr um obige Konventionen zu kümmern.

In den nachfolgenden Kapitel werden alle möglichen BDOS-Aufrufe zusammengefaßt. Beispiel der Anwendung werden im Kapitel 4 dieses Abschnittes in Form von Listings gezeigt.

Kapitel 2 Die CP/M 2 kompatiblen BDOS-Aufrufe

Die nachfolgenden Funktionen sind mit Ausnahme der Funktionen 7 und 8 kompatibel zu den Funktionsaufrufen unter CP/M 2. Sie können also in allen Programmen verwendet werden, die unter beiden CP/M-Versionen störungsfrei funktionieren sollen.

Einige Funktionen wie z.B. Funktion 9, 'können' unter CP/M 3 mehr, als dies unter CP/M 2 der Fall war; dies bedeutet in diesem Zusammenhang jedoch nur, daß bestimmte neue 'features' unter CP/M 2 nicht vorhanden sind, die Funktion selbst bleibt davon unberührt.

Im umgekehrten Falle, CP/M 2 Programme unter CP/M 3, sind normalerweise ebenfalls keine Störungen zu befürchten.

Es gibt hierbei jedoch zwei Ausnahmen: die Funktionen 7 und 8. Unter CP/M 2 wird mit dieser Funktion das IO-Byte gelesen bzw. verändert. Mit dem IO-Byte konnte, ähnlich wie mit DEVICE, unter gewissen Umständen eine Art Umlenkung (redirection) der Ausgabe-Kanäle vorgenommen werden.

Diese Umlenkung war prinzipiell von CP/M vordefiniert. Sie wurde jedoch immer vom BIOS bestimmt. In vielen Implementationen wurde das IO-Byte überhaupt nicht berücksichtigt.

Sollten CP/M-2 Programme unter CP/M 3 durch die IO-Byte-Funktion Schwierigkeiten bereiten, so empfiehlt es sich u.U., diese Funktion durch ein RSX-Programm abzufangen und entsprechend den Gegebenheiten anzupassen.

Beim Ablauf von CP/M 2-Programmen unter CP/M 3, ist auf jeden Fall, beim Auftreten von Fehlern, zu prüfen, inwieweit u.U. direkte Aufrufe des BIOS zur Diskettenbearbeitung die Fehlerquelle sein können. Sollte dies der Fall sein, so kann evtl. das am Ende dieses Teiles abgedruckte RSX-Programm RSX22 Abhilfe schaffen.

Nachstehend folgt eine Zusammenstellung aller CP/M-2 kompatiblen BDOS-Funktionen.

Funktion 0	SYS-Reset
Eingabe	C=00H

Mit dieser Funktion wird ein Warmstart von CP/M veranlaßt.

Diese Funktion hat die gleiche Wirkung wie ein Sprung nach Adresse 0. Nach diesem Funktionsaufruf wird der CCP aus dem Bankbereich neu geladen und exekutiert. Er meldet sich mit dem entsprechenden Prompter.

Funktion 1	CONSOL Eingabe
Eingabe	C=01H
Zurück	A=ASCII Eingabe

Diese Funktion liest ein Zeichen von der Eingabe-Konsole.

Alle darstellbaren Zeichen sowie CR, LF und BS werden auf die Ausgabe-Konsole ausgegeben. TAB's werden auf jede 8. Spalte expandiert.

Es wird auf CNTRL-S als Haltesignal und CNTRL-Q als 'Weiter'-Signal reagiert. Mit CNTRL-P kann der Drucker zu- oder abgeschaltet werden.

Der Funktionsaufruf kommt nicht zurück, bevor ein Zeichen eingegeben wurde.

Funktion 2	CONSOL Ausgabe
Eingabe	C=02H
	E=ASCII-Zeichen

Das Zeichen in Register <E> wird auf die Konsole ausgegeben.

Wie unter Funktion 1 werden TAB's expandiert und CNTRL-S und CNTRL-Q sowie CNTRL-P bearbeitet.

Funktion 3	AUX Eingabe
Eingabe	C=03H
Zurück	A=Eingabe-Zeichen

Diese Funktion liest ein Zeichen vom AUX-Port.

Kontrollzeichen erhalten keine 'Sonderbehandlung', sondern werden ausgegeben wie sie sind.

Funktion 4	AUX Ausgabe
Eingabe	C=04H
	E=Ausgabe-Zeichen

Diese Funktion gibt das Zeichen in Register <E> auf den AUX-Port aus.

Kontrollzeichen erhalten keine 'Sonderbehandlung', sondern werden ausgegeben wie sie sind.

Funktion 5	PRINT Ausgabe
Eingabe	C=05H
	E=ASCII-Zeichen

Das Zeichen in Register <E> wird auf den Drucker ausgegeben.

Das 8. Bit wird von dieser Funktion nicht maskiert. Es können also auch Grafik-Zeichen ausgegeben werden, falls dies vom BIOS zugelassen wird.

Funktion 6	Konsolen Ein-/Ausgabe direkt
Eingabe	C=06H
	E=0FFH Eingabe/Status
	E=0FEH Status
	E=0FDH Eingabe
	E=ASCII Ausgabe
Zurück	A=ASCII-Zeichen
	oder A=00/FF für Status

Mit dieser Funktion können Zeichen in Register <E> unter Umgehung aller Prüfungen (wie dies in Funktion 1 und Funktion 2 der Fall ist) direkt auf die Konsole ausgegeben werden.

Ist Register <E>=FF, wird die Konsolen abgefragt ob ein Zeichen (von der Tastatur) vorliegt. Liegt ein Zeichen an, wird dieses in <A> zurückgegeben. Liegt kein Zeichen vor, wird A=00 zurückgegeben.

Ist Register <E>=FE, wird nur der Status überprüft. Liegt ein Zeichen vor, wird A=FF zurückgegeben andernfalls wird A=00.

Ist Register <E>=FD, wird auf eine Eingabe gewartet. Die Funktion kehrt in diesem Falle nicht vor Eingabe eines Zeichens zurück. Enthält Register <E> ein ASCII-Zeichen, wird dieses an die Konsole ausgegeben wie es ist.

Ob das 8. Bit maskiert ist oder nicht, hängt von der Implementation des Treibers im BIOS ab.

Funktion 7	AUX Eingabestatus
Eingabe	C=07H
Zurück	A=Status

Diese Funktion prüft den aktuellen Eingabestatus des AUX-Kanales.

Liegt eine Eingabe vor, wird Register A=00, liegt keine Eingabe vor wird, A=FF, zurückgegeben.

Unter früheren Versionen von CP/M wurde diese Funktion genutzt, um das IO-Byte zu lesen.

Funktion 8	AUX-Ausgabestatus
Eingabe	C=08H
	A=Status

Diese Funktion prüft den aktuellen Ausgabestatus des AUX-Kanales.

Liegt eine Eingabe vor, wird Register A=00, liegt keine Eingabe vor, wird A=FF zurückgegeben.

Unter früheren Versionen von CP/M wurde diese Funktion genutzt, um das IO-Byte zu setzten.

Funktion 9	Textausgabe
Eingabe	C=09H
	DE=Textstart

Mit dieser Funktion können Texte beliebiger Länge ausgegeben werden.

Textabschluß muß ein '$'-Zeichen sein (dieses kann mit dieser Funktion NICHT ausgegeben werden). TAB's werden expandiert, ebenso wird auf CNTRL-S (Stop), CNTRL-Q (Weiter) und CNTRL-P (Drucken) geprüft. Nach CNTRL-Q kann mit CNTRL-C die Ausgabe abgebrochen werden.

Mit Funktion 110 (nur unter CP/M 3) kann das Textende-Zeichen auf jedes beliebige andere ASCII-Zeichen umdefiniert werden, falls das '$'-Zeichen innerhalb eines Textes benötigt wird. Es ist jedoch zu berücksichtigen, daß diese Umdefinition sicherheitshalber vor jeder entsprechenden Anwendung erfolgen sollte, da der CCP bei jedem Warmstart die ursprüngliche '$'-Installation wieder einführt.

Das Textende-Zeichen wird nicht ausgegeben.

Funktion 10	Texteingabe
Eingabe	C=0AH
	DE=Pufferadresse

Diese Funktion schreibt die Zeichen, die über die Konsole eingegeben werden, in einen Puffer.

Die Startadresse des Puffers wird im Doppelregister <DE> übergeben. Die Texteingabe wird durch CR abgeschlossen oder falls die maximale Vorgabelänge erreicht wurde.

Der Puffer muß wie folgt aufgebaut sein:

```
DE  +0  +1  +2  +3  .... +n
    mx  ct  C1  C2       Cn
```

wobei MX die Pufferlänge (in HEX- maximal 255 Bytes) angibt, CT enthält nach Rückkehr der Funktion die Anzahl der wirklich eingegebenen Zeichen, während C1..CN die eingegebenen Zeichen sind. CR und LF werden nicht in den Puffer übernommen.

Bei der Eingabe werden folgende Steuerzeichen erkannt:

CNTRL-A bewegt den Cursor ein Zeichen nach links bis zum '>', ohne das entsprechende Zeichen zu löschen.

CNTRL-B bewegt den Cursor an die Position direkt nach dem '>', ohne den Text zu verändern. Ist der Cursor bereits an dieser Stelle, wird er zum Textende bewegt.

CNTRL-C Veranlaßt einen Warmstart, wenn es als erstes Zeichen eingegeben wird.

CNTRL-E erzwingt ein CR ohne dabei die Befehlszeile logisch abzuschliessen. Dies ist von Bedeutung, wenn man lange Befehlssequenzen eingeben muß, aber eine bessere Übersicht auf dem Bildschirm behalten möchte.

CNTRL-F bewegt den Cursor ein Zeichen nach rechts, ohne das entsprechende Zeichen zu löschen.

CNTRL-G löscht das Zeichen unter dem Cursor und rückt den Text rechts davon nach links auf, der Cursor bleibt jedoch an seiner augenblicklichen Position.

CNTRL-H oder BS (backspace) löscht ein Zeichen und bewegt den Cursor eine Position nach links. Die gleiche Funktion ist u.U. auch mit der DEL(ete)-Taste möglich. Dies ist abhängig davon, ob dies mit GENCPM entsprechend deklariert wurde.

CNTRL-I oder TAB, bewegt den Cursor um eine TAB-Position nach rechts. TAB-Position ist jede 8. Spalte.

CNTRL-J oder LF (linefeed) signalisiert das Befehlsende. Es hat den gleichen Effekt wie CNTRL-M. Dieses Zeichen kann an jeder beliebigen Cursorposition eingeben werden und veranlaßt dann die Abarbeitung des kompletten

Befehls in der Befehlszeile. D.h. auch der Text hinter diesem Zeichen wird abgearbeitet.

CNTRL-K löscht die Restzeile ab der Cursorposition.

CNTRL-M oder CR siehe CNTRL-J.

CNTRL-R schreibt den bisher editierte Text (der besseren Übersicht halber) mit allen vorgenommenen Korrekturen noch einmal.

CNTRL-U löscht den Zeileninhalt und rückt eine Zeile tiefer.

CNTRL-W bringt die 'alte' Befehlszeile zurück in den Zeilenpuffer zum Editieren. So können von CP/M reklamierte Fehleingaben einfach korrigiert werden.

CNTRL-X löscht die Befehlszeile und setzt den Cursor direkt nach dem '>'.

Dem Benutzer stehen, falls diese Funktion aufgerufen wird, alle Vorteile des in CP/M 3 eingebauten Zeileneditors zur Verfügung.

Anmerkung:

Unter CP/M 2 ist nur ein Teil der genannten Kontroll-Sequenzen funktionsfähig. Wird versucht die entsprechende Sequenzen doch zu benutzen, wird das CNTRL-Zeichen mit vorangestelltem Aufwärtspfeil (z.B. ^W) ausgegeben.

Funktion 11	Konsolen-Status
Eingabe	C=0BH
Zurück	A=00 oder 01H

Diese Funktion gibt den Status der Eingabe-Konsole zurück.

Liegt eine Eingabe vor, dann ist Register A=01H. Liegt keine Eingabe vor, dann ist Register A=00.

Mit Funktion 109 (nur unter CP/M 3) kann die Statusabfrage in soweit geändert werden, daß nur nach Eingabe von CNTRL-C das Register A=01H zurückgibt.

Anmerkung:

Diese Funktion ist nicht CP/M 2 kompatibel, dort wird bei Vorliegen eines Zeichens (beliebiger Art) in Register A=FF zuückgegeben.

Die Abfrage sollte hier in jedem Falle nach A=00 oder Bit 0=1 erfolgen, keinesfalls auf A=FF wenn die Funktion unter CP/M 2 und CP/M 3 benutzt werden soll.

Funktion 12	Versionsnummer
Eingabe	C=0CH
Zurück	HL=Versionsnummer

Diese Funktion gibt die (CP/M)-Versionsnummer zurück.

Es wird ein 2-Byte-Wert im Doppelregister <HL> zurückgegeben, wobei H=00 ist (für CP/M im Unterschied zu MP/M) und L=31H (oder größer).

Funktion 13	Reset Diskettensystem
Eingabe	C=0DH

Diese Funktion setzt alle Diskettenparameter zurück (Siehe auch Funktionen 28 und 29).

Als Vorgabelaufwerk wird Laufwerk A angesprochen und als Vorgabe-DMA-Adresse wird Adresse 80H gesetzt. Mit dieser Funktion kann der gleiche Zustand erreicht werden, wie er direkt nach dem Booten herrscht.

Die Datei PROFILE.SUB wird jedoch nach diesem Software-Reset nicht aufgerufen, ebensowenig wie der CCP.

Funktion 14	Ansprechen eines Laufwerkes
Eingabe	C=0EH
	E=Laufwerksnummer
Zurück	A=Fehlerflag
	H=physikalische Fehlernummer

Diese Funktion versucht das in Register <E> angegebene Laufwerk anzusprechen.

Die Laufwerknummer entspricht einer HEX-Zahl von 00=A, 01=B, 02=C usw. Es können maximal 16 Laufwerke angesprochen werden.

Bei Rückkehr der Funktion ist Register A=0, wenn das Laufwerk logisch und physikalisch vorhanden ist. Es kann hierzu notwendig sein, daß auch eine Diskette im Laufwerk steckt und die Klappe geschlossen ist.

Im Fehlerfalle ist Register A=FF. In diesem Falle sind nähere Angaben in Register H zu finden.

Ist H=01 kann auf das Laufwerk nicht physikalisch zugegriffen werden. Es kann sich hierbei um eine fehlende oder falsche Diskette handeln.

Ist H=04 ist das Laufwerk logisch nicht vorhanden, d.h. es ist im BIOS gar nicht eingebunden (vorgesehen).

Funktion 15	Eröffnen einer Datei
Eingabe	C=0FH
	DE=FCB-Adresse
Zurück	A=Fehlercode
	H=Fehlernummer

Diese Funktion aktiviert den FCB einer Datei, die im Inhaltsverzeichnis bereits unter dem angesprochenen Laufwerk und der gültigen USER-Ebene vorhanden ist.

In Byte 0 des FCB wird die Laufwerkangabe erwartet, gefolgt von Dateiname und Index. Byte 12 des FCB gibt einen möglichen EXtend an, wird jedoch normalerweise auf 0 gesetzt.

Wenn die Datei durch ein Paßwort im Lesemodus geschützt ist, muß dieses Paßwort in die ersten 8 Bytes der momentan gültigen DMA geschrieben werden, oder das Paßwort muß vorher als 'default'-Paßwort gesetzt worden sein. (Siehe hierzu den SET-Befehl).

Wird im FCB das CR-Feld auf FF gesetzt, gibt die Funktion im CR-Feld den Bytezähler für den letzten (logischen) Sektor zurück. (Siehe Funktion 30 hierzu).

Wird eine Datei sequentiell gelesen oder geschrieben, muß das CR-Feld auf 00 gesetzt sein, wenn von Beginn an gelesen werden soll.

Wird eine Datei in einer USER-Ebene eröffnet, so wird in dieser USER-Ebene zuerst nach der angegebenen Datei gesucht. Wird sie nicht gefunden, wird in USER-Ebene 0 gesucht. Ist das Attribut T2 (SYS-Datei) gesetzt, wird die Datei zum Lesen eröffnet. Ein Schreibzugriff ist in diesem Falle nicht vorgesehen.

Ist die Eröffnung der Datei erfolgreich, werden alle Daten aus dem FCB des Inhaltsverzeichnisses in den Benutzer-FCB kopiert. Gleichzeitig werden Datums- und Zeitmarken im Inhaltsverzeichnis eingetragen, falls dies vom Benutzer vorgesehen ist.

War die Eröffnung erfolgreich, kehrt die Funktion mit Register A=00 zum Aufrufer zurück, andernfalls ist Register A=FF.

Im Fehlerfalle meldet das Register H eine genauere Fehlerangabe:

H=00	Datei nicht gefunden
H=01	Lesefehler (Disk I/O)
H=04	Ungültige Laufwerksangabe
H=07	Ungültiges Paßwort
H=09	Joker im Dateinamen oder Index

Funktion 16	schließen einer Datei
Eingabe	C=10H
	DE=FCB-Adresse
Zurück	A=Fehlercode
	H=Fehlernummer

Mit dieser Funktion wird eine zuvor geöffnete, Datei wieder geschlossen.

Das Schließen einer Datei bewirkt das Umgekehrte wie das Eröffnen einer Datei. Der FCB der zu schließenden Datei muß zuvor durch eine erfolgreiche Eröffnung oder Aufbau eines FCB (Funktionen 15 oder 22) aktiviert worden sein. Beim Schließen einer Datei wird der FCB in das Inhaltsverzeichnis zurückgeschrieben, wenn durch Schreiboperationen der geöffnete FCB geändert wurde.

Es erfolgt kein Rückschreiben in das Inhaltsverzeichnis, wenn die eröffnete Datei nur gelesen wurde, diese Funktion braucht in diesem Falle also auch nicht aufgerufen werden.

Eine Besonderheit muß hier noch erwähnt werden. Wird Bit 7 des 5. Buchstabens im Dateinamen (f5') gesetzt, wird die Datei als Zwischendatei (temporary) markiert. Alle Dateien dieser Art können unter CP/M 3 mit Funktion 93 durch einen Funktionsaufruf wieder gelöscht werden.

Die Funktion kehrt mit Register A=00 zurück, wenn die Datei einwandfrei geschlossen werden konnte. Im Fehlerfalle ist Register A=FF und in Register H werden folgende Fehlermeldungen zurückgegeben:

H=01	Lese oder Schreibfehler (Disk I/O)
H=02	Diskette ist Schreibgeschützt
H=04	Ungültiges Laufwerk

Funktion 17	Suche ersten DIR-Eintrag
Eingabe	C=11H
	DE=FCB-Adresse
Zurück	A=DIR-Code
	H=Fehlernummer

Mit dieser Funktion beginnt das Suchen nach einem Eintrag im Inhaltsverzeichnis einer Diskette.

Gesucht wird in dem im FCB angegebenen Laufwerk oder, ist dort 00 spezifiziert, im gerade gültigen Laufwerk. Gesucht wird nach dem ersten Eintrag im Inhaltsverzeichnis, der sich mit dem Dateinamen und Index im FCB deckt.

Im FCB können Fragezeichen als Joker verwendet werden. Auch in diesem Falle wird die erste Datei gesucht, die zum angegebenen FCB paßt.

Anstelle eines FCB's kann <DE> auch auf ein einzelnes Frage-

zeichen zeigen. In diesem Falle wird der erste Eintrag des Inhaltsverzeichnisses eingelesen. Diese Art der Suche ist notwendig, falls nach dem Namen einer Diskette gesucht wird. Dieser ist dadurch kenntlich gemacht, daß anstelle einer Laufwerksbezeichnung (00..0FH) ein 20H eingetragen ist.

Wird der Dateiname nicht gefunden oder tritt ein anderer Fehler auf, kehrt die Funktion mit Register A=FF zurück. Ist Register H=01 trat ein Lesefehler auf, bei Register H=04 wurde ein ungültiges Laufwerk angesprochen.

Konnte die Datei gefunden werden, kehrt die Funktion mit Register A=00 bis A=03 zurück. In diesem Falle ist der Inhalt von Register A ein Zeiger auf den richtigen Eintrag im Inhaltsverzeichnis.

Dies ist so zu verstehen, daß die Funktion jeweils 4 Einträge in die (80H große) momentane DMA einliest. Jeder Eintrag ist 32 Bytes groß. Ist Register A=00, dann ist der erste Eintrag gültig, ist A=01 der Zweite usw.

Ist das Inhaltsverzeichnis (mit INITDIR) für Datums- und Zeitmarken erweitert, wird der jeweils 4. Eintrag in einem Block für die entsprechenden Daten reserviert. Dieser 4. Eintrag beginnt mit 21H. In diesem Falle kehrt jedoch auch Register A nur mit Werten von 0..2 zurück.

Mit diese Funktion wird auch die Funktion 18 initialisiert, mit der jedes weitere Suchen im Inhaltsverzeichnis fortgesetzt wird.

Funktion 18	Suche nächsten DIR-Eintrag
Eingabe	C=012H
Zurück	A=DIR-code
	H=Fehlernummer

Diese Funktion sucht nach Aufruf der Funktion 17 nach weiteren Einträgen im Inhaltsverzeichnis, die mit dem Namen im angegebenen FCB übereinstimmen.

Der Rückgabecode entspricht dem unter Funktion 17 beschriebenen.

Funktion 19	Löschen einer Datei
Eingabe	C=12H
	DE=FCB-Adresse
Zurück	A=Fehlercode
	H=Fehlernummer

Mit dieser Funktion können Dateien aus dem Inhaltsverzeichnis gelöscht werden.

Der Dateiname im FCB kann Fragezeichen als Joker enthalten.

Es wird jede Datei gelöscht, die dem Namen im FCB entspricht. Es wird hierbei genaugenommen nicht der ganze zugehörige FCB im

Inhaltsverzeichnis gelöscht, sondern nur die Laufwerksbezeichnung durch ein E5H ersetzt. Die Datei wird dann nicht mehr im Inhaltsverzeichnis dargestellt und der zugehörige FCB vom nächsten Eintrag überschrieben.

Konnte(n) die angegebene(n) Datei(en) gelöscht werden, kehrt die Funktion mit Register A=00 zurück, andernfalls ist Register A=FF.

Im Fehlerfall hat der Inhalt des H-Registers bei der Rückkehr folgende Bedeutung:

H=01	Schreib-/Lesefehler (Disk-I/O)
H=02	Diskette ist schreibgeschützt
H=03	Datei ist schreibgeschützt (Read-Only)
H=04	Ungültige Laufwerksangabe
H=07	Ungültiges Paßwort

Funktion 20	sequentiell Lesen
Eingabe	C=14H
	DE=FCB-Adresse
Zurück	A=Fehlercode
	H=Fehlernummer

Diese Funktion liest die nächsten 1..128 logischen 128 Byte Sektoren einer Datei an die gültige DMA-Adresse.

Mit Funktion 44 kann definiert werden, wieviele logische Sektoren bei jedem Aufruf eingelesen werden sollen. Bei jedem Warmstart wird dieser Wert vom CCP auf '1' gesetzt und ist somit CP/M 2 kompatibel.

Vor dem Lesen muß die Datei mit Funktion 15 erfolgreich eröffnet worden sein.

Soll eine Datei von Beginn an gelesen werden, so muß das CR-Feld im FCB auf 0 gesetzt werden. Das CR-Feld wird vom BDOS im FCB mitgehalten bis die Daten des FCB's im Inhaltsverzeichnis den Daten im Benutzer-FCB entsprechen. (d.h. die Datei vollständig eingelesen wurde).

Ist eine Datei größer als 16K wird die entsprechende EXtend-Datei automatisch eröffnet und aus dem 'neuen' FCB weiter gelesen.

Kehrt die Funktion mit Register A=00 zurück, konnte ein Sektor fehlerfrei gelesen werden. Andernfalls werden in Register A folgende Fehler gemeldet:

A=01	Lesen von ungeschriebenen Daten (...Dateiende)
A=09	falscher FCB oder bereits geschlossener FCB
A=10	Diskette wurde während des Lesens gewechselt
A=FF	hardware-Fehler

Bei allen Fehlern (außer den Hardware-Fehlern) enthält das Register H die Anzahl der bereits erfolgreich gelesenen logischen Sektoren.

Wurde ein Hardware-Fehler gemeldet, enthält das Register H nähere Angaben:

H=01	Lesefehler (Disk-I/O)
H=04	Ungültiges Laufwerk

Funktion 21	sequentiell Schreiben
Eingabe	C=15H
	DE=FCB-Adresse
Zurück	A=Fehlercode
	H=Fehlernummer

Diese Funktion schreibt die nächsten 1..128 logische 128 Byte Sektoren einer Datei beginnend bei der momentanen DMA-Adresse auf Diskette.

Mit Funktion 44 kann gesetzt werden, wieviele logische Sektoren mit einem Aufruf geschrieben werden sollen. Bei jedem Warmstart wird dieser Wert vom CCP auf '1' zurückgesetzt, damit die Funktion CP/M 2 kompatibel ist.

Vor dem Schreiben muß der FCB mit Funktion 15 oder 22 erfolgreich eröffnet worden sein. Nach dem Eröffnen muß das CR-Feld auf 00 zurückgesetzt werden, wenn von Dateibeginn an geschrieben werden soll.

Die Funktion eröffnet von sich bei Dateien von über 16k Größe weiter FCB's.

Die Funktion kehrt mit Register A=00 zurück, wenn erfolgreich geschrieben werden konnte. Im Fehlerfall enthält Register A folgenden Fehlercode:

A=01	Inhaltsverzeichnis voll
A=02	Diskette voll
A=09	Falscher FCB oder bereits geschlossener FCB
A=10	Diskettenwechsel während des Schreibens
A=FF	Hardware-Fehler

Bei allen Fehlermeldungen außer bei hardware-Fehler, enthält Register H die Anzahl der bereits geschriebenen logischen Sektoren.

Bei Hardware-Fehler enthält das Register H folgenden Fehlercode:

H=01	Schreibfehler (Disk I/O)
H=02	Diskette ist Schreibgeschützt
H=03	Datei ist Schreibgeschützt (Read-Only) oder Datei in USER-Ebene 0 aus einer anderen USER-Ebene eröffnet oder Datei Paßwortgeschützt beim Schreiben
H=04	Ungültiges Laufwerk

Funktion 22	Eröffnen eines DIR-Eintrages
Eingabe	C=16H
	DE=FCB-Adresse
Zurück	A=Fehlercode
	H=Fehlernummer

Diese Funktion wird aufgerufen, wenn eine neue Datei erstellt werden muß, die noch nicht im Inhaltsverzeichnis aufgeführt ist.

Die Auswirkung dieser Funktion entspricht im Übrigen allem, was zu Funktion 15 gesagt wurde, incl. den möglichen Fehlermeldungen.

Funktion 23	Umbenennen einer Datei
Eingabe	C=17H
	DE=FCB-Adresse
Zurück	A=Fehlercode
	H=Fehlernummer

Mit dieser Funktion können Dateien umbenannt werden.

Der FCB dieser Funktion muß in den ersten 16 Bytes den alten Dateinamen enthalten (wie in allen FCB's gewohnt). In den zweiten 16 Bytes dieses FCB's muß dann, im gleichen Format, der neue Namen der Datei stehen. In diesem Falle sind keine Fragezeichen als Joker zugelassen.

Ist die umzubenennende Datei durch ein Paßwort geschützt, muß das richtige Paßwort in die ersten 8 Bytes der gültigen DMA geschrieben werden.

Das DR-Feld im FCB wird zur Auswahl des betroffenen Laufwerkes herangezogen.

Wurde die Umbenennung erfolgreich durchgeführt, kehrt die Funktion mit Register A=00 zurück.

Im Fehlerfalle ist Register A=FF und Register H enthält folgenden Fehlercode:

H=01	Schreib/Lesefehler (Disk I/O)
H=02	Diskette schreibgeschützt
H=03	Datei schreibgeschützt
H=04	Ungültiges Laufwerk
H=07	Ungültiges Paßwort
H=08	Dateiname bereits vorhanden
H=09	Joker (?) im Dateinamen

Funktion 24	Lese Login Vektor
Eingabe	C=18H
Zurück	HL=Login Vektor

Diese Funktion bringt im Doppelregister <HL> einen Wert zurück, der die eingeloggten (nach dem Booten oder CNTRL-C zumindest einmal benutzten) Laufwerke repräsentiert.

Bit 0 des übergebenen 16 Bitwortes entspricht dabei Laufwerk A und Bit 15 Laufwerk P.

Ein Laufwerk ist aktiviert, wenn das entsprechende Bit gleich 1 ist, deaktiviert (nicht eingeloggt), wenn das entsprechende Bit 0 ist.

Zur Kompatibilität mit früheren Versionen von CP/M enthält Register A den gleichen Wert wie Register L.

Funktion 25	monentanes Laufwerk
Eingabe	C=19H
Zurück	A=momentanes Laufwerk

Diese Funktion stellt fest, welches Laufwerk zuletzt aktiviert war.

Der entsprechende Code wird in Register A zurückgegeben mit 00=A, 01=B, 02=C usw.

Funktion 26	Setzten der DMA-Adresse
Eingabe	C=1AH
	DE=DMA-Adresse

Diese Funktion setzt die Adresse der DMA.

DMA ist die CP/M-typische Abkürzung für Direct Memory Address im Sinne von Direkter Speicher Adresse. Die DMA ist üblicherweise ein Puffer, aus welchem gelesen oder in welchen geschrieben werden kann.

Die Größe des Puffers kann mit Funktion 44 gesetzt werden. Bei jedem Rücksprung in den CCP wird von diesem die Puffergröße automatisch auf 128 Bytes gesetzt, um mit CP/M 2 kompatibel zu bleiben.(Ein logischer 128 Byte Sektor). Solange der CCP aktiv ist oder SID und Co., ist die DMA fest auf Adresse 80H eingestellt.

Die DMA-Adresse wird nicht NUR zum Lesen oder Schreiben von Dateien benötigt, vielmehr erwarten oder übergeben einige Funktionen im DMA-Puffer ihre Daten.

Wurde eine DMA-Adresse neu gesetzt, so behält sie ihren Wert bis eine neuer Funktionsaufruf ihn ändert.

Mit Funktion 49 ist es möglich die gerade gültige DMA-Adresse festzustellen, falls dies in einem Programm wichtig sein sollte.

Anmerkung:

Die Bezeichnung DMA sollte auf keinen Fall mit der gleichlautenden Bezeichnung für entsprechende Hardware-Bauteile verwechselt werden.

Funktion 27	hole Belegungsvektor (Allocation)
Eingabe	C=1BH
Zurück	HL=Vektor-Adresse

Diese Funktion übergibt einen Zeiger auf die Belegungsvektoren.

Diese Funktion ist unter CP/M 3 nur noch bedingt verwendbar, da im gebankten System der Vektor in Bank 0 liegen kann und sich so dem Zugriff aus der TPA heraus verschließt.

Aus dem Belegungsvektor kann errechnet werden, inwieweit eine Diskette bereits belegt ist bzw. wieviel freier Speicherplatz auf ihr noch zur Verfügung steht.

Wurde (mit GENCPM) der Vektor im gemeinsamen Speicherbereich belassen kann die Funktion weiter benutzt werden. In jedem Falle wird die Adresse des Vektors in Register HL zurückgegeben.

Im Fehlerfalle übergibt das Doppelregister <HL> den Wert FFFF.

Soll die Restkapazität unter CP/M 3 festgestellt werden, kann hierzu Funktion 46 verwendet werden.

Anmerkung:

Bei Anwendung dieser Funktion in CP/M 2 Programmen unter CP/M 3 können hier Fehlangaben bzgl. des verbleibenden Speicherplatzes auftreten. Eine einfache Lösung des Problemes ist nicht so ohne weiters möglich. Denkbar wäre es, mit Funktion 46 die Restkapazität festzustellen und über einen RSX einen 'eigenen' Vektor aufzubauen, dessen Adresse an den Funktionsaufrufer zurückgegeben wird.

Funktion 28	Schreibschutz einschalten
Eingabe	C=1CH

Mit dieser Funktion wird das zuletzt aktive Laufwerk softwaremäßig mit einem Schreibschutz versehen.

Jeder Schreibzugriff auf dieses Laufwerk wird verwehrt, bis entweder Funktion 13 oder 37 aufgerufen wurde.

Funktion 29	Schreibschutz feststellen
Eingabe	C=1DH
Zurück	HL=Schreibschutzvektor

Mit dieser Funktion kann festgestellt werden, welche Laufwerke mit Funktion 28 schreibgeschützt wurden.

Die Information wird in einem 16 Bit-Wort in Register HL zurückgegeben. Bit 0 entspricht Laufwerk A, Bit 1 Laufwerk B usw. Ist das entsprechende Bit=1, dann ist das Laufwerk schreibgeschützt, ist es 0, dann kann das Laufwerk beschrieben werden.

Diese Funktion kann NICHT feststellen, ob eine Diskette hardwaremäßig schreibgeschützt ist.

Funktion 30	Setzen von Dateiattributen
Eingabe	C=1EH
	DE=FCB-Adresse
Zurück	A=Fehlercode
	H=Fehlernummer

Mit Aufruf dieser Funktion kann ein Programm die Datei-Attribute setzten oder ändern.

Dies ist nur mit dieser Funktion möglich. Es ist nicht durch einfaches Ändern des FCB's beim Schreiben oder Lesen möglich.

Die Attribute (Bit 7 gesetzt=EIN zurückgesetzt=AUS) F1 bis F4 und T1 bis T3 müssen im FCB entsprechend gesetzt sein, wenn diese Funktion aufgerufen wird. Joker in Dateinamen sind nicht zugelassen.

Falls die aufzurufende Datei mit einem Paßwort versehen ist, muß dieses in die ersten 8 Speicherstellen der gültigen DMA geschrieben werden oder der 'default'-Modus muß eingeschaltet sein.

Die Funktion sucht im Inhaltsverzeichnis der vorgegebenen USER-Ebene und dem im DR-Feld angegebenene Laufwerk nach einer Datei mit dem entsprechenden Namen. Wird die Datei gefunden, werden die Attribute angepaßt und die Funktion kehrt mit Register A=00 zurück.

Im Fehlerfalle ist Register A=FF. Eine nähere Fehlerbezeichnung wird in Register H mit folgender Bedeutung übergeben:

H=00	Datei nicht gefunden
H=01	Schreib/Lesefehler (Disk I/O)
H=02	Diskette ist schreibgeschützt
H=04	ungültiges Laufwerk
H=07	ungültiges Paßwort
H=09	Joker (?) im Dateiname

Funktion 31	hole DPB-Adresse
Eingabe	C=1FH
Zurück	HL=DPB-Adresse

Mit dieser Funktion wird die Adresse des DPB (Disketten Parameter Block) des zuletzt aktiven Laufwerkes übergeben.

Aus dem DPB können die wichtigsten Disketten-Parameter übernommen werden. Näheres hierzu in Teil IV.

Im Fehlerfalle (siehe Funktion 45) übergibt das Doppelregister <HL> den Wert FFFF.

Der DPB liegt unter CP/M 3 im gemeinsamen Speicherbereich, auf welchen auch aus der TPA heraus zugegriffen werden kann.

Funktion 32	setzen/lese USER-Nummer
Eingabe	C=20H
	E=FFH (lesen) oder USER-Nummer
Zurück	A=USER-Nummer oder Ungültig

Diese Funktion ermöglicht es, aus einem Programm heraus die USER-Ebene umzuschalten.

Die USER-Nummer wird in HEX(adezimal, 00..0FH) in Register <E> übergeben.

Wird Register <E> beim Aufruf auf FFH gesetzt, wird die momentane USER-Nummer gelesen und in Register A zurückgeben.

Wenn ein Aufruf mit Register E=USER-Nummer erfolgt, ist bei der Rückkehr der Funktion der Inhalt des Registers A ungültig.

Funktion 33	Wahlfreier Lesezugriff
Eingabe	C=21H
	DE=FCB-Adresse
Zurück	A=Fehlercode
	H=Fehlernummer

Mit dieser Funktion können einzelne Sektoren einer Datei wahlfrei gelesen werden.

Der wahlfreie Lesezugriff ist ähnlich dem sequentiellen Lesen. Der Lesezugriff wird jedoch über einen 24-Bit Wert in r0,r1 und r2 des FCB's gesteuert, der den Offset (in logischen Sektoren) vom Beginn der ersten Datei an angibt.

Die genannten Zeiger werden an einen normalen FCB als Byte 33,34 und 35 angehängt. Die Werte werden mit r0 als unterstes Byte, r1 als mittleres Byte und r3 als höchstes Byte abgelegt.

Es können max. 262143 Sektoren in einem DIR-Eintrag abgelegt werden.

Um eine Datei mit dieser Funktion zu lesen, muß zuerst der zugehörige FCB mit einem EX von 0 eröffnet werden. Hierdurch wird der FCB vom BDOS korrekt initialisiert. Die Funktion liest danach für jeden Aufruf den im RC-Feld angegebenen Sektor in den vorgegebenen DMA-Puffer.

Im Gegensatz zum sequentiellen Lesen wird der EX(tend) jedoch nicht inkrementiert, dies muß vom Programm übernommen werden.

Wurde mit Funktion 44 ein Mehrfachzugriff zugelassen, so wird die dort vorgegebene Anzahl von logischen Sektoren gelesen.

Konnte fehlerfrei gelesen werden, wird Register A=00 zurückgegeben. Andernfalls meldet Register A folgende Fehler:

A=01	Dateiende
A=03	der momentane Ext. kann nicht geschlossen werden
A=04	der gewünschte Extend wurde nicht beschrieben
A=06	Sektornummer zu groß (über 262143)
A=10	Diskette wurde gewechselt
A=FF	physikalscher Fehler

Wird ein physikalischer Fehler gemeldet, sind nähere Angaben aus Register H zu entnehmen:

H=01	Lesefehler (Disk I/O)
H=04	ungültiges Laufwerk

Handelt es sich nicht um einen physikalischen Fehler, wird in Register H die Anzahl der mit Funktion 44 vorgegebenen Sektoren angegeben, die mit dem momentanen Funktionsaufruf bereits gelesen wurden. Ist Funktion 44 mit 1 gesetzt, bleibt H immer auf 00H.

Funktion 34	Wahlfreier Schreibzugriff
Eingabe	C=22H
	DE=FCB-Adresse
Zurück	A=Fehlercode
	H=Fehlernummer

Mit dieser Funktion können einzelne Sektoren einer Datei wahlfrei beschrieben werden.

Das wahlfreie Schreiben ist das Gegenstück zum wahlfreien Lesen. Hier werden die Daten auf die Diskette geschrieben.

Das BDOS übernimmt automatisch die Belegungszuweisung im gültigen FCB, der zuvor mit Funktion 15 oder 22 eröffnet werden muß. Hierbei muß ebenfalls der EXtend auf 0 gesetzt werden, um eine korrekte Initialisierung zu Gewährleisten.

Ist der FCB noch leer (wurden noch keinerlei Daten geschrieben), muß Funktion 22 zur Eröffnung aufgerufen werden.

Sind Datums- und Zeitmarken aktiviert, werden diese automatisch eingetragen.

Mit Funktion 44 kann das Schreiben von mehreren logischen Sektoren bei einem Funktionsaufruf erzwungen werden.

Konnte fehlerfrei geschrieben werden, kehrt die Funktion mit Register A=00 zurück. Andernfalls meldet der Inhalt von Register A folgende Fehler:

A=02	Es steht kein Datenblock mehr zur Verfügung
A=03	Der momentane EXtend kann nicht geschlossen werden
A=05	Inhaltsverzeichnis voll
A=06	Sektornummer zu groß
A=10	Diskettenwechsel während des Schreibens
A=FF	physikalischer Fehler

Tritt ein physikalischer Fehler auf, werden nähere Angaben in Register H mit folgender Bedeutung übergeben:

H=01	Schreibfehler (Disk-I/O)
H=02	Diskette ist schreibgeschützt
H=03	Datei ist schreibgeschützt oder USER-Ebene falsch
H=04	ungültiges Laufwerk

Funktion 35	Berechne Dateigröße
Eingabe	C=23H
	DE=FCB-Adresse
Zurück	A=Fehlercode
	H=Fehlernummer
	r0..r2-Feld gesetzt

Dies Funktion berechnet die Dateigröße der im FCB angegebenen Datei.

Genaugenommen wird als Ergebnis die Adresse des Sektors angegeben, der dem letzten Sektor der angegebenen Datei folgt.

Zur Anwendung dieser Funktion müssen im FCB die Bytes r0..r2 (Bytes 33 bis 35) reserviert sein. Diese Bytes werden nach Aufruf der Funktion auf die letzte Sektornummer (+1), die von der Datei belegt wurde, gesetzt.

Ist Byte r2=04H, dann ist die maximal mögliche Anzahl wahlfreier Sektoren erreicht. (262144).

Ein Programm kann an eine vorhandene Datei Daten anhängen, wenn vor dem Schreiben der neue Sektorzähler mit dieser Funktion gesetzt wurde.

Es ist nicht notwendig, daß beim Aufruf dieser Funktion die entsprechende Datei bereits eröffnet wurde. Sie muß auf alle Fälle jedoch vor einem Aufruf geschlossen worden sein, da sonst falsche Daten übergeben werden können.

Wurde die Funktion fehlerfrei beendet, wird Register A=00 zurück gegeben und die Bytes r0..r2 im FCB enthalten die gewünschten Daten. Andernfalls enthält Register A=FF.

Nähere Fehlermeldungen bei Hardware-Fehler werden in Register H mit folgender Bedeutung übergeben:

H=01	Lesefehler (Disk-I/O)
H=04	ungültiges Laufwerk

Funktion 36	Setze wahlfreien Sektor
Eingabe	C=24H
	DE=FCB-Adresse
Zurück	r0..r2 gesetzt

Mit dieser Funktion wird der Sektor festgelegt, auf den mit der nächsten Funktion wahlfrei (direkt) zugegriffen werden soll.

Diese Funktion kann benutzt werden, wenn eine Datei bis zu einem gewissen Punkt (mit Funktion 20 bzw. 21) gelesen oder beschrieben wurde und auf weitere Daten nun wahlfrei zugegriffen werden soll.

Die Funktion setzt die Bytes r0..r2 entsprechend der belegten Sektornummer.

Der FCB muß vor Aufruf der Funktion erfolgreich eröffnet worden sein.

Fehler sind mit dieser Funktion nicht zu erwarten.

Funktion 37	Laufwerk zurücksetzen
Eingabe	C=25H
	DE=Laufwerksvektor
Zurück	A=00

Diese Funktion setzt alle angegebenen Laufwerke in den RESET-Zustand zurück.

Dies bedeutet, daß beim nächsten Aufruf eines zurückgesetzten Laufwerkes vom BDOS zuerst ein erneuter Login veranlaßt wird (Ein erneutes Einstellen aller Diskettenparameter).

Der Laufwerksvektor ist ein 16-Bit Wort mit Bit 0=Laufwerk A und Bit 15=Laufwerk P. Ein Laufwerk wird zurückgesetzt, wenn das entsprechende Bit im Vektor auf 1 gesetzt wird. Ist das entsprechende Bit=0, bleibt das Laufwerk in seinem bisherigen Zustand.

Funktion 38	auf Laufwerk zugreifen
Eingabe	C=26H
Zurück	A=00H unter CP/M 3

Dies ist eine von CP/M 3 nicht unterstützte MP/M-Funktion.

Funktion 39	Laufwerk freigeben
Eingabe	C=27H
Zurück	A=00H unter CP/M 3

Dies ist eine von CP/M 3 nicht unterstützte MP/M-Funktion

Funktion 40	wahlfreies Schreiben mit Auffüllen von 00H
Eingabe	C=28H
	DE=FCB-Adresse
Zurück	A=Fehlercode
	H=Fehlernummer

Diese Funktion entspricht in allem der Funktion 34 (wahlfreies Schreiben), mit der Ausnahme, daß alle Sektoren vor dem Beschreiben mit 00H (überschrieben) gelöscht werden.

Kapitel 3 Die CP/M 3 typischen Funktionen

Werden Programme geschrieben, die auch unter CP/M 2 funktionieren sollen, können die nachfolgenden Funktionen (leider) nicht aufgerufen werden.

Um unliebsame Effekte beim CP/M 2 Benutzer zu vermeiden, falls er doch Programme dieser Art in die Hand bekommt, sollten alle Programme die Funktionsaufrufe aus diesem Kapitel benutzen, immer zuerst über Funktion 12 die Versionsnummer abfragen und bei Inkompatibilität das Programm mit einer Fehlermeldung abbrechen.

Es sei bei dieser Gelegenheit auch auf die Nichtverwendbarkeit von Funktion 7 und 8 unter CP/M 2 hingewiesen; auch Funktion 30 kann zu unliebsamen Störungen führen.

Die nachfolgend aufgelisteten Funktionen haben nicht mehr unbedingt aufeinanderfolgende Funktionsnummern. Es steht zu vermuten, daß sich Digital Research die fehlenden Funktionsnummern für Erweiterungen in 16-Bit CP/M Systemen reserviert hat.

Werden Funktionen, die nicht angegeben sind, aufgerufen, kehren sie ohne 'murren' zurück.

Funktion 41	Testen und Schreiben eines Sektors
Eingabe	C=29H
	DE=FCB-Adresse
Zurück	A=Fehlercode
	H=Fehlernummer

Dies ist eine unter CP/M 3 nicht unterstützte MP/M-Funktion.

Wird sie unter CP/M aufgerufen kehrt die Funktion mit Register A=FF und Register H=00 zurück.

Funktion 42	Datei sichern
Eingabe	C=2AH
	DE=FCB-Adresse
Zurück	A=00H

Dies ist eine unter CP/M 3 nicht unterstützte MP/M-Funktion.

Funktion 43	Datei entsichern
Eingabe	C=2BH
	DE=FCB-Adresse
Zurück	A=00H

Dies ist eine unter CP/M 3 nicht unterstützte MP/M-Funktion.

Die nachfolgenden Funktionsbeschreibungen beginnen alle am Anfang einer neuen Seite um eine bessere Übersicht zu erhalten.

Funktion 44	Vielfach-Sektor Zähler setzen
Eingabe	C=2CH
	E=Sektor-Zähler
Zurück	A=Fehlercode

Mit dieser Funktion wird ein Zähler gesetzt, der die Anzahl der mit einem Funktionsaufruf zu bearbeitenden logischen Sektoren angibt.

Unter CP/M werden normalerweise nur logische Sektoren mit jeweils 128 Bytes mit einem Schreib- oder Lesevorgang bearbeitet. Auch der Puffer in der Standard-DMA auf Adresse 80H ist nur so groß, daß ein logischer Sektor gerade 'hinein paßt'.

Unter CP/M 3 ist es nun möglich, beim Schreiben und Lesen von Dateien die behandelte Sektorgröße auf ein Vielfaches der logischen Sektorgröße festzulegen. Es sind Faktoren von 1...128 möglich.

Die neu gesetzte Sektorgröße bleibt solange erhalten, bis sie von einem erneuten Aufruf der Funktion geändert wird.

Es ist jedoch immer zu beachten, daß der CCP bei jedem Warmstart die Sektor-Größe wieder auf 1 zurücksetzt, um evtl. Schwierigkeiten mit dem Standard-Puffer sicher zu umgehen, falls CP/M 2-Programme aufgerufen werden.

Diese Funktion kehrt nach Aufruf mit Register A=00 zurück, wenn der neue Sektorzähler im Bereich von 1..128 gesetzt wurde. Werden andere Bereiche angegeben, wird der Befehl ignoriert und das Register A kehrt mit A=FF zurück.

Funktion 45	Setze BDOS Fehlermodus
Eingabe	C=2DH
	E=Fehlermodus

Diese Funktion definiert die Art der Fehlermeldung des BDOS.

Es sind drei Modi möglich:

E=FFH

Fehler werden nur in den für einen Funktionsaufruf vorgesehenen Registern zurückgemeldet

E=FEH

Fehler werden in den für den Funktionsaufruf vorgesehenen Registern zurückgemeldet und darüberhinaus noch auf der Konsole als Klartext wie im BIOS vorgesehen.

E=

jede beliebige andere Eingabe veranlaßt die Art der Fehlermeldung wie sie bei der Systemgenerierung mit GENCPM festgelegt wurde. Zusätzlich wird der Abbruch des Programmes veranlasst.

Funktion 46	lese restlichen Speicherplatz
Eingabe	C=2EH
	E=Laufwerksnummer
Zurück	A=Fehlercode
	H=Fehlernummer
	DMA=Daten

Mit dieser Funktion kann die restliche (unbelegte) Speichergröße einer Diskette abgefragt werden.

Die Angabe erfolgt als Anzahl der verbleibenden logischen (128-Byte) Sektoren.

Die Laufwerksangabe wird in HEX-Werten verlangt, wobei A=00, B=01 usw. entspricht.

Ist bei der Rückkehr der Funktion das Register A=00, konnte die Funktion einwandfrei ausgeführt werden. In diesem Falle stehen die gesuchten Werte in den ersten 3 Bytes der momentanen DMA-Adresse, wobei DMA+0 der untere Wert, DMA+1 der mittlere und DMA+2 der obere Wert ist.

Im Fehlerfalle kehrt das Register A=FF zurück, nähere Fehlerangaben befinden sich dann in Register H mit folgender Bedeutung:

H=01	Lesefehler (Disk I/O)
H=04	ungültiges Laufwerk

Achtung:

Wenn das Laufwerk schreibgeschützt ist, kann der von Funktion 46 zurückgegebene Wert fehlerhaft sein!

Funktion 47	Programm übergeben
Eingabe	C=2FH
	E=Flag

Diese Funktion ermöglicht es, aus einem Programm heraus nach dessen Abschluß, ohne Eingriff des Benutzers, ein weiteres Programm anzuhängen.

Hierzu ist es notwendig, in die derzeitige DMA den nächsten Programmaufruf als Befehl, wie vom CCP gewohnt, einzuschreiben. Diese Befehlszeile muß mit 00H abgeschlossen sein.

Ist bei der Übergabe Register E=FFH, werden alle Daten bezüglich USER-Ebene und anzusprechendem Laufwerke vom aufrufenden Programm übernommen. Andernfalls werden die CCP-Vorgabewerte zum Aufruf verwendet.

Mit Funktion 108 kann ein 2-Byte Wert zusätzlich an das folgende Programm übergeben werden.

Funktion 48	Puffer freigeben
Eingabe	C=30H
	E=Flag
Zurück	A=Fehlercode
	H=Fehlernummer

Diese Funktion veranlaßt, daß alle Daten, die noch in den internen Puffer vorhanden sind, auf Diskette geschrieben werden, oder Puffer, die z.B. zum Datenvergleich angelegt wurden, freigegeben werden.

Vergleichspuffer (wie von PIP benötigt) werden freigegeben, wenn Register E=FFH ist. Andernfalls werden nur die internen (oder BIOS-abhängigen) Sektorpuffer abgearbeitet.

Ist die Funktion erfolgreich, kehrt sie mit Register A=00 zurück. Andernfalls ist Register A=FF. In diesem Falle enthält Register H einen Fehlercode mit folgender Bedeutung:

H=01	Schreibfehler (Disk-I/O)
H=02	Diskette ist schreibgeschützt
H=04	ungültiges Laufwerk

Funktion 49	lesen (schreiben) aus (in) SCB
Eingabe	C=31H
	DE=SCBPB-Adresse
Zurück	A=Byte Wert
	HL=Wort-Wert

Diese Funktion ermöglicht den Datenaustausch mit dem SCB (System-Kontroll-Block).

Dieser Kontroll-Block ist ein 80H-Byte großes Datenfeld, in dem Flags, Daten und Zeiger abgelegt werden, die vom BDOS, BIOS und SCB gemeinsam benötigt werden.

Zur Benutzung dieser Funktion ist es notwendig dem BDOS die Adresse eines Datenfeldes zu übergeben (SCBPB=SCB-Parameter-Block), das wie folgt aufgebaut ist:

```
 1
 2
 3 0000' 00        SCBPB:  ds   1      ; hier wird der Offset eingetragen (gibt Entfernung des
 4                                      ; gesuchten Wertes vom Start des SCB's
 5                                      ; an.
 6
 7 0001' 00                ds   1      ; dieses Flag hat folgende Bedeutung:
 8                                      ; FF=setzen eines Bytes im SCB
 9                                      ; FE=setzten eines Wortes im SCB
10                                      ; FD..01 nicht zugelassen
11                                      ; 00=lesen eines Wertes aus dem SCB
12 0002' 0000             ds   2      ; hier wird der Wert des zu schreib-
13                                      ; enden Bytes oder Wortes übergeben.
14                                      ; wobei für Bytes nur der erste Wert
15                                      ; gültig ist.
```

Das Lesen von Werten aus dem SCB ist immer gefahrlos. Werden jedoch Werte neu gesetzt, liegt es allein beim Benutzer der Funktion, die Datensicherheit zu überprüfen.

Falsche Daten können sehr einfach zu sehr unliebsamen Folgen führen.

Wird als Offset ein Wert über 63H eingegeben, so bleibt die Funktion (ohne Störung) unbeantwortet; jedoch werden die Register A und HL mit den Werten 0 zurückgegeben.

Der komplette Inhalt des SCB's ist nur teilweise offengelegt worden. Die bekannten Felder sind nachstehender Tabelle zu entnehmen:

Achtung:

Diese Funktion ist CP/M 3 typisch und kann nicht unter MP/M verwendet werden.

Offset	Beschreibung
00..04H	Reserviert vom Betriebssystem
05H	BDOS-Versionsnummer
06..09H	Benutzer-Flags (frei verwendbar)
0A..0FH	Reserviert vom Betriebssystem
10..11H	Fehlercode
12..19H	Reserviert vom Betriebssystem
1AH	Zeichen pro Zeile für Konsolenausgabe
1BH	momentane Spaltenposition für Konsolenausgabe
1CH	Zeilen pro Seite für Konsolenausgabe
1D..21H	Reserviert vom Betriebssystem
22..23H	CONIN Belegungsvektor
24..25H	CONOUT Belegungsvektor
26..27H	AUXIN Belegungsvektor
28..29H	AUXOUT Belegungsvektor
2A..2BH	LSTOUT Belegungsvektor
2CH	Seitenmodus (Flag) (page - nopage)
2DH	Reserviert vom Betriebssystem
2EH	CNTRL-H Aktiv-Flag
2FH	DEL Aktiv-Flag
30..32H	Reserviert vom Betriebssystem
33..34H	Konsolen Modus
35..36H	Reserviert vom Betriebssystem
37H	Ausgabe Delimiter (Funktion 9 Standard ist '$')
38H	Druckerflag (Cntrl-p)
39..3BH	Reserviert vom Betriebssystem
3C..3DH	momentane DMA-Adresse
3EH	Momentanes Laufwerk
3F..43H	Reserviert vom Betriebssystem
44H	Momentane USER-Nummer (Ebene)
45..49H	Reserviert vom Betriebssystem
4AH	BDOS Vielfach-Sektor-Zaehler (Funktion 44)
4BH	BDOS Fehler Modus (Funktion 45)
4C..4FH	Laufwerkssuchkette (Laufwerk 1..2..3)
50H	Laufwerk für zeitweise Datenablage
51H	Laufwerk in dem zuletzt ein Fehler auftrat
52..56H	Reserviert vom Betriebssystem
57H	BDOS-Flag
58..5CH	Datum- und Zeit
5D..5EH	Startadresse des gemeinsamen Speicherbereiches
5F..FFH	Reserviert vom Betriebssystem

Tabelle 14 Eintragungen im System Controll Block

Funktion 50	Direkter BIOS-Aufruf
Eingabe	C=32H
	DE=BIOSPB-Adresse
Zurück	Register entsprechend den BIOS Rückgaben gesetzt

Mit dieser Funktion können BIOS-Unterprogramme aus der BIOS-Sprungleiste direkt aufgerufen werden.

In einem ungebankten (CP/M 2) System konnten aus der TPA heraus alle Funktionen, die aus der (festgelegten) Einsprungleiste des BIOS erreichbar waren, angesprungen werden. Unter CP/M 3 in einem gebankten System ist dies nur noch bedingt möglich.

Nach wie vor erreichbar sind alle Unterprogramme die mit der EIN- oder AUS-Gabe von Zeichen auf die Konsole, den AUX-Kanal und dem Drucker zu tun haben. (Fast) alle anderen Einsprünge zeigen auf Unterprogramme, die im gebankten Bereich sind.

Nun kann zwar ein TPA-Programm problemlos die entsprechende Einsprungadresse in der Sprungleiste des BIOS erreichen, die Adresse des Sprunges zeigt dann jedoch auf einen zufälligen Speicherinhalt im TPA-Bereich. Das kann zu (zum mindesten) merkwürdigen Ergebnissen führen.

Viele Hilfsprogramme zum Reparieren von Disketten u.Ä. unter CP/M 2 benutzten die Unterprogrammes des BIOS zum direkten Diskettenzugriff. Unter CP/M 3 ist dies nun nur noch über Funktion 50 möglich.

Zur Datenübergabe ist es notwendig, das Register DE mit einem Zeiger auf ein spezielles Datenfeld, dem BIOSPB (BIOS-Parameter-Block) zu laden. Dieses Datenfeld ist vor dem Funktionsaufruf mit folgenden Daten zu laden:

```
 1
 2 0000' 00        BIOSPB:  ds   1       ; Nummer des BIOS-Einsprunges
 3                                       ; BOOT=1, WBOOT=2 usw
 4
 5                          ;
 6                          ; nachfolgend können die vom BIOS benötigten Register-
 7                          ; inhalte übergeben werden. Im BDOS werden alle
 8                          ; benannten Register vor dem BIOS-Einsprung entsprechend
 9                          ; geladen. Die Reihenfolge ist unveränderbar
10                          ;
11
12 0001' 00                ds   1       ; Inhalt des A-Registers ohne Flags
13 0002' 0000              ds   2       ; Inhalt des Registers BC
14 0004' 0000              ds   2       ; Inhalt des Registers DE
15 0006' 0000              ds   2       ; Inhalt des Registers HL
```

Nach Aufruf der Funktion werden dem Benutzer die Register mit den Originalinhalten wie vom BIOS gesetzt zurückgegeben.

Es ist zu beachten, daß der Einsprung 27 (Bankauswahl) über Funktion 50 nicht angesprochen werden kann.

Nähere Angaben über die BIOS Sprungleiste sind in Teil IV (Hardware-Anpassung) zu finden. Die Sprungleiste selbst ist im Unterprogramm BIOSKERNL abgedruckt.

Der Vollständigkeit halber werden alle BIOS-Aufrufe mit den Übergabekonventionen in der nachfolgenden Tabelle noch einmal zusammen gefaßt:

Nr	Funktion	Eingabe	Zurück
0	BOOT	keine	Startet CP/M (Kaltstart
1	WBOOT	keine	Lädt CCP (Warmstart)
2	CONST	keine	A=FF wenn bereit
			A=00 wenn nicht bereit
3	CONIN	keine	A=Zeichen
4	CONOUT	C=Zeichen	nichts
5	LIST	C=Zeichen	nichts
6	AUXOUT	C=Zeichen	nichts
7	AUXIN	keine	A=Zeichen
8	HOME	keine	nichts
9	SELDSK	C=Laufwerk	HL=DPH-Adresse
		E=Initflag	HL=0 wenn ungültig
10	SETTRK	BC=Tracknummer	nichts
11	SETSEC	BC=Sektornummer	nichts
12	SETDMA	BC=DMA-Adresse	nichts
13	READ	keine	A=00 Fehlerfrei
			A=01 hardware A=FF media
14	WRITE	C=debl. code	A=00 Fehlerfrei
			A=01 hardware A=FF media
15	LISTST	keine	A=00 wenn bereit
			A=FF wenn nicht bereit
16	SECTRN	BC=log SEK-Nr	HL=phys.Sektornummer
		DE=Adresse der	Translatetabelle
17	CONOST	keine	A=FF wenn bereit
			A=00 wenn nicht bereit
18	AUXIST	keine	A=FF wenn bereit
			A=00 wenn nicht bereit
19	AUXOST	keine	A=FF wenn bereit
			A=00 wenn nicht bereit
20	DEVTBL	keine	HL=DEVTBL-Adresse
21	DEVINI	C=DEV-Nummer	nichts
22	DRVTBL	keine	HL=DRVTBL-Adresse
			oder HL=FFFF
23	MULTIO	C=Sek-Zähler	nichts
24	FLUSH	keine	A=00 wenn Fehlerfrei
25	MOVE	HL=Zieladresse	HL=HL+BC+1
		DE=Quelladresse	DE=DE+BC+1
		BC=Länge	
26	TIME	C=00 lesen	nichts
		C=FF schreiben	
27	SELMEM	A=Banknummer	nichts
28	SETBNK	A=Banknummer	nichts
29	XMOVE	B=Zielbank	nichts
		C=Quellbank	
30	USERF	für den Benutzer reserviert	
31	RESERV1	für das Betriebssystem reserviert	
32	RESERV2		

Tabelle 15 Zusammenfaßung aller BIOS-Aufrufe

Funktion 59	lade Overlay
Eingabe	C=3BH
	DE=FCB-Adresse
Zurück	A=Fehlercode
	H=Fehlernummer

Mit dieser Funktion kann eine overlay-Datei an einen (fast) beliebigen Speicherplatz innerhalb der TPA geladen werden.

Diese Funktion kann nur von Programmen benutzt werden, die mit GENCOM einen RSX-Kopf geladen haben, da die Funktion nur vom LOADER aus unterstützt wird, der seinerseits, nur wenn ein RSX geladen wird, resident bleibt.

Mit dieser Funktion können COM- oder PRL-Dateien als overlay geladen werden.
Unter einer overlay-Datei ist eine Datei zu verstehen, die nicht (unbedingt) nach Adresse 100H geladen wird, sondern nach einer im Dateienkopf (Header) angegebenen Adresse.

Overlay-Dateien sind im allgemeinen Hilfs-Dateien, die von einem ablauffähigen Programm bei Bedarf nachgeladen werden. Ein typisches Beispiel hierfür sind die OVR-Dateien von Wordstar (R).

Unter PRL-Dateien versteht man Programmsegmente, die so gelinkt wurden, daß sie auf jede beliebige gerade Seite transferiert werden können und dort ablauffähig sind. Unter Seite ist dabei (CP/M-typisch) jede Adresse zu verstehen, die ein Vielfaches von 100H ist. (z.B. 1400H, 3900H, E000H usw.) Diese Relozierung ist möglich durch ein dem Programm angehängtes Datenfeld, welches dem LOADER alle notwendigen Relozierungsdaten zur Verfügung stellt.

Vor dem Laden eines overlay-Programmes muß der zugehörige FCB erfolgreich eröffnet werden. Die Zieladresse der Datei wird dann in den Bytes r0 und r1 des FCB's eingetragen. Der LOADER meldet einen Fehler, wenn die Zieladresse unter 100H liegt oder das Programm den LOADER überlagern würde.

Die Funktion kehrt nach erfolgreichem Laden mit Register A=00 zum Aufrufer zurück. Die geladene Datei wird nicht aus dieser Funktion heraus ausgeführt.

Wenn bei Aufruf der Datei der LOADER (d.H. ein RSX) nicht gefunden werden kann, kehrt Register A=FF zurück.

Entdeckt der LOADER eine ungültige Zieladresse oder steht zuwenig Speicherraum zur Verfügung, um das Programm zu laden, dann kehrt Register A=FE zurück.

Im Fehlerfall A=FF meldet Register H noch folgende Fehler:

H=00	LOADER nicht gefunden
H=01	Lesefehler (DISK-I/O)
H=04	Ungültiges Laufwerk.

Funktion 60	Aufruf eines RSX
Eingabe	C=3CH
	DE=RSXPB-Adresse
Zurück	A=Fehlercode
	H=Fehlernummer

Diese spezielle Funktion ermöglicht es dem Programmierer Unterprogramme in einem (bereits geladenen) RSX aufzurufen.

Der Zugriff erfolgt über das RSXPB-Datenfeld (RSX-Parameter-Block), mit folgender Bedeutung:

```
1
2 0000' 00        RSXPB:    ds   1            ; RSX-Funktionsnummer
3 0001' 04                  db   p_Anzahl     ; Anzahl der folgenden Parameter
4 0002' 0010      p_start:  dw   parameter1   ; Vektor des Unterprogrammes 1
5 0004' 0020                dw   parameter2   ; Vektor des Unterprogrammes 2
6 0006' 0030                dw   parameter3   ; Vektor des Unterprogrammes 3
7 0008' 0040                dw   parameterN   ; letzter Vektor
8       0004      p_Anzahl  equ  ($-p_start)/2 ; Z(nlen )bernimmt der Assembler
```

Wenn die genannte Funktion (d.h. Nummer des folgenden Parametervektors) nicht mit dem RSXPB übereinstimmt (dort nicht genannt ist), kehrt die Funktion mit Register A=FF zurück, andernfalls ist Register A=00 .

Als Funktionsnummern sind 0..127 (0..7FH) für den Programmierer zugelassen. Die Nummern 128..255 (80..FFH) sind für das Betriebssystem reserviert.

Funktion 98	Freigabe belegter Blöcke
Eingabe	C=62H
Zurück	A=Fehlercode
	H=Fehlernummer

Mit dieser Funktion können Zwischendateien wieder freigegeben (d.h. im Inhaltsverzeichnis gelöscht) werden.

Einige Programme legen Zwischendaten (temporary) auf Diskette ab, wenn der Arbeitsspeicher in der TPA zu klein wird. Zwischendateien dieser Art werden zwar mit Funktion 16 geschlossen, wurden jedoch durch Setzen von Bit 7 im 5. Buchstabens des Dateinamen markiert.

Bei Aufruf der Funktion werden alle Dateien in allen initialisierten Laufwerken überprüft.

Diese Funktion wird vom CCP nach jedem Warmstart aufgerufen. Es ist DAHER notwendig, daß alle (noch benötigten) Dateien die geöffnet sind, vor einem Warmstart korrekt geschlossen werden.

Wurde die Funktion fehlerfrei abgeschlossen, kehrt sie mit Register A=00 zurück, andernfalls ist A=FF und Register H=04.

Funktion 100	Direktory Label setzen
Eingabe	C=64H
	DE=FCB-Adresse
Zurück	A=Fehlercode
	H=Fehlernummer

Mit dieser Funktion kann einer Diskette ein Namen (Label) zugewiesen oder ein bestehender Name geändert werden.

Der gewünschte Diskettenname wird in einem FCB übergeben, der den üblichen Regeln für den Aufbau eines Dateinamen entsprechen muß.

In Byte 0 des FCB's wird das anzusprechende Laufwerk definiert.

Das EX-Feld kann einen zusätzlichen code mit folgender Bedeutung enthalten:

Bit 7 Es wird ein Paßwort für entsprechend geschützte Dateien benötigt.

Bit 6 Es sind Datums- und Zeitmarken bei jedem Dateienzugriff einzutragen (access).

Bit 5 Es sind Datums- und Zeitmarken nach jeder Dateiveränderung einzutragen (update).

Bit 4 Es sind Datums- und Zeitmarken für jede neu generierte Datei einzutragen (create).

Bit 0 Dem Directory Label wird ein neues Paßwort zugewiesen.

Gesetzt wird eine Funktion, wenn das entsprechende Bit gesetzt (=1) ist.

Ist der Diskettenname (Label) bereits durch ein Paßwort geschützt, muß bei jedem Aufruf dieser Funktion dieses Paßwort in die ersten acht Bytes der momentanen DMA geschrieben werden, bevor die Funktion aufgerufen wird. Dies ist nicht notwendig, wenn zuvor mit Funktion 106 das Paßwort als Vorgabewert (default) deklariert wurde.

Wird der Diskette ein neues Paßwort zugewiesen (Bit 0=1), so wird dieses in den zweiten 8 Bytes der momentanen DMA-Adresse erwartet.

Die Benutzung dieser Funktion setzt ein erweitertes Inhaltsverzeichnis der fraglichen Diskette voraus. Dieses erweiterte Inhaltsverzeichnis wird mit dem INITDIR-Befehl generiert.

Es ist nicht zugelassen die Funktions-Bits 6 und 4 gleichzeitig zu setzen (siehe hierzu den SET-Befehl).

Die Funktion kehrt mit Register A=00 zum Aufrufer zurück, wenn sie fehlerfrei erledigt wurde.

Im Fehlerfalle meldet Register A=FF mit einer genaueren Fehlerangabe in Register H mit folgender Bedeutung:

H=00	Inhaltsverzeichnis voll oder nicht mit INITDIR erweitert.
H=01	Schreib- oder Lesefehler (Disk-I/O)
H=02	Diskette ist schreibgeschützt
H=04	ungültiges Laufwerk
H=07	ungültiges Paßwort

Funktion 101	Daten des Directory Label lesen
Eingabe	C=65H
	E=Laufwerksnummer
Zurück	A=Datenbyte des Labels
	H=Fehlernummer

Mit dieser Funktion kann das EX-Feld (Bit 12) des einer Diskette zugewiesenen Namens-FCB zurückgelesen werden.

Dies gilt für die Diskette in dem durch Register <E> spezifizierten Laufwerk, wobei Laufwerk A=00, B=01 usw ist.

Das zurückgegebene Datenbyte enthält eine Verschlüsselung mit folgender Bedeutung:

Bit 7 Es wird ein Paßwort für entsprechend geschützte Dateien benötigt.

Bit 6 Es sind Datums- und Zeitmarken bei jedem Dateienzugriff einzutragen (access).

Bit 5 Es sind Datums- und Zeitmarken nach jeder Dateiveränderung einzutragen (update).

Bit 4 Es sind Datums- und Zeitmarken für jede neu generierte Datei einzutragen (create).

Bit 0 Dem Directory Label wird ein neues Paßwort zugewiesen.

Gesetzt wird eine Funktion, wenn das entsprechende Bit gesetzt (=1) ist.

Kehrt die Funktion mit Register A=00 zurück, bedeutet dies, daß der angesprochenen Diskette kein Name zugewiesen wurde.

Kehrt die Funktion mit Register A=FF zurück, ist ein Fehler aufgetreten, der in Register H mit folgender Bedeutung näher bezeichnet wird:

H=01 Lesefehler (Disk-I/O)
H=04 ungültiges Laufwerk

Funktion 102	Lesen der Datums- und Zeitmarken und des Schutzmodus einer Datei
Eingabe	C=66H
	DE=FCB-Adresse
Zurück	A=Fehlercode
	H=Fehlernummer

Mit dieser Funktion kann das Datums- und Zeitmarkenfeld der Neugenerierung oder des letzten Zugriffes sowie der letzten Änderung der im FCB spezifizierten Datei zurückgelesen werden.

Darüberhinaus werden Informationen darüber zurückgegeben, inwieweit die Datei durch ein Paßwort (Kennwort) geschützt ist.

Die Daten sind nach erfolgreichem Aufruf der Funktion (Register A=00) dem entsprechend geänderten FCB zu entnehmen.

Der Modus des Paßwort-Schutzes steht im EX-Feld (Byte 12) des FCB's mit folgender Bedeutung:

Bit 7	Leseschutz
Bit 6	Schreibschutz
Bit 4	Datei kann nicht gelöscht werden

Ein Modus ist dann aktiv, wenn das entsprechende Bit gesetzt ist (=1).

Ist das RX-Feld auf Null gesetzt, bedeutet dies, daß der Datei kein Paßwort zugeordnet wurde.

Die Datums- und Zeitmarken sind die Bytes

24..27	für die Neugenerierung oder dem letzten Zugriff (create oder access) und
28..31	für die letzte Änderung

der Datei enthalten. Die entsprechenden Datenfelder sind auf Null zurückgesetzt, wenn keine Datums- oder Zeitmarken definiert wurden.

Das Format der Datums- und Zeitmarken ist unter Funktion 104 beschrieben.

Im Fehlerfall kehrt die Funktion mit Register A=FF zurück mit einer näheren Fehlerbeschreibung in Register H bei folgender Bedeutung:

H=00	Datei nicht gefunden
H=01	Lesefehler (Disk-I/O)
H=04	ungültiges Laufwerk
H=09	unzulässiger Joker im Dateiname

Funktion 103	schreiben eines erweiteren (X)FCBs
Eingabe	C=67H
	DE=FCB-Adresse
Zurück	A=Fehlercode
	H=Fehlernummer

Diese Funktion ermöglicht es einen erweiterten (eXtended) FCB zu generieren oder zu verändern.

Ein erweiterter FCB enthält wie gewohnt die Laufwerksbezeichnung und den Dateinamen mit Index. Das EX-Feld ist hier jedoch nicht auf Null gesetzt, sondern enthält nähere Angaben über den Paßwortschutz der angegebenen Datei.

Die einzelnen Bits dieses 12. Bytes im XFCB haben dabei folgende Bedeutung wenn sie auf '1' gesetzt sind:

Bit 7	Leseschutz
Bit 6	Schreibschutz
Bit 5	Datei kann nicht gelöscht werden
Bit 0	Zuweisung eines neuen Paßwortes für die Datei

Wenn die Datei bereits durch ein Paßwort geschützt ist, wird dieses Paßwort in den ersten 8 Bytes der momentanen DMA-Adresse erwartet.

Wird das Paßwort geändert, so wird das neue Paßwort in den zweiten 8 Bytes der momentanen DMA erwartet. Die Paßwortangabe ist nicht notwendig, wenn vor dem Aufrufen der Funktion mit Funktion 106 das Paßwort vorgegeben wurde.

Die Funktion kehrt mit Register A=00 zurück, wenn der XFCB erfolgreich eröffnet oder verändert werden konnte.

Andernfalls ist Register A=FF mit eine nähere Fehlerangabe in Register H mit folgender Bedeutung:

H=00	Datei nicht gefunden oder kein Paßwort zugewiesen.
H=01	Lese- oder Schreibfehler (Disk-I/O)
H=02	Diskette ist schreibgeschützt
H=04	ungültiges Laufwerk
H=07	falsches Paßwort
H=09	nicht zugelassener Joker im Dateinamen

Funktion 104	Datum und Zeit setzen
Eingabe	C=68H
	DE=DTF-Adresse

Mit dieser Funktion ist es möglich Datums- und Zeitangaben in den SCB zu übernehmen.

Das Registerpaar <DE> muß beim Aufrufen der Funktion auf ein Datenfeld (Date-Time-Field) zeigen, in welchem die Uhrzeit und das Datum wie folgt abgelegt sind:

```
1
2 0000' 0000      DTF:    ds   2          ; Datum als Offset vom 1.1.78
3 0002' 00                ds   1          ; Stunde (24h) in BCD
4 0003' 00                ds   1          ; Minute 0..59 in BCD
```

Die Angaben zum Datum sind als 16-Bit Wort (in HEX) zu setzen als Offset zum 1. Januar 1978. Dieses Datum entspräche dem Wert 0, der 10. Januar 1978 dem Wert 0AH usw.

Es sind CP/M 3 Versionen auf dem Markt, in welchen das Datumsfeld im SCB (System-Controll-Block) bereits auf einen anderen Zeitpunkt voreingestellt ist. Dieser Zeitpunkt kann leicht durch Aufruf des DATE-Befehles festgestellt werden (falls keine Uhr installiert ist).

Die Angaben zur Stunde und Minute erfolgen als BCD-Werte, also 13 für 13 Uhr oder 59 für 59 Minuten.

Beim Aufrufen der Funktion wird das Sekundenfeld im SCB auf 00 gesetzt. Soll das Sekundenfeld ebenfalls gesetzt werden, so kann dies nur über Funktion 49 geschehen.

Aus dem CCP heraus sind alle Datums- und Zeitmarken mit dem DATE-Befehl setzbar.

Funktion 105	lesen des Datums und der Zeit
Eingabe	C=69H
	DE=DTF-Adresse
Zurück	A=Sekunden
	DTF gesetzt

Mit dieser Funktion kann das Datum und die momentane Uhrzeit gelesen werden.

Die Datenübergabe erfolgt im DTF-Datenfeld. (siehe hierzu Funktion 104)

In Register A werden zusätzlich zu den Daten im DTF-Datenfeld noch Angaben über den aktuellen Sekundenstand, ebenfalls in BCD, zurückgegeben.

Funktion 106	Vorgabepaßwort setzen
Eingabe	C=6AH
	DE=Paßwortadresse

Mit dieser Funktion ist es möglich, das Paßwort vor dem Zugriff auf paßwortgeschützte Dateien vorzugeben.

Dies erleichtert dem Paßwort-Besitzer die Arbeit erheblich, da das Paßwort nicht immer wieder neu eingegeben werden muß. Es ist jedoch zu berücksichtigen, daß nach Aufruf dieser Funktion der Paßwortschutz quasi aufgehoben ist, bis ein Warmstart oder Diskettenwechsel stattgefunden hat.

Beim Aufruf der Funktion muß das Registerpaar DE auf ein 8 Byte Datenfeld zeigen, welches das gültige Paßwort enthält.

Ist das angegebene Paßwort ungültig, dann bleibt der Dateischutz erhalten.

Funktion 107	lese Seriennummer
Eingabe	C=6BH
	DE=SNF-Adresse

Mit dieser Funktion kann die Seriennummer des benutzten BDOS in ein 6 Byte (Serien Nummer Feld)-Datenfeld gelesen werden.

Achtung:

Diese Funktion funktioniert im nichtgebankten CP/M 3 wie sie soll, im gebankten BDOS können jedoch Probleme auftreten.

Hier scheint ein Fehler im BDOS vorzuliegen - aber wer braucht diese Funktion schon .....

Funktion 108	lesen oder setzen eines Rückkehr-codes bei Programmende
Eingabe	C=6CH DE=FFFFH (lesen) oder DE=Programmcode (schreiben)
Zurück	HL=Programmcode (falls lesen)

Diese Funktion ermöglicht es dem Benutzer einen 16-Bit-Code einem danach aufgerufenen Programm zu übergeben. (siehe hierzu auch Funktion 47).

Folgende Vorgaben stehen dabei zur Verfügung:

0000..FEFFH	erfolgreicher Programmabschluß.
FF00..FFFEH	Programm wurde mit Fehler abgeschlossen wobei folgendes gilt:
FF80..FFFCH	reserviert vom Betriebssystem.
FFFDH	Programm wurde wegen nicht behebbarem BDOS-Fehler abgebrochen.
FFFEH	Programm wurde durch CNTRL-C abgebrochen.
0000H	wird vom CCP automatisch eingefügt, wenn das nächste Programm nicht mit Funktion 47 aufgerufen wird.
FFFFH	reserviert für Datenabfrage.

Alle nicht reservierten Kombinationen können zu beliebiger Datenübergaben zwischen verketteten Dateien benutzt werden.

Funktion 109	Konsolenmodus setzen oder lesen
Eingabe	C=6DH
	DE=FFFFH (lesen) oder
	DE=Modus (setzen)
Zurück	HL=Modus (wenn lesen)

Mit dieser Funktion kann die Konsole auf verschiedene Art der Dateneingabe gesetzt oder die gesetzten Werte zurückgelesen werden.

Die einzelnen Bits des übergebenen Datenwortes haben folgende Bedeutung:

Bit 0 =1 Der mit Funktion 11 aufgerufene Status kehrt nur bei CNTRL-C mit dem Wert 01H zurück.

=0 Mit Funktion 11 wird bei jeder Zeicheneingabe mit 01H zurückgekehrt.

BIT 1 =1 Bildschirmausgabe wird mit CNTRL-S angehalten und erneut freigegeben (CP/M-2 Modus).

=0 Bildschirmausgabe wird mit CNTRL-S angehalten und mit CNTRL-Q wieder freigegeben.

BIT 2 =1 ungefilterte Konsolenausgabe. Es wird die Expansion von TAB's in den Funktionen 2,9 und 111 unterdrückt, ebenso die Unterstützung des Mitdruckens mit CNTRL-P.

=0 Normale Konsolenausgabe wie unter Funktion 2 angegeben.

BIT 3 =1 Programme können nicht mit CNTRL-C abgebrochen werden.

=0 Programme können mit CNTRL-C abgebrochen werden.

BIT 4..7 reserviert vom Betriebssystem.

BIT 8,9 Diese Bits geben an, wie der Konsolenstatus definiert wird, wenn ein RSX benutzt wird um den Konsoleneingang (auf andere Kanäle) umzulenken:

8=0-9=0 Wird der Konsolenstatus aufgerufen, so kehrt die Funktion immer mit A=00 zurück, außer wenn dem ersten Aufruf sofort ein zweiter Aufruf folgt.

8=0-9=1 Der Status ist immer A=00

8=1-9=0 Der Status ist immer A=FF

8=1-9-1 Umlenken der Konsoleneingabe nicht zugelassen

Die restlichen Bits werden nicht benutzt. Der Konsolenmodus wird vom CCP vor dem Laden jedes Programmes auf 0 gesetzt, außer ein RSX setzt Funktion 109 außer Kraft.

Funktion 110	Lesen oder Setzen des Endezeichens bei der Textausgabe
Eingabe	C=6EH
	DE=FFFFH (lesen)
	E=Endezeichen (setzen)
Zurück	A=Endezeichen (beim Lesen)

Mit dieser Funktion kann die Textende-Marke neu definiert werden.

Mit Funktion 9 können beliebige Texte ausgegeben werden, bis das Textabschlußzeichen '$' erreicht wird.

Diese Zuweisung wird nach jedem Aufruf des CCP's neu deklariert. Sollen Texte ausgegeben werden, in welchem das Dollarzeichen '$' verwendet werden muß, so kann anstelle des Dollarzeichens ein anderes Zeichen definiert werden.

Funktion 111	Ausgabe eines Textblockes
Eingabe	C=6FH
	DE=CCB-Adresse

Diese Funktion ähnelt der Funktion 9, nur wird hier keine Textendemarke verwendet sondern dafür ein spezielles Datenfeld eingerichtet. (Character Controll Block). Dieses Datenfeld ist wie folgt definiert ist:

```
1
2 xxxx' xxxx      CCB:    ds   2      ; hier ist die Adresse bei der der
3                                     ; Text beginnt
4                         ds   2      ; hierher Laenge des Textes
```

Mit dieser Funktion ist es möglich z.B. Texte in einem Block zu schreiben und durch geschicktes Auslegen des CCB's aus diesem Textblock Ausschnitte zur Mehrfachverwendung auszuwählen.

Funktion 112	Drucken eines Textblockes
Eingabe	C=70H
	DE=CCB-Adresse

Diese Funktion entspricht der Funktion 111 nur wird die Ausgabe direkt auf den Drucker (LST-Device) ausgegeben.

Funktion 152	Generieren eines FCB's
Eingabe	C=98H
	DE=FCBPB-Adresse
Zurück	HL=Fehlercode

Mir dieser Funktion ist es möglich die Generierung eines FCB's dem BDOS selbst zu überlassen. Bei Aufruf der Funktion muß das Doppelregister DE lediglich auf eine Datenfeld zeigen (FCB-Parameter-Block), in welchem folgende Zeiger einzutragen sind:

```
1
2 0000' 0000        FCBPB:  ds   2      ; Adresse des Textpuffers der den
3                           ;            Dateinamen enthält
4 0002' 0000                ds   2      ; Adresse des (32-Bytes) Puffers
```

Der Textpuffer soll den Dateinamen in folgender Form enthalten:

```
(d:)dateiname(.typ)(;kennwort)
```

Es kann Groß- oder Kleinschreibung (auch gemischt) verwendet werden. Der Dateiname kann 1..8 Zeichen lang sein, der Index 0..3 Zeichen und das Paßwort ebenfalls 0..8 Zeichen.

Der Inhalt des Textpuffers kann vom Programmierer vorgegeben oder z.B. mit Funktion 10 vom Benutzer eingegeben werden.

Bei der Namenseingabe ist zu berücksichtigen, daß einige Zeichen als Textendemarke erkannt werden, d.h. Text, der auf eine Textendemarke folgt wird nicht mehr interpretiert. Leerzeichen oder TAB's am Anfang des Puffers werden unterdrückt. Die infrage kommenden Textendemarken sind nachfolgend aufgelistet:

```
Leerzeichen (blank)
TAB (Tabulator)
RETURN (cr)
NULL (00H)
; (Semikolon) ausser vor einem Kennwort (Passwort)
= (Gleichheitszeichen)
< und > (groesser oder kleiner als ...)
: (Doppelpunk) ausser nach der Laufwerksangabe
, (Komma)
| (Senkrechter Strich)
[ und ] (eckige Klammern)
```

Wenn die erste gefundene Textendemarke eine NULL oder ein RETURN ist, kehrt die Funktion mit dem Doppelregister HL=0000 zurück. Wird eine andere Textendemarke gefunden, kehrt HL mit der Adresse dieser Marke zurück.

Konnte kein gültiger Dateiname generiert werden, wird HL=FFFF übergeben.

Der aufgebaute FCB entspricht dem nachfolgenden Listing:

```
fcb:    db      drive           ; Laufwerksnummer 00=gerade gueltiges
                                ; Laufwerk, 01=Laufwerk A, 01=Laufwerk B usw
        dw      'FILENAME'      ; Dateiname (im FCB in Grossbuchstaben)
                                ; 1 Zeichen minimal 8 Zeichen maximal
                                ; Unbenutzte Stellen mit 20H (Leerzeichen)
                                ; auffuellen
        dw      'EXT'           ; Index des Dateinamens (in Grossbuchstaben(
                                ; 0 Zeichen minimal 3 Zeichen maximal
                                ; Unbenutzte Stellen mit 20H (Leerzeichen)
                                ; auffuellen
        db      0,0,0,0,        ; Mit 00 fuellen (interne Zeiger)
        dw      'KENNWORT'      ; Namen des Kennwortes (in Grossbuchstaben)
                                ; maximal 8 Zeichen. Auffuellen mit 20H fuer
                                ; nicht benutztes Kennwort oder nicht benutzte
                                ; Stellen.
        ds      8,0             ; benutzt vom Betriebssystem
```

Das Programm selbst fängt die Eingabe von CNTRL-C ab, um einen Warmstart in den CCP zu ermöglichen (und nicht einen Kaltstart über RESET zu benötigen).

```
0000'           contest:
                        ;
                        ; Unterprogramm zum Testen von software zur Konsolenunterstuetzung
                        ;
0000' 3E FD             ld      a,0fdh          ; direkte Zeicheneingabe
0002' CD 000F'          call    console         ; verlangen
0005' FE 03             cp      03H             ; CNTRL-C ?
0007' CA 0000           jp      z,0             ; wenn ja dann Warmstart
000A' CD 000F'          call    console         ; sonst Zeichen ausgeben
000D' 18 F1             jr      contest         ; und alles von vorne ...

000F' 0E 06     console:ld      c,6             ; Funktion 6
0011' 5F                ld      e,a             ; Zeichenuebergabe in Register E
0012' CD 0005           call    5               ; BDOS-Aufruf
0015' C9                ret
```

Die Anwendung einer der CP/M 3 typischen neueren Funktionen soll das nächste Beispiel verdeutlichen.

Hier wird mit Funktion 111 Text ausgegeben, der keine Textendemarke hat wie dies von Funktion 9 benötigt wird. Das Beispiel zeigt auch, daß es mit dieser Funktion möglich ist, beliebige Ausschnitte aus einem Textblock auszuwählen.

Der Funktion wird in einem Kontrollblock vor jedem Aufruf ein Zeiger auf den Text-Anfang und die Textlänge übergeben.

```
0000'           test:
                        ;
                        ; Ausgabe von Textbloecken auf die Konsole
                        ;
0000' CD 002D'          call    sm              ; Ausgabe des Textes wie definiert
0003' 21 0061'          ld      hl,text3        ; Zeiger auf den Text 'NIE'
0006' 22 0036'          ld      (ccb),hl        ; in den CCB
0009' 21 0003           ld      hl,3            ; 3 Zeichen
000C' 22 0038'          ld      (ccb+2),hl      ; ebenfalls in den CCB
000F' CD 002D'          call    sm              ; nun diesen Text ausgeben
0012' 21 004F'          ld      hl,text2        ; Zeiger auf den Text 'VERZAGT ...'
0015' 22 0036'          ld      (ccb),hl
0018' 21 0009           ld      hl,9            ; 9 Zeichen (incl Leerzeichen)
001B' 22 0038'          ld      (ccb+2),hl
001E' CD 002D'          call    sm
0021' 21 003A'          ld      hl,text1        ; Zeiger auf 'EIN MEGER ...'
0024' 22 0036'          ld      (ccb),hl
0027' 21 0009           ld      hl,9            ; 9 Zeichen ausgeben
002A' 22 0038'          ld      (ccb+2),hl
```

```
22
23                              ;
24                              ; Aufruf der Blockausgaberoutine
25                              ;
26
27 002D' 0E 6F          sm:     ld      c,111
28 002F' 11 0036'               ld      de,ccb
29 0032' CD 0005                call    5
30 0035' C9                     ret
31
32                              ;
33                              ; Character Controll Block
34                              ;
35
36 0036' 003A'          ccb:    dw      text1
37 0038' 002C                   dw      len1
38
39                              ;
40                              ; Textbereich
41                              ;
42
43 003A' 45 69 6E 20 text1:  db      'Ein Neger mit Gazelle'
44 004F' 20 76 65 72 text2:  db      ' verzagt im Regen '
45 0061' 6E 69 65 0D text3:  db      'nie',0dh,0ah
46         002C         len1    equ     $-text1
47
48                              end
```

Die Generierung eines RSX-Programmes.

Der Übersichtlichkeit halber ist das folgende Beispiel sehr einfach gehalten. Das Programm soll lediglich bei der Druckerausgabe die Ausgabe des Kontrollzeichens 0AH (linefeed) verhindern.

```
0000' 00 00 00 00       db   0,0,0,0,0,0        ; Platz fuer CP/m Seriennummer
0006' C3 001B'          jp   RSXPROG            ; Einsprung in RSX-Programm
0009' C3 0000     next: jp   $-$                ; Einsprung in's BIOS
000C' 0000        prev: dw   0                  ; Adresse vorheriges Modul
000E' FF          remov: db  0FFH               ; 0FFH=ja 000H=nein
000F' 00          nonbnk:db  0                  ; 0FFH=nicht gebankt 0=gebankt
0010' 52 53 58 20       db   'RSX-NAME'         ; RSX-Name
0013' 00 00 00          db   0,0,0              ; CP/M reserviert

001B'             RSXPROG:

001B' 79                ld     a,c              ; BIOS-Befehl in <C>
001C' FE 05             cp     5                ; LIST
001E' 20 02             jr     nz,done          ; ... wenn nicht LIST
0020' 7B                ld     a,e              ; ASCII-Zeichen
0021' C8                ret    z                ; ..LF nicht drucken

                        ;
                        ; LF wird in <A> dem BIOS-Caller zurueckgegeben
                        ; ( RET Z ) oder 'Rest' wird gedruckt
                        ;

0022' C3 0009'    done: jp     NEXT

                        end
```

Dieses Programm muß nun noch CP/M 3-gerecht aufbereitet werden:

```
A>M80 rsxname,=rsxname
A>LINK rsxname[B]
A>REN rsxname.rsx=rsxname.spr
A>GENCOM top rsxname
A>TOP
```

Zuerst muß das Programm assembliert werden. (Die Syntax kann hier bei Verwendung eines anderen Assemblers anders sein - es wird auf jeden Fall die Generierung einer REL-Datei erwartet).

Mit dem CP/M-Programm LINK.COM wird nun eine SPR-Datei generiert. Darunter ist eine Datei zu verstehen, die aus einer REL-Datei eine Datei generiert, die vom LOADER auf jede beliebige Seite reloziert werden kann. Unter Seite ist dabei jede Adresse zu verstehen die ein ganzzahliges Vielfaches der Adresse 100H ist. (Also z.B. 4500H oder 0EA00H). Ermöglicht wird diese Relozierung (Verlagerung mit Adressanpassung) durch eine mitgenerierte Tabelle, aus welcher der LOADER erkennen kann, welches Byte im Programm an die Zieladresse angepaßt werden muß.

Zur Verwendung der Datei mit dem Hilfsprogramm GENCOM muß der Index .SPR in .RSX umbenannt werden. Die Inhalt der Datei ändert sich dabei selbstverständlich nicht. Die Umbenennung ist aber notwendig, da eine RSX-Datei wie aus dem Listing zu erkennen ist, einen sog. HEADER (Kopf) hat, Eine normale SPR Datei hat diesen aber nicht. Der RSX-Index dient so der Erkennung und verhindert eine falsche Anwendung der Datei.

Mit GENCOM wird nun die RSX-Datei z.B. mit dem Textprozessor TOP.COM 'verbunden'.

Schaut man sich nach dem Anhängen des RSX-Programmes das 'neue' TOP.COM mit z.B. DUMP an, so stellt man fest, daß der erste Befehl nun 'C9' lautet - RET(urn). Dies würde unter CP/M 2 immer zu einem Warmstart führen. Dort könnte ein RSX-Programm also keinesfalls funktionieren, genauergesagt nicht einmal aufgerufen werden.

Unter CP/M 3 prüft der LOADER nun nach, ob es sich bei dem gerade geladenen Programm um ein RSX-Programm handelt. Ist dies der Fall, wird der RSX-Teil vor den LOADER transferiert und in das System eingebunden. Dabei wird die Sprungadresse des BDOS-Einsprunges auf Adresse 0005H auf die Einsprungadresse des RSX-Programmteiles abgeändert. (Sie zeigt im Beispiel also auf die Adresse auf welcher die Anweisung 'jp RSXPROG' steht). Danach wird im Programm der RSX-Anteil 'abgehängt' und das eigentliche Programm an seine Zieladresse (100H) nach vorne transferiert und ausgeführt.

Jeder Aufruf der Adresse 0005H führt nun unweigerlich über den RSX. Dort wird in unserem Beispiel geprüft, ob in Register C die Funktionsnummer 5 (Drucken des Zeichens in Register E) steht. Ist dies nicht der Fall, wird zum regulären BDOS weitergesprungen. Ist es Funktion 5, wird geprüft, ob das Zeichen in <E> ein 0AH (linefeed) ist, wenn nicht wird ebenfalls zum BDOS weiter gesprungen. Andernfalls wird das Zeichen nach Register <A> umgeladen und dem Aufrufer unbenutzt zurückgegeben.

Soll der RSX von unserem TOP-Programm wieder ab'gehängt werden, genügt die einfache Eingabe:

```
A>GENCOM TOP <cr>
```

Als weiteres Beispiel ist ein Programm aufgeführt, welches die Benutzung der Diskettenfunktionen verdeutlichen soll.

Aufgabe des Programmes DUMP (das im Übrigen nicht mit dem mitgelieferten DUMP.COM identisch ist) ist es, den Inhalt einer Datei auf dem Bildschirm in HEX- und ASCII-Darstellung auszugeben.

Im Beispiel wird die Tatsache ausgenutzt, daß der CCP den Speicherraum 0000H..00FFH zur Ablage von Dateinamen und als Puffer benutzt.

Der folgende Speicherausdruck zeigt z.B. den Speicherinhalt von Adresse 0000..00FFH nachdem der Befehl

```
dump b:sid.com
```

eingegeben wurde, also die Aufforderung den Inhalt des Programmes SID.COM auf Laufwerk 'B' auf der Konsole auszugeben.

```
0000: C3 03 F3 FF 01 C3 00 02 FF FF FF FF FF FF FF FF  ................
0010: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
0020: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
0030: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
0040: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
0050: FF FF FF FF FF FF FF FF FF FF FF FF FF 02 53 49 44  ..............SID
0060: 20 20 20 20 20 43 4F 4D 00 00 00 00 00 00 20 20 20      COM ......
0070: 20 20 20 20 20 20 20 20 00 00 00 00 00 FF FF FF          ........
0080: 09 62 3A 73 69 64 2E 63 6F 6D 00 FF FF FF FF FF  .b:sid.com......
0090: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
00A0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
00B0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
00C0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
00D0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
00E0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
00F0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
```

Der Textinhalt nach dem Programmaufruf DUMP ist im Speicherbereich 81H abgelegt; abgeschlossen mit 00H.

In Speicherstelle 80H ist die Länge der (Rest-) Befehlszeile eingetragen (09H= 9 Zeichen). Ab Adresse 5CH ist nun die erste Zeile eines FCB's eingetragen, beginnend mit der Laufwerksnummer. Im Beispiel ist dies 02H entsprechend Laufwerk 'B'. Laufwerk A wäre 01H. Wäre eine 00H eingetragen, so würde damit das derzeitig gültige Laufwerk gemeint sein (das default-Laufwerk), falls es nicht explizit in der Befehlszeile eingegeben würde.

Ab Adresse 5DH steht nun der Dateiname, umgewandelt wie von CP/M benötigt in Großbuchstaben und auf 8 Stellen mit Leerzeichen aufgefüllt.

Ab Adresse 65H beginnt der Index, ebenfalls in Großbuchstaben. Die darauffolgenden 4 Bytes sind auf NULL gesetzt und repräsentieren das EX-Feld, Das S1- und S2-Feld sowie das RC-Feld.

Zu einem kompletten FCB gehören nun noch die Felder D0..Dn und das CR-Feld. Der Inhalt dieser Felder wird beim Eröffnen einer Datei vom BDOS eingetragen, kann beim Programmaufruf also noch unberücksichtigt bleiben. Der CCP benutzt diesen Speicherraum

(von 6CH ... FFH) um wie oben gezeigt einen Rumpf-FCB einer zweiten Datei einzutragen, falls diese in der Befehlszeile angegeben wurde z.B. wie folgt:

```
A>myprog b:teil1.bin e:teil2.bin
```

In diesem Falle sähe der Speicherbereich wie so aus:

```
0050: FF FF FF FF FF FF FF FF FF FF FF FF 02 54 45 49  ..............TEI
0060: 4C 31 20 20 20 42 49 4E 00 00 00 00 05 54 45 49  L1   BIN.....TEI
0070: 4C 32 20 20 20 42 49 4E 00 00 00 00 00 FF FF FF  L2   BIN.........
0080: 17 62 3A 74 65 69 6C 31 2E 62 69 6E 20 65 3A 74  .b:teil1.bin e:t
0090: 65 69 6C 32 2E 62 69 6E 00 FF FF FF FF FF FF FF  eil2.bin........
```

Im Beispiel wird nun der bei 5CH beginnende FCB direkt ausgenutzt und der Bereich 80..FFH als Puffer verwendet. Dieser Bereich ist bereits vom CCP als Vorgabe DMA initialisiert. Ebenso wie der CCP Funktion 44 mit dem Vorgabewert 1 initialisiert hat.

Dies bedeutet, daß jeder Leseaufruf 128 Byte (=80H) in den vorgegebenen Puffer schreibt und zwar solange, bis das Dateiende erkannt wird.

Im Beispiel werden keine anderen Fehlermöglichkeiten abgefragt. Wenn Register A nicht NULL ist, wird davon ausgegangen, daß das Dateiende ereicht wurde.

```
1                ; ***************************************************************
2                ; *                                                             *
3                ; * DUMP   - Hilfsprogramm zur Darstellung des Dateieninhaltes  *
4                ; *          in HEX und ASCII auf der Konsole                   *
5                ; *          Geschrieben von Raoul O. Koerber, Detmold          *
6                ; ***************************************************************
7
8                        ;
9                        ; Vereinbarungen
10                       ;
11
12       0005    bdos    equ     0005h           ; BDOS Einsprungadresse
13       0100    tpa     equ     0100h           ; Beginn der TPA
14
15       005C    tfcb    equ     5ch             ; dort ist der FCB
16       0080    tsize   equ     80h             ; Puffergroesse
17       0080    tbuff   equ     80h             ; Pufferadresse
18
19                       ;
20                       ; ASCII-Zeichen
21                       ;
22
23       000D    cr      equ     0dh             ; RETURN
24       000A    lf      equ     0ah             ; linefeed
25       0020    blank   equ     20h             ; Leerzeichen
26
```

```
27                              ;
28                              ; Programmstart
29                              ;
30
31          0100                org     tpa
32
33 0100 3A 005D     dump:       ld      a,(tfcb+1)      ; ist eine Datei angegeben?
34 0103 FE 20                   cp      blank           ; dann darf hier kein 'blank' sein
35 0105 20 08                   jr      nz,dump1        ; weiter wenn nicht
36 0107 11 01C3                 ld      de,logo         ; sonst Syntax melden
37 010A 0E 09       sm:         ld      c,9             ; Textausgabe
38 010C C3 0005                 jp      bdos            ; ueber das BDOS und Warmstart
39
40 010F CD 015F     dump1:      call    open            ; Datei eroeffnen
41 0112 21 0100                 ld      hl,tpa          ; Angenommene Startadresse
42 0115 CD 01A1     loop:       call    reads           ; Sektor lesen
43 0118 B7                      or      a               ; Fehler ?
44 0119 C0                      ret     nz              ; dann Warmstart (zurueck)
45 011A CD 011F                 call    wbuff           ; Pufferhinalt ausgeben
46 011D 18 F6                   jr      loop
47
48 011F 06 08       wbuff:      ld      b,8             ; 8 Zeilen
49 0121 11 0050                 ld      de,tbuff        ; Pufferadresse
50 0124 CD 0131     wbuf1:      call    crlf            ; neue Zeile
51 0127 CD 0172                 call    sndhl           ; angenommene Startadresse ausgeben
52 012A EB                      ex      de,hl           ; Pufferadresse
53 012B CD 0138                 call    dline           ; Ausgabe einer Zeile des Pufferinhaltes
54 012E EB                      ex      de,hl           ; Pufferadresse rette
55 012F D5                      push    de              ; Pufferadresse retten
56 0130 11 0010                 ld      de,16h          ; Neue Adresse
57 0133 19                      add     hl,de           ; berechnen
58 0134 D1                      pop     de              ; Pufferadresse
59 0135 10 ED                   djnz    wbuf1
60 0137 C9                      ret
61
62 0138 D5          dline:      push    de              ; TPA-Adresse
63 0139 C5                      push    bc              ; Zaehler retten
64 013A E5                      push    hl              ; Zeiger retten
65 013B 06 08                   ld      b,8             ; 8* Wort ausgeben
66 013D 56          dlin1:      ld      d,(hl)          ; Inhalt erstes Byte
67 013E 23                      inc     hl
68 013F 5E                      ld      e,(hl)          ; Inhalt zweites Byte
69 0140 23                      inc     hl              ; Zeiger auf naechstes Byte
70 0141 EB                      ex      de,hl           ; Register tauschen
71 0142 CD 0172                 call    sndhl           ; und Inhalt von HL ausgeben
72 0145 EB                      ex      de,hl           ; Register zurueck
73 0146 10 F5                   djnz    dlin1           ;  .. 8*
74 0148 E1                      pop     hl              ; Zeiger auf Zeilenanfang
75 0149 06 10                   ld      b,16            ; 16 Zeichen
76 014B 7E          dlin2:      ld      a,(hl)          ; Zeichen lesen
77 014C 23                      inc     hl              ; und Zeiger auf naechstes Zeichen
78 014D FE 7F                   cp      7fh             ; ASCII (Obergrenze?)
79 014F 30 04                   jr      nc,dlin3        ; wenn nicht
80 0151 FE 20                   cp      blank           ; ASCII (Untergrenze?)
81 0153 30 02                   jr      nc,dlin4
82 0155 3E 2E       dlin3:      ld      a,'.'           ; nicht Druckbares als Punkt ausgeben
83 0157 CD 0198     dlin4:      call    conout          ; Zeichen ausgeben
84 015A 10 EF                   djnz    dlin2           ; 16*
85 015C C1                      pop     bc              ; Zaehler
86 015D D1                      pop     de
87 015E C9                     ret
```

```
                ;
015F AF         open:   xor     a               ; A=0
0160 32 007C            ld      (tfcb+32),a     ; RC loeschen
0163 11 005C            ld      de,tfcb         ; Zeiger auf FCB
0166 0E 0F              ld      c,15            ; Datei eroeffnen
0168 CD 0005            call    bdos            ; mit BDOS-Aufruf
016B B7                 or      a               ; Fehler ?
016C C8                 ret     z               ; ... wenn nicht
016D 11 01AC            ld      de,nofile       ; sonst melden 'Datei nicht ...'
0170 18 98              jr      sm
                ;
0172 7C         sndhl:  ld      a,h             ; zuerst <H>
0173 CD 017E            call    sh              ; in HEX ausgeben
0176 7D                 ld      a,l             ; dann <L>
0177 CD 017E            call    sh
017A 3E 20              ld      a,blank         ; dann ein Leerzeichen anhaengen
017C 18 1A              jr      conout
                ;
017E F5         sh:     push    af              ; Zeichen retten
017F 0F                 rrca
0180 0F                 rrca
0181 0F                 rrca
0182 0F                 rrca                    ; Oberes nibble nach unten schieben
0183 CD 0187            call    htoa            ; und in HEX ausgeben
0186 F1                 pop     af              ; nun oberes nibble
                ;
0187 E6 0F      htoa:   and     00001111b       ; oberes nibble maskieren
0189 C6 90              add     a,90h           ; und in dezimal wandeln
018B 27                 daa                     ; ein bisschen kompliziert
018C CE 40              adc     a,40h           ; aber kurz und schnell
018E 27                 daa
018F 18 07              jr      conout
                ;
0191 3E 0D      crlf:   ld      a,cr
0193 CD 0198            call    conout
0196 3E 0A              ld      a,lf
                ;
0198 D9         conout: exx                     ; alle Register retten
0199 0E 02              ld      c,2             ; Konsolenausgabe
019B 5F                 ld      e,a             ; Zeichen
019C CD 0005            call    bdos
019F D9                 exx
01A0 C9                 ret
                ;
01A1 D9         reads:  exx                     ; alle Register retten
01A2 11 005C            ld      de,tfcb         ; FCB-Adresse
01A5 0E 14              ld      c,20            ; seq. lesen eines Sektors
01A7 CD 0005            call    bdos
01AA D9                 exx                     ; alle Register zurueck
01AB C9                 ret
                ;
01AC 0D 0A 44 61 nofile: db     cr,lf,'Datei nicht gefunden$'
01C3 0D 0A 44 55 logo:  db      cr,lf,'DUMP wird aufgerufen mit DUMP (d:)file(.ext)$'
                ;
                        end
```

Ein etwas 'ausgefalleneres' Programm.

Das letztes Beispiel soll noch einmal die Anwendung eines RSX-Programmes zeigen. Die Aufgabe dieses Programmes es ist, in einem CP/M 3 System Programme ablaufen zu lassen, die durch direkte Aufrufe der Diskettenunterprogramme im BIOS unter CP/M 3 eigentlich nicht funktionsfähig wären. Das Listing zeigt auch deutlich die Anwendung des BDOS Funktionsaufruf 50. Ebenfalls wird gezeigt (da es in diesem Falle emuliert werden muß), wie der blocking-, deblocking Algorithmus funktioniert.

Eingebunden werden kann dieses Programm z.B. in der älteren Ausführung von DU.COM oder DUTIL oder (zumindest für teilweise Benutzung) bei POWER.COM. Es sollte auch unter SUPERZAP.COM funktionieren.

Das Listing folgt ab der nächsten Seite.

Die gezeigten Beispiel sind sicherlich nicht ausreichend, um alle Möglichkeiten auszuschöpfen, die CP/M dem Programmierer bietet.

Die CP/M Users Group bietet jedoch dem Anfänger, dem Fortgeschrittenen und dem Profi eine Vielzahl von Programmen mit Source-Listings, die so ziemlich jedes Detail und jede Möglichkeit der Programmierung unter CP/M zeigen.

Im Übrigen ist es hier wie überall, nur die Übung macht den Meister und Probieren (Programmieren) geht über Studieren.

```
                ;###############################################################
                ; Dieses CP/M3-RSX erlaubt es 'alte' CP/M2 Programme die direkt
                ; auf das BIOS zugreifen, ablaufen zu lassen
                ;###############################################################

                        ;
                        ; ################
                        ; Vereinbarungen
                        ; ################
                        ;

        FFFF    ja              equ     -1
        0000    nein            equ     not ja

                        ;
                        ; ################
                        ; RSX Header
                        ; ################
                        ;

0000' 00 00 00 00               defs    6,0             ; wird ueberschrieben

0006' C3 007B'  entry:  jp      boot
0009' C3 0000   next:   jp      $-$             ; naechstes Modul
000C' 0000      prev:   defw    0               ; vorheriges Modul
000E' FF        remove: defb    ja              ; RSX loeschen bei Warmstart
000F' 00        nonbnk: defb    nein            ; nur ungebankt ?
0010' 52 53 58 32       defb    'RSX22  '       ; RSX-Titel
0018' 00 00 00          defb    0,0,0

                        ;
                        ; ################
                        ; BIOS(2.2) Sprungtabelle
                        ; ################
                        ;

001B' C3 009F'  cbt:    jp      wboot           ; Kaltstart
001E' C3 009F'  wbt:    jp      wboot           ; Warmstart
0021' C3 0051'          jp      xconst          ; Konsolen Status
0024' C3 0054'          jp      xconin          ; Konsolen Eingabe
0027' C3 0057'          jp      xconout         ; Konsolen Ausgabe
002A' C3 005A'          jp      xlist           ; Drucker
002D' C3 005D'          jp      xauxout         ; AUX Ausgabe
0030' C3 0060'          jp      xauxin          ; AUX Eingabe
0033' C3 006D'          jp      home            ; Disk home
0036' C3 00D5'          jp      seldsk          ; Disk anwaehlen
0039' C3 010D'          jp      settrk          ; Track setzen
003C' C3 011A'          jp      setsec          ; Sektor setzen
003F' C3 0112'          jp      setdma          ; DMA Adresse setzen
0042' C3 011F'          jp      read            ; Sektor lesen
0045' C3 012B'          jp      write           ; Sektor schreiben
0048' C3 0078'          jp      xliststt        ; Drucker Status
004B' C3 0117'          jp      sectran         ; Sektor umrechnen

                        ;
                        ; ################
                       ; BIOS (?) Sprungtabelle
```

```
                        ; ****************
                        ;
                        
004E' C3 0000   xwboot: jp      0
0051' C3 0000   xconst: jp      0
0054' C3 0000   xconin: jp      0
0057' C3 0000   xconout:jp      0
005A' C3 0000   xlist:  jp      0
005D' C3 0000   xauxout:jp      0
00E0' C3 0000   xauxin: jp      0
0063' C3 0000           jp      0
0066' C3 0000           jp      0
0069' C3 0000           jp      0
006C' C3 0000           jp      0
006F' C3 0000           jp      0
0072' C3 0000           jp      0
0075' C3 0000           jp      0
0078' C3 0000   xliststt:jp     0

                        ;
                        ; ****************
                        ; Kaltstart
                        ; ****************
                        ;

0076' F5        boot:   push    af              ; alle BIOS-Register
007C' D9                exx                     ; retten
007D' 21 0003'          ld      hl,next         ; und Einsprung
0080' 22 0007'          ld      (entry+1),hl    ; umdefinieren
0083' CD 00A6'          call    init            ; BIOS 'Ersatz'
0086' 2A 0001           ld      hl,(1)          ; alte Sprungadresse
0089' 22 02AC'          ld      (old$addr),hl   ;  retten
008C' 11 004E'          ld      de,xwboot       ; interne Sprungtabelle
008F' 01 002D           ld      bc,15*3         ; mit Originaladressen
0092' ED B0             ldir                    ; fuellen
0094' 21 001E'          ld      hl,wbt          ; neuer Warmstart
0097' 22 0001           ld      (1),hl          ; nach 0001 ablegen
009A' D9                exx                     ; Register zurueck
009B' F1                pop     af
009C' C3 0003'          jp      next

                        ;
                        ; ************************************************
                        ; Ab hier beginnt das 'neue' (CP/M-2) BIOS
                        ; ************************************************
                        ;


                        ;
                        ; ****************
                        ; Warmstart
                        ; ****************
                        ;

009F' 2A 02AC'  wboot:  ld      hl,(old$addr); zurueckspeichern des alten
00A2' 22 0001           ld      (1),hl       ; BIOS-einsprunges
00A5' E9                jp      (hl)

                        ;
```

```
                        ; ****************
                        ; interna
                        ; ****************
                        ;
                        ;
                        ; hier geht es hauptsaechlich darum den blocking-
                        ; deblocking Puffer zu initialisieren
                        ;

00A5' AF        init:   xor     a
00A7' 32 0298'          ld      (hstwrt),a  ; Hostpuffer an den Anfang
00AA' 32 0297'          ld      (hstact),a
00AD' 21 0080           ld      hl,80h      ; Standard DMA setzen
00B0' 22 02AA'          ld      (dmaadr),hl
00B3' C9                ret

                        ;
                        ; Unterprogramm um gebanktes BIOS ueber BDOS-call '50'
                        ; zu bearbeiten
                        ;

00B4' 0E 32     xbios:  ld      c,50        ; BDOS-Funktion
00B6' 11 02BE'          ld      de,biospb   ; Parameterblock
00B9' 12                ld      (de),a      ; BIOS-Funktion ablegen
00BA' C3 0009'          jp      next        ;  und Funktion ausfuehren

                        ;
                        ; ****************
                        ; home
                        ; ****************
                        ;

00BD' 3A 0298'  home:   ld      a,(hstwrt)  ; pruefe ob 'schreiben'
00C0' B7                or      a
00C1' C4 0241'          call    nz,writehst ; Dump puffer auf Diskette
00C4' AF                xor     a
00C5' 32 0298'          ld      (hstwrt),a  ; Puffer geschrieben
00C8' 32 0297'          ld      (hstact),a  ; Puffer inaktiv
00CB' 32 0299'          ld      (unacnt),a  ; loesche alloc-Zaehler
00CE' 32 028E'          ld      (sektrk),a  ;    und Trackzaehler
00D1' 32 028F'          ld      (sektrk+1),a
00D4' C9                ret

                        ;
                        ; ****************
                        ; Disk ansprechen
                        ; ****************
                        ;

00D5' 79        seldsk: ld      a,c         ; Laufwerksnummer
00D6' 32 028D'          ld      (sekdsk),a
00D9' 32 02C0'          ld      (bcreg),a   ; in BIOSFB ablegen
00DC' 3E 09             ld      a,9         ; BIOS Funktionsnummer
00DE' CD 00B4'          call    xbios
00E1' 7C                ld      a,h         ; Fehler pruefen
00E2' B5                or      l
00E3' C8                ret     z           ;   wenn Selectfehler
00E4' 5E                ld      e,(hl)      ; sonst Adresse der XLT-Tabelle
```

```
                        inc     hl
                        ld      d,(hl)          ; in dummy-DPH
                        ex      de,hl
                        ld      (xlat),hl       ; retten
                        ld      hl,11           ; Offset
                        add     hl,de
                        ld      e,(hl)
                        inc     hl
                        ld      d,(hl)
                        ex      de,hl
                        ld      (dpb),hl        ; dummy DPB
                        ld      a,(hl)          ; Sektoren/Track
                        ld      (spt),a
                        inc     hl
                        inc     hl
                        inc     hl              ; block shift mask
                        ld      a,(hl)
                        ld      (bsm),a         ; ablegen
                        ld      de,12           ; offset auf psh
                        add     hl,de
                        ld      a,(hl)          ; psh lesen
                        ld      (psh),a         ; und ablegen
                        ld      hl,dpn          ; Adresse des dummy
                        ret

                        ;
                        ; ****************
                        ; Track setzen
                        ; ****************
                        ;

settrk: ld      (sektrk),bc
                        ret

                        ;
                        ; ****************
                        ; DMA setzen
                        ; ****************
                        ;

setdma: ld      (dmaadr),bc
                        ret

                        ;
                        ; ****************
                        ; Sektor berechnen
                        ; ****************
                        ;
                        ; hier ist eine potentielle Fehlerquelle, falls Programme
                        ; sich die XLT-Tabelle selbst berechnen kann hier
                        ; ein Fehler 'uebergeben' werden....
                        ;

sectran:ld      l,c             ; Eingabesektor=Ausgabesektor
                        ld      h,b
                        ret

                        ;
```

```
                        ; ##################
                        ; Sektor setzen
                        ; ##################
                        ;
                
011A' 79        setsec: ld      a,c
011B' 32 0290'          ld      (seksec),a
011E' C9                ret
                
                        ;
                        ; ##################
                        ; Sektor lesen
                        ; ##################
                        ;
011F' 3E 01     read:   ld      a,1             ; lesen
0121' 32 02A7'          ld      (readop),a      ; ankuendigen
0124' 3C                inc     a               ; a=2 = unalloc
0125' 32 02A9'          ld      (wrtype),a
0128' C3 018E'          jp      alloc
                
                        ;
                        ; ##################
                        ; Sektor schreiben
                        ; ##################
                        ;
012B' AF        write:  xor     a               ; schreiben
012C' 32 02A7'          ld      (readop),a      ; ankuendigen
012F' 79                ld      a,c
0130' 32 02A9'          ld      (wrtype),a
0133' FE 02             cp      2               ; unalloc Block ?
0135' 20 19             jr      nz,chkuna
                
                        ;
                        ; schreibe in den 1. Sektor eines nicht belegten Blocks
                        ; das nachfolgende Programmsegment entspricht dem
                        ; (de)blocking wie unter CP/M 2 verwendet
                        ;
                
0137' 3A 02A3'          ld      a,(bsm)         ; block shift mask
013A' 3C                inc     a               ; Ausgleich
013B' 32 0299'          ld      (unacnt),a      ; Zaehler
013E' 3A 028D'          ld      a,(sekdsk)      ; und flags initialisieren
0141' 32 029A'          ld      (unadsk),a
0144' 3A 028E'          ld      a,(sektrk)
0147' 32 0298'          ld      (unatrk),a
014A' 3A 0290'          ld      a,(seksec)
014D' 32 029D'          ld      (unasec),a
                
0150' 3A 0299'  chkuna: ld      a,(unacnt)      ; unbelegte Sektoren
0153' B7                or      a               ; in diesem Block ?
0154' 28 38             jr      z,alloc         ; ... wenn nicht
0156' 3D                dec     a
0157' 32 0299'          ld      (unacnt),a
015A' 3A 028D'          ld      a,(sekdsk)
015D' 21 029A'          ld      hl,unadsk
0160' BE                cp      (hl)            ; sekdsk=unadsk?
```

```
291 0161' 20 2B          jr    nz,alloc    ; ...  wenn nicht
292 0163' 3A 028E'       ld    a,(sektrk)
293 0166' 21 0296'       ld    hl,unatrk
294 0169' BE             cp    (hl)        ; sektrk=unatrk ?
295 016A' 20 22          jr    nz,alloc    ; ...  wenn nicht
296 016C' 3A 0290'       ld    a,(seksec)
297 016F' 21 0290'       ld    hl,unasec
298 0172' BE             cp    (hl)        ; seksec=unasec?
299 0173' 20 19          jr    nz,alloc    ; ...  wenn nicht
300 0175' 34             inc   (hl)        ; naechster Sektor
301 0176' 7E             ld    a,(hl)
302 0177' 21 02A0'       ld    hl,spt
303 017A' BE             cp    (hl)        ; sektor=spt ?
304 017B' 38 0B          jr    c,noovf     ; ... wenn kein Ueberlauf
305 017D' 2A 0296'       ld    hl,(unatrk)
306 0180' 23             inc   hl
307 0181' 22 0296'       ld    (unatrk),hl
308 0184' AF             xor   a
309 0185' 32 0290'       ld    (unasec),a  ; reset Sektorzaehler
310 0188' AF     noovf:  xor   a
311 0189' 32 02A6'       ld    (rsflag),a  ; nicht 'vorher'-lesen
312 018C' 18 03          jr    rwoper
313
314 018E' AF     alloc:  xor   a           ; vorher lesen
315 018F' 32 0299'       ld    (unacnt),a
316 0192' 3C             inc   a
317 0193' 32 02A6'       ld    (rsflag),a
318
319 0196' AF     rwoper: xor   a
320 0197' 32 02A5'       ld    (erflag),a  ; keine Fehler bisher ...
321 019A' 3A 02A4'       ld    a,(psh)     ; physical shift factor
322 019D' B7             or    a
323 019E' 47             ld    b,a
324 019F' 3A 0290'       ld    a,(seksec)  ; logischer Sektor
325 01A2' 21 02C6'       ld    hl,hstbuf   ; 'eigener' Sektorpuffer
326 01A5' 11 0080        ld    de,128      ; log sektor Groesse
327 01A8' 28 0D          jr    z,noblk     ; ... kein blocking
328 01AA' EB             ex    de,hl
329 01AB' EB     shift:  ex    de,hl
330 01AC' 0F             rrca
331 01AD' 30 01          jr    nc,shl
332 01AF' 19             add   hl,de
333 01B0' EB     shl:    ex    de,hl
334 01B1' 29             add   hl,hl       ; verdopple offset
335 01B2' E6 7F          and   07fh
336 01B4' 10 F5          djnz  shift
337 01B6' EB             ex    de,hl       ; hl=Pufferadresse
338 01B7' 32 0256'   nobik: ld    (sekhst),a
339 01BA' 22 029E'       ld    (sekbuf),hl
340 01BD' 21 0297'       ld    hl,hstact   ; Puffer aktiv-flag
341 01C0' 7E             ld    a,(hl)
342 01C1' 36 01          ld    (hl),1      ; aktivieren
343 01C3' B7             or    a           ; bereits fertig
344 01C4' 28 22          jr    z,filnst    ; sonst Puffer fuellen
345 01C6' 3A 028D'       ld    a,(sekdsk)
346 01C9' 21 0291'       ld    hl,hstdsk   ; gleiches Laufwerk
347 01CC' BE             cp    (hl)
348 01CD' 20 12          jr    nz,nomatch  ; .. wenn nicht
```

```
                        ld      a,(sektrk)
                        ld      hl,hsttrk       ; gleicher Track
                        cp      (hl)
                        jr      nz,nomatch      ; .. wenn nicht
                        ld      a,(sekhst)
                        ld      hl,hstsec       ; gleicher Puffer ?
                        cp      (hl)
                        jr      z,match         ; .. wenn ja

nomatch:                ld      a,(hstwrt)      ; Puffer geandert ?
                        or      a
                        call    nz,writehst     ; .. Puffer loeschen

filhst:                 ld      a,(sekdsk)
                        ld      (hstdsk),a
                        ld      hl,(sektrk)
                        ld      (hsttrk),hl
                        ld      a,(sekhst)
                        ld      (hstsec),a
                        ld      a,(rsflag)      ; lesen ?
                        or      a
                        call    nz,readhst      ; wenn ja
                        xor     a
                        ld      (hstwrt),a      ; schreiben inaktiv

match:                  ld      hl,(dmaadr)
                        ex      de,hl
                        ld      hl,(sekbuf)
                        ld      a,(readop)
                        or      a               ; lesen oder schreiben
                        jr      nz,rwmove       ; ... wenn lesen
                        ld      a,1
                        ld      (hstwrt),a
                        ex      de,hl           ; hl=dma de=puffer

rwmove:                 ld      bc,128          ; log sektor Groesse
                        ldir                    ; transfer...
                        ld      a,(wrtype)
                        cp      1               ; DIR-Operation ?
                        jr      nz,exit         ; fertig
                        ld      a,(erflag)
                        or      a
                        jr      nz,exit         ; kein DIR falls
                        xor     a
                        ld      (hstwrt),a      ; flag Puffer geschrieben
                        call    writehst        ; ... schreiben
exit:                   ld      a,(erflag)      ; Fehlerflag
                        ret

                        ;
                        ; ****************
                        ; Sektor lesen (CP/M3)
                        ; ****************
                        ;

readhst:                call    rw$init
                        ld      a,13            ; BIOS READ
                        call    xbios
```

```
0230' 32 02A5'          ld      (erflag),a   ; evtl Fehler ablegen
0240' C9                ret
                        ;
                        ; ################
                        ; Sektor schreiben (CP/M3)
                        ; ################
                        ;
0241' CD 024D'  writehst:call   rw$init
0244' 3E 0E             ld      a,14         ; BIOS WRITE
0246' CD 00B4'          call    xbios
0249' 32 02A5'          ld      (erflag),a
024C' C9                ret
                        ;
                        ; ################
                        ; Anpassungen
                        ; ################
                        ;
024D' 3A 0294'  rw$init:ld      a,(hstsec)   ; phys. Sektornummer
0250' 6F                ld      l,a
0251' 26 00             ld      h,0
0253' 22 02C0'          ld      (bcreg),hl   ; <BC>=Sektornummer
0256' 2A 02A1'          ld      hl,(xlat)
0259' 22 02C2'          ld      (dereg),hl   ; <DE>=XLT-Adresse
025C' 3E 10             ld      a,16         ; BIOS SECTRN
025E' CD 00B4'          call    xbios
0261' 7D                ld      a,l          ; aktueller Sektor
0262' 32 0295'          ld      (actsec),a
0265' 22 02C0'          ld      (bcreg),hl   ; <BC>=Sektornummer
0268' 3E 0B             ld      a,11         ; BIOS SETSEC
026A' CD 00B4'          call    xbios
026D' 2A 0292'          ld      hl,(hsttrk)  ; phys.Tracknummer
0270' 22 02C0'          ld      (bcreg),hl   ; <BC>=Tracknummer
0273' 3E 0A             ld      a,10         ; BIOS SETTRK
0275' CD 00B4'          call    xbios
0278' 21 02C6'          ld      hl,hstbuf    ; Sektorpuffer
027B' 22 02C0'          ld      (bcreg),hl   ; <BC>=interne DMA-Adresse
027E' 3E 0C             ld      a,12         ; BIOS SETDMAR
0280' CD 00B4'          call    xbios
0283' 3E 01             ld      a,1
0285' 32 02BF'          ld      (areg),a     ; Banknummer
0288' 3E 1C             ld      a,28         ; BIOS SETBNK
028A' C3 00B4'          jp      xbios        ; DMA-Bank setzen
                        ;
                        ; ################
                        ; variable
                        ; ################
                        ;
0280'   0001    sekdsk: defs    1            ; logisches Laufwerk
028E'   0002    sektrk: defs    2            ; logischer Track
0290'   0001    seksec: defs    1            ; logischer Sektor
0291'   0001    hstdsk: defs    1            ; phys Laufwerk
0292'   0002    hsttrk: defs    2            ; phys Track
```

```
0294' 0001      hstsec: defs    1           ; phys Sektor
0295' 0001      actsec: defs    1           ; uebersetzter Sektor
0296' 0001      sekhst: defs    1           ; phys Sektor (Zwischenablage)
0297' 0001      hstact: defs    1           ; Pufferflag
0298' 0001      hstwrt: defs    1           ; Puffer-wechsel flag
0299' 0001      unacnt: defs    1           ; unalloc Sektorzaehler
029A' 0001      unadsk: defs    1           ; Laufwerk
029B' 0002      unatrk: defs    2           ; Track
029D' 0001      unasec: defs    1           ; Sektor
029E' 0002      sekbuf: defs    2           ; log Sektoradresse im Puffer
02A0' 0001      spt:    defs    1           ; Sektoren pro Track (aus DPB)
02A1' 0002      xlat:   defs    2           ; XLT-Adresse
02A3' 0001      bsm:    defs    1           ; block shift maske
02A4' 0001      psh:    defs    1           ; phys shift factor
02A5' 0001      erflag: defs    1           ; Fehlerflag
02A6' 0001      rsflag: defs    1           ; .. Sektor lesen
02A7' 0001      readop: defs    1           ; 1 wenn lesen - 0 wenn schreiben
02A8' 0001      rwflag: defs    1           ; phys Leseflag
02A9' 0001      wrtype: defs    1           ; schreiben
02AA' 0002      dmaadr: defs    2           ; DMA-Puffer Adresse
02AC' 0002      old$addr:defs   2           ; Adresse des 'alten' bios

                        ;
                        ; ****************
                        ; int. DPH
                        ; ****************
                        ;
                        ; Die Daten hierzu werden aus dem CP/M 3 DPH uebernommen
                        ;
02AE' 00 00     dph:    defs    2,0         ; kein XLT
02B0' 0008              defs    8           ; 2.2 scratch
02B8' 0002      dpb:    defs    2           ; dpb
02BA' 0002              defs    2           ; csv
02BC' 0002              defs    2           ; alv

                        ;
                        ; ****************
                        ; BIOSPB
                        ; ****************
                        ;

02BE' 0001      biospb: defs    1
02BF' 0001      areg:   defs    1
02C0' 0002      bcreg:  defs    2
02C2' 0002      dereg:  defs    2
02C4' 0002      hlreg:  defs    2

                        ;
                        ; ****************
                        ; Hostpuffer
                        ; ****************
                        ;
                        ; Dieser Bereich dient als Puffer zum Umsetzen von
                        ; max 1024 Byte Sektoren unter CP/M 3 in 128 Byte-
                        ; Sektoren unter CP/M 2
                        ;
```

```
 523 02C6'   0400      hstbuf: defs    1024      ; solle jecentalls reichen
 524
 525
 526                           ;
 527                           ; Der gesamte benoetigte Speicherbereich muss definiert
 528                           ; werden, damit der RSX nicht in das BDOS hinein'ragt'.
 529                           ;
 530
 531                           end
0 Error(s) Detected. 1734 Program Bytes.
78 Symbols Detected.
```

Glossar

Dieses GLOSSAR soll helfen, immer wieder verwendete Begriffe etwas zu verdeutlichen:

Adresse

Der Speicher eines Computers ist in eine Vielzahl von einzelnen Bytes (mit jeweils 8 Bits) unterteilt. Zur Orientierung werden alle Bytes durchnummeriert. Die jeweilige Nummer eines Bytes, beginnend mit 0 bezeichnet man als Adresse. Adressen werden üblicherweise hexadezimal (sedezimal) angegeben.

Algorithmus

Unter diesem Begriff ist eine Rechenvorschrift zu verstehen, mit deren Hilfe bestimmte Rechenvorgänge erledigt werden.

Assembler

Programm mit dessen Hilfe Programmieranweisungen, die in Mnemonics (siehe dort) geschrieben sind, in die entsprechende Maschinensprache umgewandelt werden.

So wird z.B. aus der Anweisung: JP F400

der Maschinencode: C3 00 F4

Attribut

Im Umgang mit Computern ist es recht häufig notwendig für bestimmte Vorgänge Kennzeichen zu setzen um damit Entscheidungskriterien beim Programmablauf zu bekommen. Ein typisches Beispiel ist das Setzen eines Paßwortes. Diese Kennzeichen bezeichnet man als Attribut, wenn es es zu einem bereits vorhandenen (und anderweitig benutzten) Zeichen hinzugefügt wird. Für Attribute bietet sich das (unbenutzte) 8. Bit von ASCII-Zeichen an, das vor der Bearbeitung wieder maskiert (im Sinne von wegnehmen) werden.

BDOS

(Engl: Basic Disk Operating System - Disketten (Grund)-Betriebs-System). Dieser Begriff gilt allgemein für ein entsprechendes Programm und besonders für einen Teil des Betriebs-Systemes CP/M. Es ist in diesem Falle der hardware-unabhängige Teil von CP/M.

BIOS

(Engl: Basic Input Output System - Grundsystem für Eingabe und Ausgabe). Dies ist unter CP/M der hardware-abhängige Teil des Betriebs-Systemes.

Bank

Normalerweise ein Platz, wo man sich setzen kann - oder wo's Geld gibt. In diesem besonderen Fall sind darunter 64k-Speicherbereiche zu verstehen, die nicht direkt von der CPU adressiert (angesprochen) werden können. Durch spezielle hardware können die 64K-Segmente über Port-Befehle einzeln gesetzt werden. Jedes 64K-Segment bezeichnet man als Bank.

Befehl

Ein Befehl, auch wenn er in verschiedenen Variationen auftreten kann, bedeutet im Enderfolg jedoch immer dasselbe: eine Anweisung an den Computer etwas ganz bestimmtes zu tun.

Die unterste Ebene eines Befehles ist der Maschinenbefehl. Dies ist ein Code-Wort, das von der CPU direkt 'verstanden' und ausgeführt wird.

Die höchste Ebene sind Programme, die geschrieben wurden, um einen ganz bestimmten Vorgang einzuleiten.

Betriebssystem

Hierunter ist eine Anzahl von Programmen und Programmsegmenten zu verstehen, die es ermöglichen einen Computer in soweit in Betrieb zu nehmen, so daß dieser eine sinnvolle Arbeit verrichten kann und Massenspeicher (wie Disketten oder Kassetten) als zusätzliches Speichermedium bedient werden können.

Bit

Dies ist die kleinste 'Recheneinheit' eines Computers. Ein Bit kann nur zwei Werte annehmen 0 oder 1 (falsch oder wahr). Auf diesen zwei möglichen Werten basiert die gesamte Boolsche Algebra.

Block

Ein Block ist unter CP/M die kleinste Einheit, die mit einem Eintrag im Inhaltsverzeichnis einer Diskette abgelegt werden kann. Bei Standard-CP/M (8" einfache Dichte) sowie bei allen Disketten mit einem Speicherplatz < 256K ist die Blockgröße 1024 Bytes (=1K-Byte). Bei Disketten mit größeren Speicher-kapazitäten beträgt die Blockgröße meist 2048 Bytes (=2K-Byte). Die höchstmögliche Blockgröße ist 16K-Byte.

Etwas inkonsequent werden von Digital Research in den Unterlagen zu CP/M mehrmals Sektoren als Blöcke bezeichnet.

Boolsche Algebra

Rechenvorschriften im Umgang mit Bits und Bytes. Es gelten folgende Regeln:

AND	OR	XOR	NOT
0 and 0 = 0	0 or 0 = 0	0 xor 0 = 1	not 0 = 1
0 and 1 = 0	0 or 1 = 1	0 xor 1 = 0	not 1 = 0
1 and 0 = 0	1 or 0 = 1	1 xor 0 = 0	
1 and 1 = 1	1 or 1 = 1	1 xor 1 = 1	

Boot

(von Engl: bootstrap=Schnürsenkel). Dieser Ausdruck wurde von der amerikanischen Redewendung ' sich am Schnürsenkel aus dem Dreck ziehen' abgeleitet. Gemeint sind im Zusammenhang mit Computern alle notwendigen Vorgänge, bis der Computer (vielmehr sein Betriebs-System) bereit zur 'Arbeitsaufnahme' ist.

Byte

Dies ist die Zusammenfassung von jeweils 8 Bits. Ein Byte ist die kleinste adressierbare Einheit in einem Computer-System. Ein Byte kann folgende Werte annehmen:

binär	hexadezimal	dezimal
00000000	00	0
00000001	01	1
00000010	02	2
00000011	03	3
00000100	04	4
usw		
11111100	FC	252
11111101	FE	253
11111110	FE	254
11111111	FF	255

Es ist zu beachten, daß ein Byte zwar 256 Werte annehmen kann, diese in Zahlen ausgedrückt jedoch nur im Bereich von 0..255 gehen.

In einem Byte sind die einzelnen Bits von Rechts nach Links nummeriert. Das äußerste rechte Bit ist Bit 0, das äußerste linke Bit ist Bit 7.

CCP

(Engl: Console-Command-Prozessor - Konsolen-Befehls-Ausführer). Dies ist ein Programmsegment von CP/M. Der CCP erlaubt einen gewissen 'Dialog' mit dem Betriebs-System und ermöglicht es Programme aus dem Speicherraum einer Diskette aufzurufen.

Code

Im allgemeinen die Verschlüsselung von Worten, Anweisungen oder Befehlen. Bei Computern spricht man bei den Maschinen-Befehlen meist ebenfalls von Code (im Sinne einer verschlüsselten Mnemonic-Anweisung).

Wenn bestimmte Bits in einem Byte als Attribut verwendet werden, kann es sich ebenfalls um einen Code handeln.

CP/M

Es wurden schon viele Übersetzungen dieses Namens 'angeboten'. Digital Research, der 'Besitzer' von CP/M gibt dazu keine Angaben. Die Übersetzung 'Control Program/Monitor' dürfte noch am sinnvollsten sein.

CP/M ist ein Diskettenorientiertes Betriebssystem, das ursprünglich für die 8-Bit CPU's 8080 und Z80 von Gary Kindall entwickelt wurde.

CPU

(Engl: Central-Prozessor-Unit - Zentrale-Rechnungs-Einheit). Man versteht darunter einen Baustein (oder im Falle von Großcomputern auch eine Gruppe von Bausteinen), die Maschinenbefehle entgegennehmen können und entsprechend deren Kodierung bestimmte Befehle ausführen.

CPU's der heutigen micro-Generation können im allgemeinen 8 bis 32 Bits mit einem Befehl 'bearbeiten' und Speicherräume von 64K bis einige Megabyte (1M=1000K) direkt adressieren.

Cross-Referenz

In diesem Zusammenhang bedeutet dies eine Tabelle mit Querverweisen aller 'Labels' in einem Assembler-Listing.

Datei

Eine Datei ist im allgemeinen eine Zusammenfassung von Daten beliebiger Art. Eine Datei wird mit einem Namen versehen, der ihren Inhalt bezeichnet und im Normalfall zusätzlich mit einem Index, der die Art der Daten angibt.

Eine Datei im Zusammenhang mit Computern kann z.B. eine Liste von Adressen und Telefonnummern sein, aber auch ein ein Assemblerprogramm.

Daten

Unter Daten sind Angaben oder Beschreibungen jeglicher Art zu verstehen, die sortiert, geordnet oder sonstwie bearbeitet werden können. Daten können als ASCII-Text vorliegen oder in hexadezimal bzw. binär verschlüsselt sein.

Datumsmarke

Um in einem Betriebs-System wie CP/M den Zugriff auf Dateien mit Uhrzeit und Datum zu vermerken, werden die Dateien im Inhaltsverzeichnis mit einer Datumsmarke versehen. Diese 'Marke' ist zweigeteilt und besteht aus der Uhrzeit (Stunde und Minute) sowie dem eigentlichen Datum. Das Datum wird, um damit auch Rechnen zu können, als fortlaufende Nummer beginnend von einem bestimmten Tage an (bei CP/M 3 ab dem 1.1.1978) abgelegt. Diese Nummer ist 2 Bytes(=16Bits) groß und hexadezimal verschlüsselt.

Debugger

(Engl: bug=Käfer, Wanze) Es sind darunter Programme zu verstehen, mit deren Hilfe es möglich ist Fehler in (neuen) Programmen zu suchen. Debugger erlauben es im allgemeinen, die zu untersuchenden Programme Schritt für Schritt, Byte für Byte durchzugehen (zu tracen=verfolgen) und dabei die jeweiligen Registerinhalte zu untersuchen, um auf diese Art die Ursache eines Fehlverhaltens zu finden.

Default

(Engl: Vorgabe, vorgegebenes). Dieser Begriff wird unter CP/M vor allem im Zusammenhang mit Laufwerken und Paßworten benutzt. Das default-Laufwerk ist immer das Laufwerk, auf dem gerade gearbeitet wird, also dessen 'Name' vom CCP im Prompter ausgegeben wird.

Wird ein Paßwort als 'default' gesetzt, so ist darunter zu verstehen, daß das Betriebs-System sich diese Vorgabe merkt und, solange die entsprechende Diskette nicht gewechselt wird, auf eine weitere Angabe des Paßwortes beim Diskettenzugriff verzichtet.

Dichte

Disketten können mit verschiedenen Verfahren beschrieben werden. Am Anfang der 'Diskettenzeit' wurden alle Disketten mit einem Verfahren beschrieben, daß heute 'einfache' Dichte heißt. Nachdem die Mechanik besser und die Elektronik zuverlässiger wurde, konnte mit einem neuen Verfahren die doppelte Datenmenge auf den gleichen Datenträger geschrieben werden. Dieses Verfahren nannte man folgerichtig 'doppelte'-Dichte.

Disassembler

Ein Disassembler ist, wie der Name schon sagt, das Gegenstück zu einem Assembler. Er wandelt den von einem Assembler (oder sonstwie) erzeugten Maschinencode wieder in die leichter 'lesbaren' Mnemonics zurück. Ein Disassembler ist eines der wichtigsten debugger-Hilfsmittel.

Mit einem Disassembler können Speicherinhalte zwar als Maschinenbefehle übersetzt werden, ob es sich dabei aber u.U. um Datenfelder oder um Texte handelt, kann der Disassembler nicht erkennen.

DMA

(Engl: Direct-Memory-Access - Direkter-Speicher-Zugriff). Es sind darunter zwei Begriffe zu verstehen:

1. Eine Pufferadresse, wie sie von CP/M verwendet wird und

2. Ein Hardware-Baustein, der einen schnellen Speicher-zu-Speicher Transfer ermöglicht.

Unter CP/M 3 ist immer der erste Begriff gemeint.

Editor

Ein Editor ist ein Programm zur Textbearbeitung. Mit seiner Hilfe können Texte geschrieben und korrigiert (editiert) werden. Ein Editor ist das wichtigste Werkzeug eines Computers. Editoren können in manchen Programmen auch bereits eingebaut sein.

Eintrag

Jede Datei, die auf einer Diskette gespeichert ist, wird in einen sog. Inhaltsverzeichnis (Engl: Directory) eingetragen. Dieser Eintrag umfaßt nicht nur den Namen der Datei, sondern auch Angaben darüber, wo auf der Diskette die Datei abgelegt wurde und u.U. zu welcher Zeit und an welchem Tag.

Extend

(Engl: Erweiterung) Darunter ist unter CP/M zweierlei zu verstehen.

1. die engl. Bezeichnung für den Index einer Datei
2. die Angabe ob eine Datei aus mehreren Extend im Sinne von Teilen (mit jeweils eigenem FCB) besteht.

FCB

(Engl: File-Control-Block - Dateien Kontroll Block). Es ist unter einem FCB eine Datenstruktur zu verstehen, in welcher alle (für CP/M) notwendigen Angaben zu einer Datei abgelegt sind.

File

(Engl: Datei) siehe dort

Flag

(Engl: Flagge) Es ist darunter eine Marke, ein gesetztes Bit o.Ä. zu verstehen. Ein Flag dient als Kriterium, ob ein bestimmter, gewünschter Vorgang ausgeführt wird oder nicht.

Format

Auf eine Diskette können die Daten in unterschiedlicher Art und Weise aufgezeichnet werden. Laufwerksabhängig sind 40, 77 oder 80 Tracks möglich. Es kann mit einfacher oder doppelter Dichte geschrieben werden. Es können Sektorgrößen von 128, 256, 512 oder 1024 Bytes benutzt und darüberhinaus die Anzahl der Sektoren pro Track variiert werden. Auch die Art der Seitenumschaltung und die Reservierung der System-tracks spielt eine Rolle. Alle Angaben zu den genannten Faktoren bezeichnet man als Format einer Diskette.

Das Format gibt natürlich auch Auskunft über den ursprünglichen Sinn dieses Wortes, ob es sich um eine MIKRO, MINI oder MAXI-Diskette handelt. (3,5" 5,25" oder 8").

Generieren

Unter diesem Begriff ist der Vorgang einer Programmaufbereitung von der Assemblerdatei bis zum ablauffähigen Programm zu verstehen.

Harddisk

Eine Harddisk entspricht im Prinzip einer Floppy-Disk. In diesem Falle ist jedoch das Speichermedium (die Disk) nicht austauschbar (sondern hart - Engl: hard=hart,fest). Hardisks haben im allgemeinen mehrere harte Scheiben (Platten) und auch mehrere Köpfe, mit denen gelesen oder geschrieben wird. Typische Speichervolumen einer Harddisk für Mikrocomputer sind 5..20 MByte.

Harddisks für Mikrocomputer-Anwendungen werden ebenfalls in den Formaten 3.5", 5.25" und 8" geliefert.

Hardware

Unter hardware versteht man bei einem Computer alles, was man 'anfassen' kann, also jeder Baustein, der Bildschirm, die Tastatur - eigentlich der ganze Computer, so wie er da steht.

hexadezimal (sedezimal)

Computer werden mit Befehlen gesteuert, die aus einzelnen Bits bestehen. Da ein Bit nur zweiwertig ist (ja-nein, aus-ein), jedoch meist in Gruppen von 8 oder 16 Bits vom Computer benötigt wird, ist seine Darstellung zwar recht einfach, das Arbeiten mit diesen zweiwertigen Zahlen jedoch recht umständlich.

Aus diesem Grunde werden die meisten Zahlen und Adressen in Computern in Vierergruppen zusammengefaßt und mit der Basis 16 dargestellt. Die ersten 9 Zahlen entsprechen dabei dem gewohnten Bild der dezimalem Zahlenreihe. Die 'Zahlen' 10..15 (beginnend bei 0!) werden im Hexadezimalsystem mit den Buchstaben A..F bezeichent. Die nachfolgende Tabelle soll dies verdeutlichen:

binär	hexadezimal	dezimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

Der Grund, warum gerade die Zahl 16 gewählt wurde, liegt ebenfalls in der Zweiwertigkeit. Es sind nur Zahlenreihen, die eine Potenz der Zahl zwei darstellen möglich, also 2, 4, 8, 16, 32 usw. In älteren Programmen und Zusammenstellungen, wurde der Wert 8 als Zahlenbasis (Oktal) verwendet, da in einem 64K System Oktalzahlen aber nicht voll 'ausgenutzt' werden konnten, wurde wieder davon abgegangen. (11111111=oktal 377=hexadezimal ff).

Hexadezimale Zahlen werden üblicherweise mit einem angehängtem 'h' oder 'H' gekennzeichnet. Einige Programme 'verlangen' ein führendes '$'-Zeichen. Hexadezimale Zahlen beginnen üblicherweise mit einer Zahl (0..9), einem Buchstaben (A..F) ist eine NULL (0) voranzustellen.

Index

Zusätzliche Bezeichnung oder 'Anhängsel' eines Begriffes.

Initialisierung

Wird ein System neu gebootet, so müssen verschiedene Vorgaben erfüllt werden, aber auch bestimmte Hardware-Bausteine durch eine entsprechende software in einen bestimmten Zustand versetzt werden. Diesen Vorgang nennt man Initialisierung.

Interface

(Engl: face=Gesicht). Man versteht darunter einen physikalischen oder logischen Platz, an dem Daten irgendwelcher Art (von Angesicht zu Angesicht) übergeben werden können. Zwei Beispiel sollen dies verdeutlichen:

1. Einige Bausteine und (nur dies ist für den Benutzer erkennbar) ein Stecker dienen dazu (über ein Interface-Kabel) den Computer mit einem Drucker zu verbinden.

   In diesem Falle spricht man von einem Hardware- (oder physikalischen) Interface.

2. Die Adresse 0005H in der TPA-Bank eines CP/M-Computers wird aufgerufen, wenn bestimmte Befehle vom Betriebs-System ausgeführt werden sollen.

   In diesem Falle spricht man von einem Software- (oder logischem) Interface.

Der Punkt an dem zwei Interface-Geräte verbunden werden, nennt man 'Schnittstelle'.

Werden über eine Schnittstelle Signale ausgetauscht, die der Steuerung der Zeichenübergabe dienen, spricht man von einer Handshake-Verbindung. (Engl: Händeschütteln)

Interrupt

(Engl: Unterbrechung) Bestimmte Bausteine können den Computer veranlassen, seine momentane 'Arbeit' zu unterbrechen und eine bestimmte Adresse nach weiteren Informationen abzufragen. Diesen Vorgang nennt man Interrupt.

Kompatibel

Kompatibilität ist nicht unbedingt Gleichheit im Sinne des Aussehens, sondern Gleichheit im Sinne der Verhaltensweise. So können z.B. zwei Disketten völlig gleich aussehen jedoch mit unterschiedlichen Formaten beschrieben sein, sie sind damit nicht kompatibel; zwei Computer können völlig unterschiedlich aussehen, aber beide können unter CP/M Programme auf Disketten austauschen und ablaufen lassen, sie sind untereinander kompatibel.

Konfiguration

Unter Konfiguration ist die Zusammenstellung Hardware und Software zu verstehen, unter welcher ein Computer in Betrieb genommen wird. Hauptsächlich bezieht sich diese Zusammenstellung auf die benutzte hardware; ein oder mehrere Laufwerke, das Diskettenformat, welche Art der Druckersteuerung wird verwendet und und und ...

Label

(Engl: Etikett) Dieser Begriff hat sich als Bezeichnung des Namens eines Unterprogrammes, einer Konstanten oder einer Variablen in Programmen eingebürgert. Einem Label ist immer ein bestimmter Wert oder eine bestimmte Adresse zugeordnet.

Unter CP/M 3 kann auch einer Diskette ein Label (im Sinne von Namen) zugewiesen werden.

Listing

Listing ist der (vom Assembler veranlaßte) Ausdruck eines in Mnemonic's geschriebenen Programmes, das neben dem vom Programmierer geschriebenen Teil auch die vom Assembler decodierten Maschinenbefehle und Adressen angibt.

Der Ausdruck von Programmen in Hochsprachen wie BASIC, PASCAL usw. wird heute ebenfalls als Listing bezeichnet, obwohl hier keineswegs eine 'Liste der Adressen und Maschinenbefehle' ausgegeben wird.

Logisch

Unter CP/M wird als logisch bezeichnet, was als Einheit von der Software benutzt wird. Die Sektoren einer Diskette z.B. können die Größe 128, 256, 512 oder 1024 Bytes haben, unter CP/M wird immer mit einer 'logischen' Sektorgröße von 128 Bytes gerechnet.

Das gleiche gilt für Ein- und Ausgabeeinheiten. Ihre 'logische' Bezeichnung ist z.B. CONIN, ob es sich dabei um die (physikalische) Einheit einer Tastatur, eines Terminals, eines zweiten Computers oder was auch immer, handelt ist dabei völlig unwesentlich, den das ist Hardware und interessiert das Betriebssystem nicht.

Maschinensprache

Maschinensprache ist die unterste Ebene der Verständigung Mensch-Computer. Jeder Computer wird von einigen hundert Code-'Wörter' gesteuert, die jeweils eine ganz bestimmte Aktivität des Computers veranlassen.

Menü

Darstellung der Befehle oder sonstigen Möglichkeiten eines Programmes in Form einer Tabelle. Jeder Möglichkeit wird eine Zahl, ein Buchstabe oder eine CNTRL-Zeichen zugeordnet. Durch Eingabe des entsprechenden Zeichens wird der Befehl ausgeführt oder ein entsprechendes Unterprogramm aufgerufen.

Mnemonic

(Engl: 'Gedächtniskürzel) Anstelle der 'nichtssagenden' Bit-Zusammenstellung von Maschinenbefehlen wurden Abkürzungen der (englischsprachigen) Befehlsbedeutung eingefürt. Diese neuen 'Befehlsworte' werden vom Assembler in Maschinensprache übersetzt.

Modul

Teil eines Ganzen. Hier meist verwendet als Teil eines Gesamtprogrammes. Ein Modul ist jedoch keinesfalls nur ein Unterprogramm, es ist darunter vielmehr die Zusammenfassung mehrere Unterprogramme zu verstehen, die (meist) alle dem gleichen Aufgabengebiet zugewiesen sind. Beispiel sind die Module BIOSKRNL, CHARIO, MOVE, DISKIO und BOOT, die zusammen das BIOS eines CP/M 3 System bilden.

Modus

Ein Modus ist die Art und Weise wie etwas geschieht.

Monitor

Dieser Begriff ist im Zusammenhang mit Computern zweigeteilt zu verstehen:

1. Ein Monitor ist ein Fernsehgerät ohne Fernseh- und Ton-Empfangsteil. Er dient bei einem Computer zur Zeichen-Ausgabe.

2. Ein Programm, das die Grundinitialisierung eines Computers übernimmt und darüberhinaus die Möglichkeit bietet gewisse Grundfunktionen der Zeichen -Eingabe und -Ausgabe zu übernehmen. 'Vornehme' Monitorprogramme können in gewisser Weise bereits ein (stark abgemagertes) Betriebssystem darstellen, mit eingebautem debugger, Assembler usw.

Offset

Im Zusammenhang mit Computern und Programmen versteht man unter einem Offset die 'Entfernung' eines Wertes in einer Tabelle vom Tabellenanfang. Nimmt man z.B. das Alphabet als Tabelle, so hat der Buchstaben 'A' einen Offset von 0, der Buchstaben 'B' einen Offset von 1 usw.

Unter CP/M wird recht häufig mit Tabellen gearbeitet, wobei ein bestimmter Buchstaben oder eine Zahl den Offset zu einer Adresse bilden. Dies einer der schnellsten Wege des Datenzugriffes bei einer Vielzahl von Daten.

Option

Unter Option die frei Entscheidung zu verstehen, ob eine Möglichkeit etwas bestimmtes zu tun, ergriffen wird oder nicht.

Parity

Dies ist ein Begriff aus der Datenübertragung. Werden Daten bitweise übertragen, so ist es die einfachste Möglichkeit der Fehlerüberprüfung alle (binären) Einsen eines Bytes zu zählen. Je nach Art der gewünschten Prüfung wird nach der eigentlichen Übertragung ein Kontrollbit gesandt, das auf '1' gesetzt ist, wenn die Summe gerade war (even) oder auch wenn sie ungerade war (odd). Im jeweils umgekehrten Fall ist das Kontrollbit '0'.

Buchstaben und Zeichen, die über die Konsole gesandt werden, sind üblicherweise nur 7 Bit groß (das gibt immerhin 128 Möglichkeiten!). Viele Tastaturen und Terminals fügen als 8. Bit ein entsprechendes parity-Bit ein. Dieses muß zum reibungslosen Betrieb unter CP/M (im BIOS) herausmaskiert werden.

Paßwort

Mit einem Paßwort (Kennwort) können Dateien unter CP/M 3 vor jedem Fremdzugriff geschützt werden. Ein Paßwort besteht im allgemeinen aus einer beliebigen 8-stelligen Ziffern- und Zeichenkombination.

physikalisch

Dieser Begriff wird im Zusammenhang mit CP/M 3 im Sinne von greifbar, tatsächlich vorhanden benutzt. Es steht im Gegensatz zu logisch (siehe auch dort).

Port

(Engl: Tor. Türe). Es ist darunter eine Adresse zu verstehen, unter welcher auf bestimmte Interface-Bausteine zugegriffen werden kann.

Programm

Ein Programm ist eine sinnvolle Zusammenstellung von Maschinenbefehlen, die eine bestimmte, vorgegebene 'Arbeit' ausführen.

Prozessor

Dies ist eine andere Bezeichnung für die CPU.

RAM-Disk

Im Gegensatz zur Floppy-Disk und Harddisk, benutzt die RAM-Disk normale Speicher (RAM=Random Access Memory - Speicher mit wahlfreiem Zugriff). Der Vorteil ist ein schneller Zugriff auf die gewünschten Daten; der Nachteil ist es, daß die Daten bei Stromausfall (oder Abschalten des Gerätes) verloren gehen. Im Zeichen des Preisverfalles von CMOS-Speicher, die mit einer Batterie gepuffert werden können, bekommen RAM-Disks immer mehr Bedeutung als schnelle Zwischen- und Arbeitsspeicher.

record

Dieser (englische) Begriff wird verwendet als Bezeichnung von logischen oder physikalischen Sektoren oder Blöcken. Im allgemeinen bezeichnet er ein Dateiensegment.

RETURN

(Engl: Zurück) Bezeichnet ein bestimmtes CNTRL-Zeichen auf der Tastatur (Wagenrücklauf=engl: Cariage Return <CR>). Dieses Zeichen dient meist dem Befehlsabschluß.

Segment

Unter Segment ist der Teil eines Ganzen zu verstehen.

Seitenumschaltung

Einige Laufwerke (Floppy-Disk) können die Disketten beidseitig beschreiben. Welche Seite benutzt werden soll bleibt einem (BIOS-Internen) Algorithmus überlassen.

Eingebürgert haben sich vor allem 3 Methoden:

1. Alle 'geraden' Tracks sind auf Seite 0 (der Vorderseite) und 'ungeraden' Tracks auf der Rückseite. Eine sehr verbreitete Methode, da sehr einfach zu realisieren.

2. Ab einem bestimmten Track wird auf den Anfang der Rückseite umgeschaltet, (z.B. Track 40 oder Track 80). Die Methode des IBM-PC's.

3. Die Seitenumschaltung erfolgt in Abhängigkeit der Sektornummer.

Sektor

Jede Diskette ist in Tracks und Sektoren unterteilt. Die Tracks geben die Position des Schreib- Lesekopfes von außen nach innen einer Diskette an.

Jeder Track ist in verschiedene Segmente unterteilt. Jedes dieser Segmente ist (intern) nummeriert und wird als Sektor bezeichnet.

Software

Im Gegensatz zur Hardware ist die Software nicht zum Anfassen. Es sind darunter alle Programme zu verstehen, die dem Computer eine sinnvolle Tätigkeit zuweisen. (Unter sinnvoll soll hier lediglich ein Programm verstanden werden, daß der Computer 'versteht').

Speicher

Unter Speicher ist jedes Medium zu verstehen, in das computergerecht (also von diesem ansprechbar) Befehle und Daten abgelegt werden können. Speicher können Disketten und Kassetten sein, ebenso wie RAM's EPROM's oder PROM's.

string

(Engl: Kette) Als 'string' bezeichnet man im Computerbereich zusammenhängende Zeichenketten. Wird von einem Computer z.B. der Satz 'guten Tag' ausgegeben, so bilden alle Buchstaben darin, incl. dem Leerzeichen, einen Textstring.

Syntax

Unter Syntax ist eine bestimmte, vorgegebene Sprachregelung oder Anordnung von (Sprach- oder Zeichen-) Elementen zu verstehen.

Schreibt die Syntax vor, daß ein Befehl in der Form

A: tuwas

eingegeben wird, so wird der Computer nur diese Eingabeform akzeptieren und z.B. tuwas A oder Atuwas nicht erkennen.

System

Ein System ist die Zusammenstellung verschiedender Software-Module die eine bestimmte Aufgabe zusammen mit einer bestimmten Hardware erfüllen soll.

TPA

(Engl: Transient Program Area - (Speicher)Raum für 'Durch-gangs-Programme). Hierunter ist der Arbeitsspeicher eines CP/M Systems zu verstehen. In diesem Speicherteil 'laufen' alle Programme ab.

Track

Siehe hierzu Sektor

USER

(Engl: Benutzer) Der Speicherbereich einer Diskette kann in gewisser Weise in verschiedene (bis zu 16) Ebenen unterteilt werden. In jeder Ebene können beliebige Dateien abgelegt werden, auf die aus einer anderen Ebene nicht zugegriffen werden kann.

Die Benutzung von USER-Ebenen dient heute vor allem der Disketten-Organisation und der Übersichtlichkeit in den einzelnen Inhaltsverzeichnissen.

Vektor

Unter Vektor ist im Sprachgebrauch der Computer eine Zeiger im allgemeinen Sinne zu verstehen. Dieser Zeiger kann auf ein Byte, ein Wort oder eine Adresse hinweisen. Unter CP/M kann ein Vektor auch ein Wort enthalten, dessen einzelne Bits bestimmte Systemgegebenheiten signalisieren.

Wort

Unter Wort sind (je nach CPU) 2 oder 4 Bytes zu verstehen, genauer gesagt 16 oder 32 Bits. Unter CP/M 3 sind es 2 Bytes.

Zeitmarke

Unter Zeitmarke ist die Ablage der Uhrzeit im Inhaltsverzeichnis einer Diskette zu verstehen. Mit Zeitmarken wird je nach Anwendung die Entstehung einer Datei oder der Zugriff darauf sowie jede Veränderung markiert.

Zeitmarken sind im Übrigen zusammen mit der Datumsmarke ein wichtiges Ordnungselement bei der Organistation von Dateien.

Zugriff

Wird eine Datei gelesen oder geschrieben, verändert oder gelöscht, spricht man von einem Zugriff (des Betriebs-Systemes) auf diese Datei. Zugriffe jeglicher Art können unter CP/M 3 mit Datums- und Zeitmarken versehen werden.

Inhaltsverzeichnis der CP/M-Befehls- und Programmbeschreibung

Inhaltsübersicht Band II (NDR-Klein-Computer)

Dieser Band beschäftigt sich mit der Hardware-Anpassung von CP/M3 an den NDR-Klein-Computer.

Es werden ALLE notwendigen Programmsegmente beschrieben incl. dem LOADER-BIOS und der notwendigen Monitor-Anpassung mit ELMON.

Alle Programme sind mit vollständigen, deutsch kommentierten Listings abgedruckt.

Zusätzlich wird, ebenfalls mit Listing, die Implementation einer RAM-Disk, einer software-Uhr und eines sehr flexiblen Formatierers beschrieben.

Band II enthält folgende Kapitel: