Commodore
Sachbuch

Prof. Dr. Wolf-Jürgen Becker

C128
Alles über CP/M 3.0

Beschreibung der CP/M-Befehle ★ Struktur von CP/M
★ CP/M-Dateien ★ Programmieren unter CP/M
(Turbo-Pascal, Microsoft-Basic)

C128 – Alles über CP/M 3.0

C 128
Alles über CP/M 3.0

- Beschreibung der CP/M-Befehle
- Struktur von CP/M
- CP/M-Dateien
- Programmieren unter CP/M (Turbo-Pascal, Microsoft-Basic)

Prof. Dr. Wolf-Jürgen Becker

Markt & Technik Verlag

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Becker, Wolf-Jürgen:
C 128 – alles über CP, M 3.0 : Beschreibung d. CP-M-Befehle ; Struktur von CP/M ; CP-M-Dateien ;
Programmieren unter CP/M (Turbo-Pascal, Microsoft-Basic) / Wolf-Jürgen Becker. –
Haar bei München : Markt-und-Technik-Verlag, 1986.
(Commodore-Sachbuch)
ISBN 3-89090-370-3

15 14 13 12 11 10 9 8 7 6 5 4 3 2
89 88 87 86

ISBN 3-89090-370-3

Einbandgestaltung: Grafikdesign Heinz Rauner
Druck: Jantsch, Günzburg
Printed in Germany

Inhaltsverzeichnis

Vorwort

Mit dem neuen Homecomputer Commodore 128 liefert die Commodore GmbH das Betriebssystem CP/M 3.0 auf Diskette zum Einsatz dieses Rechners auch als Personal Computer mit.

Dieses Betriebssystem CP/M 3.0 bzw. CP/M Plus ist die modernste Version. Für 8-Bit-Rechner stellt CP/M praktisch einen Standard bei kommerziellen Anwendungen dar, so daß eine Vielzahl von Anwenderprogrammen für die verschiedensten Aufgaben erhältlich ist. Es bereitet meist keine große Schwierigkeit, ein für CP/M geschriebenes Programm auf einen Rechner anzupassen, wenn man - wie beim C 128 - verschiedene Diskettenformate lesen kann.

Dieses Buch soll dem Leser eine Einführung in das Betriebssystem CP/M geben und ihn in die Lage versetzen, das Betriebssystem auf dem Commodore 128 zu benutzen. Es wird ausführlich auf die Befehle des CP/M3-Betriebssystems eingegangen und deren Anwendung anhand von Beispielen näher erläutert.

Besonders interessant erscheint dabei die Installation höherer Programmiersprachen, wie Pascal, Fortran und BASIC als Interpreter und Compiler. Sich diese Sprachen verfügbar zu machen, war der wesentliche Beweggrund für das Schreiben dieses Buches.

Erfolgreich konnten ein MBASIC-Interpreter sowie zwei Fortran-Compiler und ein Pascal-Compiler installiert werden. Über die Vorgehensweise bei der Installation berichtet dieses Buch. Die auf dem C 64 mit dem CP/M-Z80-Modul unter dem Betriebssystem CP/M 2.2 arbeitenden Programme können meist direkt auch auf dem C 128 verwendet werden.

Besonderer Dank gilt Herrn Dipl.-Phys. Wolfgang Burda für die zahlreichen Anregungen, besonders beim Kapitel über den *Symbolischen Debugger* SID, und Frau Claudia Brandt für die sorgfältige Erstellung des Manuskriptes.

1 Einführung

1.1 Aufbau von Datenverarbeitungsanlagen (Rechnern)

Mit Datenverarbeitungsanlagen (Rechnern bzw. Computern) soll die Arbeit des Menschen in fast allen Bereichen des täglichen Lebens erleichtert werden. Dazu muß eine solche Anlage wesentliche Aufgaben übernehmen können, die früher von Menschen ausgeführt wurden. Zur Bewältigung einer Aufgabe wird eine Arbeitsanweisung benötigt, in der die einzelnen Arbeitsschritte nacheinander aufgelistet sind. Es entsteht eine Folge von Befehlen. Eine solche Arbeitsanweisung, die aus einer Folge von Befehlen (Anweisungen) besteht, nennt man ein Programm.

Ein Programm ist also eine in einer beliebigen Sprache abgefaßte, vollständige Anweisung zur Lösung einer Aufgabe mit einer Datenverarbeitungsanlage.

Zur Durchführung der Aufgabe sind jedoch noch Daten notwendig, die mit einem Programm verarbeitet werden sollen, d.h., unter Daten versteht man im allgemeinen die Zahlenwerte, mit denen die jeweilige Aufgabe zu lösen ist.

Programme und Daten stellen also Informationen für eine Datenverarbeitungsanlage dar, die von ihr verarbeitet werden (Bild 1.1).

- In einem ersten Schritt müssen diese Informationen in die Datenverarbeitungsanlage eingegeben werden. Diese Eingabe erfolgt über eine Eingabe-Einheit, wie zum Beispiel eine Schreibmaschinen-Tastatur oder einen Klarschriftleser.

- Programme und Daten müssen der Datenverarbeitungsanlage während der Bearbeitung zur Verfügung stehen. Dazu werden sie in der Datenverarbeitungsanlage in einem **Speicher** abgelegt. Solche Arbeitsspeicher sind heutzutage im allgemeinen elektronische Halbleiterspeicher, sogenannte Schreib/Lese-Speicher bzw. RAM (random access memory).

- Eine Datenverarbeitungsanlage benötigt eine Einrichtung, die die Berechnungen durchführt, das sogenannte **Rechenwerk**.

- Beim Durchführen eines Programmes wird ein Befehl nach dem anderen abgearbeitet. Für diese Aufgabe ist in einer Datenverarbeitungsanlage ein **Steuerwerk** (Leitwerk) vorgesehen.

- Die Ergebnisse der Verarbeitung durch die Datenverarbeitungsanlage werden im letzten Schritt ausgegeben. Diese Ausgabe erfolgt über **Ausgabe-Einheiten** wie z.B. Bildschirm, Drucker oder Plotter.

- Da der Arbeitsspeicher (RAM) meist in der Kapazität aus Kostengründen begrenzt ist, ist es nicht sinnvoll, Programme und Daten im Arbeitsspeicher dauernd zu speichern, sondern ihn nur während der Verarbeitung zu benutzen. Für große zu speichernde Informationsmengen verwendet man billigere, aber im allgemeinen auch langsamere externe Speicher. Solche externen Speicher sind z.B. Magnetbänder (Kassetten), Magnetplatten (Festplatte) und Magnetdisketten (Floppy Disk).

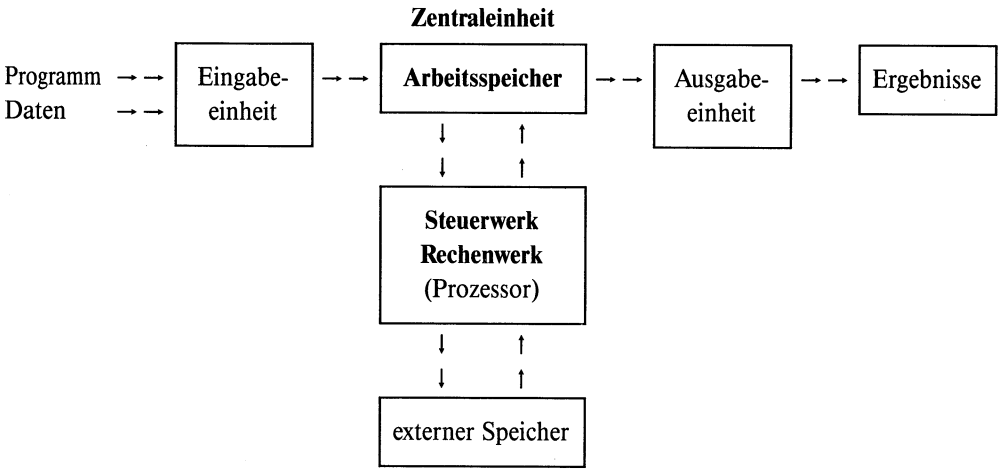


Bild 1.1: *Datenverarbeitungsanlage*

Das Steuer- und Rechenwerk faßt man im allgemeinen zu dem sogenannten Zentralprozessor CPU (central processing unit) zusammen. Als Zentraleinheit bezeichnet man den Arbeitsspeicher mit dem Zentralprozessor.

1.2 Mikrocomputer

Ein Mikrocomputer besteht aus denselben Teilen und arbeitet wie eine oben beschriebene Datenverarbeitungsanlage eines Rechenzentrums. Ein Mikrocomputer ist jedoch wesentlich kleiner und billiger. Die wesentlichen Einheiten sind

- die Zentraleinheit mit dem Prozessor (CPU) und dem Arbeitsspeicher (RAM),
- eine Tastatur zur Eingabe der Programme und Daten,
- ein Bildschirm-Sichtgerät zur direkten Anzeige (Ausgabe) der Programme, der eingegebenen Daten und der Ergebnisse,
- ein oder mehrere Diskettenlaufwerke.

Beim Commodore 128 sind die Zentraleinheit und die Tastatur in einem Gehäuse zusammengefaßt.

1.3 Programmierung

Eine Datenverarbeitungsanlage stellt zwar die technischen Funktionseinheiten bereit, aber erst mit einem Programm ergibt sich ein System, das die gestellte Aufgabe selbsttätig löst und die eingegebenen Daten wunschgemäß verarbeitet. Nur der Mensch kann eine für den Rechner verständliche Beschreibung der Arbeitsanweisungen (Anwenderprogrammierung) erstellen.

Bei einer programmgesteuerten Datenverarbeitungsanlage ist also eine Trennung zwischen der Arbeitsanweisung (Anwenderprogramm, Anwender-Software) und der ausführenden technischen Anlage (Rechner, Computer, Hardware) vorgenommen. Dadurch kann mit ein und demselben Rechner durch Ändern des Anwenderprogrammes nicht nur eine, sondern eine Vielzahl von verschiedenen Aufgaben durchgeführt werden.

Die Arbeitsanweisung bzw. Programme müssen in einer dem Rechner verständlichen Programmiersprache formuliert werden. Hierzu sind prinzipiell folgende Arten von Programmiersprachen verwendbar:

- Maschinensprachen
- Assemblersprachen
- problemorientierte Sprachen.

1.3.1 Maschinensprachen

Bei einer Maschinensprache (Maschinencode) handelt es sich um eine Verschlüsselung (Codierung) der Befehle in Binärziffern, die vom Rechner ohne weitere Umwandlung (Übersetzung) direkt als Steuersignale verarbeitet werden können, wobei der binäre Code der Maschinensprache für die einzelnen Befehle bei verschiedenen Prozessoren sehr unterschiedlich sein kann.

1.3.2 Assemblersprachen

Zur Vereinfachung der Programmierung und Verringerung des Zeitaufwandes wurden Assemblersprachen entwickelt. Hierbei besteht ein Befehl nicht mehr aus einem Binärcode, sondern aus einem einfach zu merkenden symbolischen (mnemonischen) Code. Ebenso kann eine Speicherplatzadresse mit einem symbolischen Namen gekennzeichnet werden. Da der Prozessor jedoch nur seinen eigenen Maschinencode versteht, muß der Assemblercode in die Maschinensprache übersetzt werden. Eine solche Übersetzung eines

in Assemblersprache geschriebenen Programmes in Maschinensprache wird als Assemblieren und das dazu notwendige Programm als Assembler bezeichnet. (Anmerkung: Die Umkehrung wird Disassemblieren genannt.)

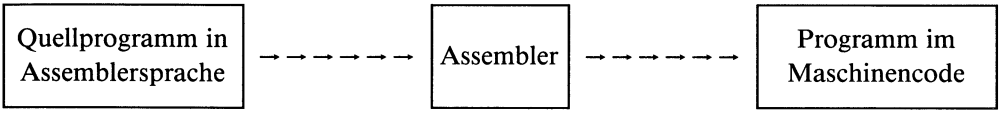


Bild 1.2: *Assemblieren eines in Assemblersprache geschriebenen Programmes im Maschinencode.*

1.3.3 Problemorientierte Programmiersprachen

Die Nachteile der maschinenorientierten Assemblersprache vermeiden die anlageunabhängigen, problemorientierten Programmiersprachen, die bei geringerem Zeitaufwand eine Programmierung in der jeweiligen Fachsprache erlauben und relativ leicht erlernbar sind.

Eine als mathematische Formel dargestellte Anweisung kann ein Rechner nicht direkt verstehen. Daher muß eine in einer problemorientierten Programmiersprache abgefaßte Anweisung in die Maschinensprache übersetzt werden. Diese Übersetzung erfolgt mit Übersetzungsprogrammen, entweder für jede Anweisung direkt im Dialog mit einem **Interpreter** oder für das gesamte vorher geschriebene Programm auf einmal mit einem **Compiler**.

Tabelle 1.1: Problemorientierte Programmiersprachen

Name	Übersetzung	Anwendungsbereich
ALGOL	Compiler	mathematisch-naturwissenschaftlich
FORTRAN	Compiler	mathematisch-naturwissenschaftlich
COBOL	Compiler	kommerziell
PL 1	Compiler	kommerziell/mathem.-naturwissenschaftlich
BASIC	Interpreter und Compiler	allgemein
PASCAL	Compiler	allgemein (strukturierte Programmierung)
FORTH	Interpreter-Compiler	Steuerung
C	Compiler	kommerziell und Betriebssysteme

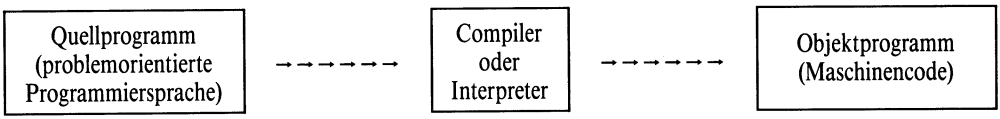


Bild 1.3: *Erstellen eines lauffähigen Programms*

1.4 Betriebssysteme

Der Benutzer oder Anwender eines Rechners (Hardware) setzt also gewisse Anwenderprogramme (Software) wie selbstgeschriebene Quellprogramme und Übersetzerprogramme ein.

Zusätzlich ist jedoch noch ein Hilfsprogramm, das sogenannte Betriebssystem, notwendig, das den Betrieb zwischen der Hardware (Rechner) und dem Anwenderprogramm organisiert, wie

- die Steuerung der Ein- und Ausgabe
  (Tastatur, Bildschirm, Diskettenlaufwerke, Drucker usw.),

- die Ablaufsteuerung des Programmes
  (Starten, Durchführen und Beenden des Programmablaufs),

- die Verwaltung des Arbeitsspeichers
  (Speichern von Programm und Daten im Arbeitsspeicher).

Ein Betriebssystem nimmt also dem Anwender Routineaufgaben ab und bestimmt damit in hohem Maße die Leistungsfähigkeit eines Rechners.

Mit einer relativ einfachen Kommandosprache kann der Benutzer das Betriebssystem zu den verschiedensten Tätigkeiten gezielt veranlassen.

Einfache Betriebssysteme für die wichtigsten Grundfunktionen bezeichnet man häufig auch als **Monitor.**

Betriebssysteme sind meist im Maschinencode oder in der zugehörigen Assemblersprache geschrieben. In kleineren Rechnern werden sie meist in Festwertspeichern (ROM) abgelegt (Firmware), z.B. das Betriebssystem des C 128 für den C-128- und C-64-Modus. Diese dauerhafte Speicherung ist einfach und billig. Erst bei aufwendigeren Mikrocomputern lohnt sich eine Speicherung auf einer Diskette, da ein Diskettenlaufwerk für den eigenen Betrieb eine aufwendige Steuerung benötigt, so daß dann insgesamt ein umfangreicheres Betriebssystem benötigt wird.

1.5 Grundlagen des CP/M-Betriebssystems

CP/M ist die Abkürzung für *Control Program for Microcomputers*, d.h. ein Betriebssystem für Mikrocomputer. Dieses von der Firma Digital Research entwickelte System stellt praktisch einen Standard für 8-Bit-Mikroprozessor-Systeme dar. Das CP/M-Betriebssystem wurde ursprünglich für den INTEL-Mikroprozessor 8080 erstellt und ist deshalb in der INTEL-8080-Assemblersprache geschrieben.

Es ist eines der ersten Betriebssysteme für Mikrocomputer gewesen, das auf mehreren Geräten verschiedener Hersteller implementiert wurde.

1.5.1 Hardware-Ausrüstung

Das Betriebssystem CP/M 3.0 erfordert eine Mindestausstattung von

- einem Mikrocomputer mit einem Arbeitsspeicher von mindestens 48 Kbyte,
- einer Eingabetastatur,
- einem Bildschirmgerät,
- einem Diskettenlaufwerk.

Ein Drucker wäre wünschenswert.

1.5.2 Software-Ausrüstung

Unbedingt notwendig ist eine CP/M-Betriebssystemdiskette (CP/M-Diskette oder Systemdiskette), auf der ein Programmpaket mit dem CP/M-Betriebssystem gespeichert ist.

Das CP/M-Betriebssystem hat folgende wesentliche Aufgaben:

- Die Eingabe-Tastatur wird ständig nach Systemkommandos bzw. CP/M-Befehlen abgefragt.
- Es sorgt für die Speicherplatzverwaltung im Hauptspeicher und auf den Disketten (Dateiverwaltung).
- Es steuert weitere Peripheriegeräte.
- Es unterstützt die Programmentwicklung des Benutzers.

1.5.3 Allgemeine Struktur des CP/M-Betriebssystems

Das CP/M-Betriebssystem besteht aus einer Vielzahl von Dienstprogrammen (englisch: utilities). Die häufig gebrauchten Teile werden an einer vorbestimmten Stelle im Arbeitsspeicher geladen und nahezu ständig zur Durchführung von Aufgaben bereitgehalten. Weniger häufig gebrauchte Dienstprogramme werden nur bei Bedarf von der CP/M-Systemdiskette in den Arbeitsspeicher geladen und gestartet.

Der Hauptteil des CP/M-Betriebssystems besteht aus drei elementaren Teilen:

BDOS

Der erste Hauptteil ist das sogenannte BDOS, das Grundbetriebssystem für Diskettenspeicher (basic disk operating system). Alle Aufrufe eines CP/M-Befehls oder eines Dienstprogramms an das CP/M-Betriebssystem, d.h. Disketten-Ein-/Ausgabe, Druckerausgabe, Bidschirmausgabe, werden mit einem Satz von Standard-Sprungbefehlen (call) in das BDOS ausgeführt. Mit anderen Worten, das BDOS dient zur Verwaltung von Dateien, führt die Datenübertragungen von und zu den Disketten in der richtigen Form aus und überprüft die Richtigkeit der Datenübertragung. Dieser Teil ist unabhängig von der Hardware für alle Rechner gleich.

CCP

Der zweite Hauptteil ist der sogenannte CCP, der Konsolen-Bedienungsprozessor (console command processor). Der CCP analysiert und interpretiert die über die Tastatur eingegebenen CP/M-Befehle und initialisiert dann die gewünschten Operationen. Je nach Art des eingegebenen Kommandos wird die Bearbeitung an das BDOS oder BIOS (siehe unten) weitergegeben. Der CCP kontrolliert die korrekte Ausführung der Befehle und gibt dem Benutzer Meldung über die korrekte Ausführung und Beendigung des Kommandos bzw. über eventuelle Fehler. Fester Bestandteil des CCP sind die residenten, d.h. eingebauten bzw. dauernd gespeicherten, CP/M-Befehle. Sie können sofort ausgeführt werden, da sie schon im Arbeitsspeicher stehen. Vom internen (residenten) CP/M-System nimmt der CCP den kleinsten Speicherbereich ein. Auch er ist unabhängig vom verwendeten Rechner.

BIOS

Der letzte Hauptteil ist das BIOS, das Grundbetriebssystem für die Ein-/Ausgabe (basic input/output system). Das BIOS ist der CP/M-Teil, der es ermöglicht, CP/M auf eine Vielzahl verschiedener Rechner anzupassen. Jeder Rechner hat ein auf ihn besonders zugeschnittenes BIOS, das die Hardware und die daran angeschlossenen Peripheriegeräte unterstützt. Über das BIOS erfolgt jede Ein- und Ausgabe, wie über die Tastatur, auf Bildschirm, Diskettenlaufwerke, Drucker usw. Das CP/M-BIOS ist dem Betriebssystem des C 128 (Kernal) sehr ähnlich. Wie das Kernal enthält das BIOS einen Satz von Standardroutinen, die einen Zugriff auf die Hardware-Funktionen ermöglichen. Für den Commodore 128 wurde ein spezielles zweiteiliges BIOS geschrieben, das ein direktes Arbeiten mit den Commodore-Peripheriegeräten über die serielle VC- oder eine zusätzliche RS-232-Schnittstelle erlaubt.

Diese drei Bestandteile des CP/M-Betriebssystems werden mit einem einfachen Ladeprogramm in den Arbeitsspeicher des Mikrocomputers geladen (boot). Der noch freibleibende Bereich des Arbeitsspeichers wird bei CP/M transienter Programmbereich TPA (transient program area) genannt. Dies ist der Speicherbereich, in dem alle Programme (Anwender- und Dienstprogramme) geladen und gestartet werden.

Er dient auch zur Aufnahme der transienten CP/M-Befehle. Diese sind CP/M-Befehle in Form von Programmen, also Dienstprogramme, die nicht dauernd im Arbeitsspeicher geladen sind wie die residenten CP/M-Befehle des CCP.

Die transienten CP/M-Befehle werden also nur bei Bedarf von der Systemdiskette in den Arbeitsspeicher geladen und gestartet. Dadurch wird ein möglichst großer Teil des Arbeitsspeichers für Anwenderprogramme freigehalten.

Viele Mikrocomputer-Betriebssysteme verwenden eine Zero-Page im Speicher, um wichtige Werte bereitzustellen. Sowohl der Commodore 128 als auch das CP/M-Betriebssystem arbeiten mit einer Zero-Page.

Die prinzipielle Aufteilung des Arbeitsspeichers unter CP/M zeigt Bild 1.4.

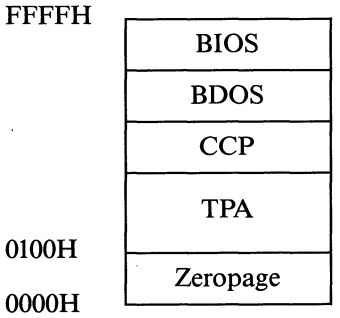


Bild 1.4: *Prinzipielle Speicheraufteilung unter CP/M*

2 CP/M 3.0 auf dem Commodore 128

2.1 Einleitung

Der 64er-Modus und der 128er-Modus des CBM-Rechners Commodore 128 sind enge Verwandte und kommen aus der Welt der *Homecomputer*. Der dritte Modus, der CP/M-Modus, gehört jedoch zur Welt der *Personal Computer* für geschäftliche Anwendungen. Bei den Homecomputern wurden die Graphik- und Musikfähigkeiten der Geräte sowie die BASIC-Programmiersprache verbessert, während bei den Personal Computern ein 80-Zeichen-Bildschirm und schnelle Diskettenoperationen im Vordergrund stehen, wobei hier das CP/M-Betriebssystem richtungweisend war.

Die Entwicklungen beider Richtungen liefen - wohl auch aus Kostengründen - bisher getrennt. Mit dem C-128-Rechner stehen dem Anwender nun beide Welten zur Verfügung.

Der normale Homecomputer-Benutzer bemerkt meist gar nicht, daß ein Betriebssystem in seinem Rechner vorhanden ist. Nach dem Einschalten des Rechners befindet man sich sofort im BASIC. Alle Fähigkeiten des Rechners erscheinen als solche des BASIC: Lesen der eingetippten Zeichen, Ausgabe von Informationen auf Bildschirm und Drucker, Einrichten, Beschreiben und Lesen von Dateien auf Disketten usw. Alle diese Eigenschaften werden jedoch nicht von BASIC zur Verfügung gestellt, sondern vom Betriebssystem des C 128 im C-64- oder C-128-Modus, wobei dieses als Kernal bezeichnete Betriebssystem hinter dem eingebauten BASIC verborgen ist.

Anders ist es im CP/M-Modus. Hier arbeitet der Benutzer direkt mit dem Betriebssystem CP/M.

Dieses versteht nur eine relativ bescheidene Anzahl von Befehlen, die sich hauptsächlich auf Disketten und Dateien beziehen. Aber CP/M kann einen 80-Zeichen-Bildschirm bedienen, und seine Diskettenoperationen sind wirklich schnell. Während die Datenübertragung zwischen Rechner und 1571 im C-128-Modus mit einer Übertragungsrate von 1500 Baud (Bits pro Sekunde) abläuft, beträgt diese unter CP/M 3500 Baud, d.h., sie ist mehr als zehnmal so groß wie im C-64-Modus.

Schnelle Diskettenoperationen und ein 80-Zeichen-Bildschirm nutzen aber nicht viel, wenn man keine Programme hat, die diese Möglichkeiten ausnutzen. Für CP/M gibt es nun weltweit fast 10 000 Programme, annähernd so viele wie für den C 64. Die überwiegende Mehrzahl der C-64-Programme dienen der Unterhaltung, bei CP/M ist es gerade umgekehrt; mehr als 90 % der CP/M-Programme dienen der kommerziellen Anwendung, d.h., es sind echte Profiwerkzeuge. Hier sind beispielsweise zu nennen:

- WordStar, der Klassiker unter den Textverarbeitungssystemen,
- dBASE II, ein Datenbanksystem und
- VisiCalc, ein Programmpaket zur Tabellenkalkulation.

Diese drei reichen zum Managen eines kleinen Betriebes aus.

Daneben gibt es unter CP/M noch zahlreiche Branchenlösungen, die auf die speziellen Belange eines Betriebes zugeschnitten sind und Aufgaben der Lagerhaltung, Fakturierung, Lohn- und Gehaltsabrechnung übernehmen.

Wenn Sie als Programmierer außer BASIC auch andere höhere Programmiersprachen anwenden wollen, so stehen Ihnen die meisten in einer ernstzunehmenden Version zur Verfügung, z.B. FORTRAN, Pascal, C, PL/1, COBOL, Ada, Modula II und sogar LISP oder PROLOG. Beispiele hierfür sollen noch in einem späteren Abschnitt behandelt werden.

Wie schon erwähnt, laufen CP/M und die dafür erhältliche Software nur auf Rechnern, die als Prozessor einen 8080 oder einen dazu kompatiblen Baustein aufweisen.

CP/M hat eine lange Entwicklung hinter sich. Es war das erste Betriebssystem für Mikrocomputer und ist wohl deswegen zum Industriestandard geworden. Die Mikrocomputer der ersten Generation konnten nur Arbeitsspeicher von maximal 64 Kbyte verwalten. Als die Preise der Speicherbausteine kostengünstiger wurden, hat die Firma Digital Research reagiert und das CP/M-Betriebssystem völlig überarbeitet. Es entstand die CP/M-Version 3.0, die einen Arbeitsspeicher von über 64 Kbyte bedienen

kann und auch sonst noch viele Verbesserungen aufweist. Diese letzte CP/M-Version 3.0 für 8-Bit-Rechner wurde von Commodore auf dem C 128 installiert.

Ein weiteres CP/M-Problem wurde von Commodore auf elegante Weise gelöst: das der Diskettenformate.

Es gibt über 80 verschiedene CP/M-Diskettenformate. CP/M hat zwar die Welt der 8-Bit-Personal-Computer standardisiert, aber die verschiedenen Computerhersteller haben sich für fast jeden neuen CP/M-Rechner ein neues CP/M-Diskettenformat einfallen lassen.

Eigentlich kann jeder CP/M-Rechner Daten und Programme jedes anderen CP/M-Rechners lesen, wenn die Diskettenformate übereinstimmen. Mit dem C 128 ist es unter CP/M (mit dem Laufwerk 1571 oder 1570) möglich, 9 Diskettenformate zu lesen. Weitere Formate können zusätzlich implementiert werden. Mit diesem Daten- und Programmaustausch stehen alle CP/M-Disketten und damit die zahlreichen CP/M-Programmpakete dem Anwender zur Verfügung.

2.2 CP/M-Modus

Im C 128 wird für das CP/M-Betriebssystem der Zusatzprozessor Z80 von Zilog verwendet, der eine Weiterentwicklung des 8080 darstellt. Der hier verwendete Zusatzprozessor Z80 besitzt einen wesentlich erweiterten Befehlssatz als der 8080-Prozessor; d.h., der Befehlssatz des 8080 ist eine Untermenge des Z80-Befehlssatzes. Jedes Programm, das für den 8080 geschrieben wurde, läuft auch auf dem Z80, aber nicht umgekehrt.

Die für den Commodore 128 lieferbare Version des CP/M-Betriebssystems ist CP/M 3.0 bzw. CP/M Plus bzw. CP/M 80.

Dieses Betriebssystem ist nicht in einem ROM-Speicher des C 128 fest installiert, sondern wird immer von einer sogenannten CP/M-Systemdiskette in den RAM-Arbeitsspeicher des Rechners geladen, so auch beim C 128.

Das Betriebssystem CP/M 3.0 steuert und überwacht alle Funktionseinheiten des Rechners wie den Speicher und die Konsole (Bildschirm und Tastatur) sowie die angeschlossenen Diskettenlaufwerke, Drucker und andere Peripheriegeräte. CP/M verwaltet die Dateien auf den Disketten in den angeschlossenen Laufwerken und ermöglicht so das Kopieren von Dateien in den Arbeitsspeicher oder zu anderen Peripheriegeräten, wie z.B. zu einem Drucker. Mit CP/M können also Programme im Arbeitsspeicher des Rechners abgelegt und in Abhängigkeit von Befehlen, die über die Tastatur der Konsole eingegeben werden, ausgeführt werden. Unter CP/M kann man eigene Programme erstellen oder bereits vorhandene Programme anwenden.

Dieses Handbuch soll dem Leser eine Einführung in das auf dem C 128 implementierte CP/M-Betriebssystem geben und gleichzeitig Hilfestellung für eigene Anwendungen vermitteln. Die CP/M-Implementierung auf dem C 128 unterscheidet sich in einigen Punkten vom normalen CP/M-Betriebssystem, wie es in einschlägigen Fachbüchern und in der Firmenliteratur von Digital Research beschrieben wird (siehe Literaturverzeichnis, Anhang A). Für weiterführende Informationen sollte der Leser die zahlreiche CP/M-Literatur zu Rate ziehen.

2.3 Mindestvoraussetzungen

Die notwendigen Voraussetzungen zum Anwenden des Betriebssystems CP/M 3.0 sind

- ein Computer mit einem 8080-, 8085- oder Z80-Mikroprozessor mit 64 Kbyte RAM-Speicher
- eine Konsole (Tastatur und Bildschirm)
- ein Diskettenlaufwerk.

Im Commodore 128 ist der Mikroprozessor Z80 bereits fest eingebaut.

2.3.1 Konsole

Als Konsole dienen die vollständige Tastatur des C 128 und eine bei CP/M übliche Bildschirmanzeige von 80 Zeichen pro Zeile (mit einem Bildschirmgerät am RGB-Ausgang des C 128).

Es kann auch mit einer Bildschirmanzeige von 40 Zeichen pro Zeile gearbeitet werden, z.B. bei Anschluß eines Monitors am Composite-Video-Ausgang des C 128. Um hierbei alle 80 Zeichen einer Zeile nacheinander sichtbar zu machen, muß der Bildschirminhalt horizontal verschoben werden. Dieses seitliche Scrollen erfolgt mit der CONTROL-Taste und gleichzeitigem Drücken der entsprechenden Cursor-Tasten (links oder rechts).

Die C-128-Konsole emuliert ein Terminal ADM-31 bzw. ADM-3A. Die gleiche Emulation findet sich z.B. beim Kaypro II und IV.

2.3.2 Diskettenlaufwerke

Als Diskettenlaufwerke können die Laufwerke Commodore 1571, 1570 und 1541 verwendet werden. Bei den neuen Laufwerken 1571 und 1570 erfolgt die Datenübertragung wesentlich schneller als beim alten 1541. Das neue 1571 beschreibt die Disketten auf beiden Seiten, während das 1570 und 1541 diese nur einseitig beschreiben. Das bedeutet, daß mit dem Laufwerk 1571 und 1570 ohne Schwierigkeiten auch mit solchen Disketten gearbeitet werden kann, die mit dem Laufwerk 1541 erzeugt wurden. Umgekehrt können mit den Laufwerken 1541 und 1570 auch Disketten vom Laufwerk 1571 bearbeitet werden, solange die Diskettendatei sich auf der ersten Diskettenseite befindet.

2.3.3 Andere Diskettenformate

Mit den Laufwerken 1571, 1570 und 1541 können Disketten bearbeitet werden, die unter dem Betriebssystem CP/M 80 bzw. CP/M 3.0 in den Formaten von

EPSON QX 10, Valdocs,
Kaypro II oder
Osborne

und unter dem Betriebssystem CP/M 86 im Format des

IBM PC bzw.
Commodore PC 10/20

erstellt wurden. Eine Erweiterung auf andere Formate ist mit Zusatzprogrammen möglich.

Nach dem ersten Zugriff auf eine Diskette wird in einem Leuchtbalken in der untersten Bildschirmzeile die Systembezeichnung des Diskettenformats angegeben bzw., wenn das Format nicht identifiziert werden konnte, erscheint die Fehlmeldung »MISSING«.

2.3.4 Programm- und Datenaustausch

Alle Programme und Daten auf Disketten in einem der obigen Formate können beliebig gelesen, bearbeitet und übertragen werden, so daß ein direkter Programm- und Datenaustausch mit anderen Rechnersystemen ohne weitere Hilfsmittel möglich ist.

Viele kommerzielle Programmpakete (Software) enthalten ein Installationsprogramm, das im allgemeinen auch eine Anpassung an das Terminal ADM-31 erlaubt, so daß keine Anpassungsschwierigkeiten wie beim C-64-CP/M bestehen (siehe Abschnitt 4).

2.4 Erste Schritte mit CP/M 3.0 auf dem C 128

2.4.1 Laden des CP/M-Betriebssystems

Beim Laden (boot) des CP/M-Betriebssystems wird dieses von der CP/M-Systemdiskette in den Speicher (RAM) des Rechners geschrieben.

Dies kann beim C 128 auf verschiedene Arten vorgenommen werden.

Kaltstart

Bei ausgeschaltetem Rechner wird die CP/M-Systemdiskette in das Diskettenlaufwerk eingelegt. Nach dem Einschalten des Rechners wird CP/M automatisch geladen.

Aus dem C-128-Modus

Befindet sich der C 128 im BASIC-Modus, kann CP/M von der vorher in das Laufwerk eingelegten CP/M-Systemdiskette mit dem BASIC-Befehl

```
BOOT
```

geladen werden. Alternativ kann das Laden von CP/M durch Betätigen der RESET-Taste gestartet werden.

Aus dem C-64-Modus

Aus dem C-64-Modus kann das Laden von CP/M nur nach vorherigem Ausschalten und nachfolgendem Einschalten des Rechners erfolgen (siehe Kaltstart).

Beim Laden von CP/M erscheint auf dem Bildschirm folgende Meldung:

```
         BOOTING CP/M PLUS
           DATA TABLES
           COMMON CODE
           BANKED CODE
           BIOS8502 CODE
BNKBIOS3 SPR  F400  0800
BNKBIOS3 SPR  CA00  1600
RESBDOS3 SPR  EE00  0600
BNKBDOS3 SPR  9600  2E00
59K TPA
                              ...(letzte Zeile)...        tt ss
```

In der letzten Zeile des Schirmbildes werden am linken Rand während des Ladens die gerade gelesenen Spuren (tt) und Sektoren (ss) der CP/M-Systemdiskette angezeigt.

Das CP/M-3.0-Betriebssystem befindet sich nicht wie bei den älteren Versionen auf den Systemspuren (Spur 0 und 1), sondern in der Systemdatei CPM+.SYS auf der Diskette. Die Systemspur der CP/M-3.0-Systemdiskette enthält nur ein kleines Ladeprogramm zum Laden der Systemdatei.

Die CP/M-Systemdatei CPM+.SYS besteht aus verschiedenen Teilen. Der Name des Teils, der gerade geladen wird, erscheint nacheinander im ersten Block. Der zweite Block zeigt die geladenen BIOS- und BDOS-Teile mit ihren Anfangsadressen und Dateilängen an.

Auf dem C 128 steht im CP/M-Modus ein freier Arbeitsspeicher TPA (transient program area) von 59 Kbyte zur Verfügung.

2.4.2 CP/M-Bereitschaftsmeldung

Nach Abschluß des Ladevorgangs erscheint auf dem Bildschirm die Meldung:

```
CP/M 3.0 on the Commodore 128 3 June 85
80 column display
A>
```

Wenn ein 40-Zeichen-Bildschirm gewählt wurde, erscheint für 80 die Zahl 40.

Als letzte Zeile werden die CP/M-Bereitschaftsmeldung (system prompt) »A>« und dahinter ein blinkender Cursor ausgegeben. CP/M wartet auf die Eingabe von Anweisungen und Befehlen durch den Benutzer.

Diskettenlaufwerke

Die Bereitschaftsmeldung »A>« zeigt an, daß das Laufwerk A das sogenannte aktuelle bzw. Bezugslaufwerk ist. Das bedeutet, daß bis zu einer anderen Laufwerksangabe alle Diskettenoperationen auf dem Laufwerk A durchgeführt werden.

Es können maximal vier Laufwerke mit den Bezeichnungen A, B, C und D angeschlossen werden. Ein Einzellaufwerk mit der Bezeichnung A

entspricht im C-128- bzw. C-64-Modus der Geräteadresse 8 (Laufwerk 0) usw.

Benutzerbereich

Bei einer vollständigen Bereitschaftsmeldung ist vor der Laufwerksangabe noch zusätzlich die Nummer u des Benutzerbereiches vorangestellt.

```
uA>
```

Beim Benutzerbereich 0 fehlt die Angabe »0«. Insgesamt sind 16 Benutzerbereiche möglich (0-15).

2.4.3 Umschalten des Bezugslaufwerks

Die Systembereitschaftsmeldung enthält die Angabe, welches Diskettenlaufwerk gerade zugeschaltet (aktiv) ist. Bei Anschluß von mehreren Diskettenlaufwerken ist ein Umschalten zwischen den Laufwerken leicht möglich, indem der neue Laufwerksname mit nachfolgendem Doppelpunkt nach der Bereitschaftsmeldung eingegeben wird.

Beispiel

```
A>B:<CR>
B>
```

Wenn der neue Laufwerksname als Bereitschaftsmeldung auf dem Bildschirm erscheint, ist dieses neue Laufwerk das Bezugslaufwerk, auf dem jetzt die danach eingegebenen CP/M-Befehle ausgeführt werden. Unter CP/M 3.0 auf dem C 128 sind insgesamt 4 Diskettenlaufwerke mit den Bezeichnungen A bis D anschließbar, wobei das erste Laufwerk mit der Gerätenummer 8 immer mit A, das zweite Laufwerk mit der Gerätenummer 9 mit B usw. bezeichnet wird.

2.4.4 CP/M-Befehlszeile

Der Rechner wird mit Befehlen gesteuert. Die Eingabe der CP/M-Befehle erfolgt im allgemeinen über die Tastatur und erscheint als CP/M-Befehlszeile nach der CP/M-Bereitschaftsmeldung auf dem Bildschirm.

Jedes Zeichen, das über die Tastatur eingegeben wird, erscheint auf dem Bildschirm. Gleichzeitig bewegt sich der Cursor um eine Stelle nach rechts. Ein CP/M-Befehl kann in Klein- oder Großbuchstaben eingegeben

werden. Dabei sollte der Befehl unmittelbar nach der Bereitschaftsmeldung erfolgen; Leerstellen werden jedoch toleriert.

Format: **A>befehl<CR>**

Jeder CP/M-Befehl wird durch Drücken der RETURN-Taste abgeschlossen und an das Betriebssystem zur Ausführung übergeben. Dieser Vorgang wird im folgenden mit <CR> abgekürzt (carriage return = Wagenrücklauf).

Ein Tippfehler kann sehr einfach korrigiert werden, indem der Cursor mit der INST/DEL-Taste oder CTRL-H entsprechend nach links zurückgesetzt und die Korrektur eingefügt wird. Ein Control-Zeichen wird durch gleichzeitiges Drücken der CTRL-Taste und der entsprechenden zweiten Taste erzeugt. Eine Zusammenstellung der Control-Steuerzeichen und deren Bedeutung ist in Tabelle 4.1 zu finden.

2.4.5 Funktionstasten

Einige wichtige CP/M-Befehle sind auf die Funktionstasten gelegt.

```
F1   -
F2   DIR<CR>
F3   DIR
F4   -
F5   -
F6   -
F7   -
F8   3 June 85 (Datum der CP/M-Version)
```

Die Belegung der Funktionstasten kann geändert werden (siehe Abschnitt 4.6).

3 CP/M-Dateien

3.1 Einleitung (Programm- und Datendatei)

Ein Programm oder eine abgeschlossene Einheit von Daten wird als Datei (file) bezeichnet. Dies könnten z.B. auch kompilierte (im Maschinencode überführte) Unterprogramme sein. Ebenso sind die externen CP/M-Befehle spezielle Programme, die in der Form von COM-Dateien auf der CP/M-Systemdiskette gespeichert sind. Eine Datei ist also eine Zusammenfassung von Informationen, wobei man zwischen einer Programmdatei, die ein Programm mit ausführbaren Befehlen enthält, und einer Datendatei, die weiterverarbeitbare Daten enthält, unterscheidet.

Eine der wichtigsten CP/M-Eigenschaften ist das Verwalten und Bearbeiten von Dateien auf Disketten und im Arbeitsspeicher TPA.

3.2 Bezeichnung einer CP/M-Datei

3.2.1 Format einer CP/M-Datei

Die Dateien lassen sich mit Dateinamen unterscheiden. Für CP/M-Dateien können die Dateibezeichnungen nicht willkürlich gewählt werden. Es sind einige Regeln zu beachten.

Für die Bezeichnung einer CP/M-Datei werden folgende Teile verwendet:

- die Laufwerksbezeichnung
- der Dateiname
- der Dateityp
- das Paßwort.

Format: **d:dateiname.typ;paßwort**

Es bedeuten:

d	eine frei wählbare Angabe eines Diskettenlaufwerkes (A bis E), in dem sich die zu bearbeitende Diskette befindet. d muß nur dann angegeben werden, wenn sich die Datei »dateiname.typ« nicht im aktiven Laufwerk (Bezugslaufwerk) befindet.
dateiname	ein willkürlicher Dateiname aus bis zu 8 Zeichen.
typ	eine frei wählbare Dateikennung bzw. Typbezeichnung aus 0 bis 3 Zeichen, mit der die Dateiklasse, d.h. die Art des Inhalts der Datei, gekennzeichnet wird.
paßwort	eine frei wählbare Ergänzung und damit Teil der Dateibezeichnung aus bis zu 8 Zeichen.

Beschreibung

Man sollte den Dateinamen, die Typbezeichnung und das Paßwort so wählen, daß man auf den Inhalt der Datei zurückschließen kann. Unbedingt erforderlich ist nur die Angabe des Dateinamens. Alle anderen Teile der Dateibezeichnung können, müssen aber nicht unbedingt angegeben werden. Alle Teile müssen mit ihren passenden Zeichen getrennt werden.

Ohne Angabe der Laufwerksbezeichnung wird die Datei auf dem aktuellen bzw. aktivierten Laufwerk gesucht.

Das Paßwort kann wahlweise eingegeben werden; es muß allerdings immer dann als Teil der Dateibezeichnung eingegeben werden, wenn eine Datei mit einem Paßwort geschützt ist.

Im folgenden wird eine vollständige Dateibezeichnung mit »datei« abgekürzt.

Beispiele

`DATEI`	Eine Datei mit dem Namen DATEI.
`PROGRAMM.COM`	Ein kompiliertes, ausführbares Programm auf der Diskette im Bezugslaufwerk.
`A:BEISPIEL.BAS`	Ein BASIC-Programm mit dem Namen BEISPIEL, das auf der Diskette im Laufwerk A bearbeitet werden soll.
`B:TEST.FOR;PW`	Ein FORTRAN-Programm mit dem Namen TEST und dem Paßwort PW, das auf der Diskette im Laufwerk B bearbeitet werden soll.

3.2.2 Reservierte Zeichen

In einer CP/M-Dateibezeichnung können alle alphanumerischen Zeichen verwendet werden, mit Ausnahme einiger Sonderzeichen, die mit einer besonderen Bedeutung reserviert sind (Tabelle 3.1).

Tabelle 3.1: Reservierte Zeichen

Zeichen/Taste	Bedeutung
TAB CR	Trennzeichen einer Befehlszeile
Leerstelle	
:	Laufwerkstrennzeichen
.	Dateityptrennzeichen
;	Paßworttrennzeichen oder Kommentartrennzeichen (am Anfang einer Befehlszeile)
* ?	Platzhalter
[ ]	Trennzeichen für CP/M-Befehlsargumente
< > = , ! & $ ( ) \ + -	weitere reservierte Zeichen

3.2.3 Reservierte Typbezeichnungen

Mit der Dateikennung »typ« werden die CP/M-Dateien nach der Form des Inhalts klassifiziert. In Tabelle 3.2 sind die allgemein verwendeten CP/M-Dateikennungen wiedergegeben. Die mit einem Stern »*« markierten Dateikennungen sind bei dieser CP/M-Version entsprechend der Tabelle mit besonderen Bedeutungen reserviert.

Tabelle 3.2: CP/M-Dateikennungen (Dateityp)

Typ	Bedeutung
*ASM	Quelldatei im Assemblercode (für ASM)
*BAK	Sicherungsdatei (backup)
BAS	BASIC-Programm
CMD	Befehlsdatei unter dBASE
*COM	direkt ausführbares, externes Programm
DAT	Datendatei
DOC	Dokumentations- oder Textdatei
*FOR	FORTRAN-Programm
*HEX	Datei oder Programm im Intel-Hexadezimal-Maschinencode (von MAC für HEXCOM)
*HLP	HELP-Datei
LIB	Bibliotheksdatei
LST	List- bzw. Printdatei eines Programmes, von einem Compiler oder Interpreter ausgegeben.
NDX	Indexdatei unter dBASE
*PAS	Pascal-Programm
*PRN	Printdatei (von MAC oder RMAC)
PRT	Printdatei
*REL	verschiebbare Objektdatei (von RMAC für LINK)
*SUB	Programmdatei mit CP/M-Befehlen (für SUBMIT)
*SYM	Datei- und Symboltabellen, von einigen Compilern und Debuggern erzeugt (von MAC, RMAC, LINK, SID, ZSID)
TEX	Textdatei
TXT	Textdatei
*$$$	temporäre Zwischendatei

3.2.4 Platzhalter

Bei einigen CP/M-Befehlen können die Ersatzzeichen »*« und »?« im Dateinamen »dateiname« und in der Dateikennung »typ« verwendet werden, so daß sich dieser Befehl auf mehrere Dateien beziehen kann. Ein Ersatzzeichen ist hierbei Platzhalter für ein oder mehrere beliebige, zulässige Zeichen in der Dateikennung.

Diese Zeichen bedeuten folgendes:

Fragezeichen »?«	Ein Fragezeichen »?« ersetzt ein einzelnes, beliebiges Zeichen an der Stelle, an der es steht (nicht jedoch mehrere hintereinanderstehende Zeichen).
Stern »*«	Ein Stern »*« ersetzt einen vollständigen Dateinamen, eine Dateikennung oder Teile daraus aus beliebigen Zeichen.

Beispiele

`*.BAK`	Es werden alle Dateien des Typs BAK angesprochen.
`PGM*.BAK`	Es werden alle Dateien des Typs BAK angesprochen, deren Dateinamen mit den ersten drei Zeichen PGM beginnen.
`PGM?.BAK`	Es werden alle Dateien des Typs BAK angesprochen, deren 3- bis 4stelliger Dateiname mit den ersten drei Zeichen PGM anfängt, z.B. PGM.BAK, PGM1.BAK, nicht jedoch PGM12.BAK.
`*.*`	Es werden alle Dateien angesprochen.

3.2.5 Benutzerbereich

Wie oben schon erwähnt, kann CP/M die Dateien auch einem Benutzerbereich (User 0-15) zuordnen, d.h. mit den Benutzerbereichsnummern können Dateien in 16 verschiedene Bereiche eingeteilt werden, z.B. beim Erstellen einer Datei.

Format: **nd> (CP/M-Bereitschaftsmeldung) oder**
nd: (Dateibezeichnung)

Es bedeuten:

n	Nummer des Benutzerbereichs (user), eine Zahl zwischen 0 und 15
d	Angabe eines Diskettenlaufwerkes

Beschreibung

Die Nummer des Benutzerbereichs wird immer der Laufwerksangabe vorangestellt. Der Benutzerbereich 0 ist nach einem Kaltstart der voreingestellte Bereich, seine Nummer wird in der Bereitschaftsmeldung nicht angezeigt.

Sind Dateien mit einem Benutzerbereich geschützt, so haben die meisten CP/M-Befehle nur auf die Dateien einen Zugriff, wenn dieser Benutzerbereich der aktuelle ist. Allerdings kann man aus allen Benutzerbereichen auf die Dateien mit einem SYS-Attribut im Benutzerbereich 0 zugreifen.

Beispiele

`A>`	**Laufwerk A, Benutzerbereich 0**
`3B>`	**Laufwerk B, Benutzerbereich 3**
`A>2B:<CR>2B>`	**Das Bezugslaufwerk und der aktuelle Benutzerbereich werden vom Laufwerk A und Benutzerbereich 0 auf Laufwerk B und Benutzerbereich 2 geändert.**

3.3 Disketten-Inhaltsverzeichnis

Auf jeder Diskette legt CP/M ein Inhaltsverzeichnis (directory) der auf der Diskette gespeicherten Dateien an. Mit dem CP/M-Befehl »DIR« kann dieses Disketten-Inhaltsverzeichnis auf dem Bildschirm angezeigt werden (siehe Kapitel 6.6).

```
A>DIR<CR>
```

Die Systemdiskette für CP/M 3.0 sollte folgendes Inhaltsverzeichnis auf dem Bildschirm auslisten:

```
A: CPM+    SYS : CCP    COM : HELP   COM : HELP   HLP : KEYFIG COM
A: KEYFIG  HLP : FORMAT COM : PUT    COM : DIRLBL RSX : DATE   COM
A: COPYSYS COM : DATEC  ASM : DATEC RSX : DEVICE COM : DIR    COM
A: DUMP    COM : ED     COM : ERASE COM : GENCOM COM : GET    COM
A: INITDIR COM : PATCH  COM : PIP    COM : RENAME COM : SAVE   COM
A: SET     COM : SETDEF COM : SHOW   COM : SUBMIT COM : TYPE   COM
A: SETUP   COM
A>
```

Will man das Disketten-Inhaltsverzeichnis sowohl auf dem Bildschirm als auch auf einem Drucker ausgeben, muß man vorher den Drucker mit

```
CTRL-P
```

aktivieren. Hierzu werden die CONTROL-Taste CTRL und die Taste »P« gleichzeitig gedrückt. Danach erst wird der auszuführende Befehl (hier »DIR«) eingegeben. Die Ausgabe auf dem Drucker wird durch nochmalige Eingabe von CTRL-P wieder ausgeschaltet. Außerdem kann die Ausgabe mit CTRL-S angehalten und mit dem Drücken einer beliebigen Taste fortgesetzt werden.

3.4 Kopieren von CP/M-Disketten und CP/M-Dateien

3.4.1 Sicherungskopien

Nach dem Laden und Starten von CP/M sollte man unbedingt eine oder mehrere Sicherungskopien (backup) der CP/M-Betriebssystemdiskette anfertigen, d.h., die mitgelieferte Diskette sollte man nicht als Arbeitsdiskette verwenden, denn es kann durchaus einmal beim Arbeiten mit CP/M zu einer unbeabsichtigten Zerstörung der Systemdiskette kommen; spätestens wenn sich der Rechner einmal *aufhängt*.

3.4.2 Formatieren einer Diskette

Bevor man irgendwelche Informationen auf eine Diskette schreiben kann, muß diese formatiert werden. Dazu muß ein geeignetes Formatierprogramm verwendet werden. Eine bereits benutzte CP/M-Diskette muß natürlich nicht formatiert werden. Auf der CP/M-Systemdiskette befindet sich ein geeignetes Programm FORMAT.COM, das gleichzeitig auf die formatierte Diskette auch den CP/M-System-Bootsektor überträgt.

Nach dem Programmaufruf

```
A>FORMAT
```

erscheint auf dem Bildschirm:

```
C 128 FORMAT PROGRAMM
      15 May 1985
Drive A is a 1571                (bzw. 1541)
Please select disk type to format
C 128 double sided               (nicht bei 1541)
C 128 single sided               (1570)
C 64 single sided                (C-64-Z80-Modul-Format)
```

Der Cursor wird in die Zeile des gewünschten Formats bewegt. Nach Drücken der RETURN-Taste erscheint:

```
Formating...
Insert Diskette TO BE FORMATED
in drive A. Type $ when ready,
any other key to abort
```

Man entfernt spätestens jetzt die Systemdiskette aus dem Laufwerk und legt die zu formatierende Diskette ein. Der Formatiervorgang wird mit dem Drücken der $-Taste gestartet. Alternativ kann man durch Drücken irgendeiner anderen Taste das Formatierprogramm abbrechen und ins CP/M zurückspringen.

Auf dem Bildschirm erscheinen beim Formatieren nacheinander die durchgeführten Vorgänge:

```
  formating diskette in drive a
   writing directory sectors
    writing boot sectors

Do you want to format another disk?
```

Durch Drücken von »Y« kann eine weitere CP/M-Diskette formatiert werden. Mit Eingeben von »N« wird das Formatierprogramm beendet:

```
     exiting format program
A>
```

3.4.3 Kopieren der CP/M-Systemdateien

Alle CP/M-Systemdateien können mit dem Dienstprogramm PIP.COM kopiert werden. Bei einem Kaltstart wird das CP/M-Betriebssystem in den Systemdateien CPM+.SYS und CCP.COM vollständig neu geladen und alle Systemparameter neu initialisiert. Das Kopieren von CP/M-Dateien kann sowohl mit einem als auch mit zwei Diskettenlaufwerken erfolgen.

3.4.4 Kopieren mit einem Diskettenlaufwerk

Zum Kopieren von CP/M-Disketten können sowohl das alte Laufwerk 1541 als auch die neuen 1571 und 1570 verwendet werden.

Das Dienstprogramm PIP.COM wird aufgerufen mit:

```
A>PIP<CR>
CP/M3 PIP VERSION 3.0
*
```

Nach der PIP-Bereitschaftsmeldung »*« werden das Quellaufwerk A und das Ziellaufwerk (hier E) sowie die zu kopierenden Dateien (hier alle Dateien mit *.*) eingegeben. Das Ziellaufwerk E ist ein virtuelles Laufwerk, d.h. dieses Laufwerk existiert nicht als Teil der Hardware.

```
*E:=A:*.*[V]<CR>
COPYING
CPM+.SYS
CCP.COM
usw.
```

Nacheinander werden alle zu kopierenden Dateien während des Kopiervorgangs aufgelistet. Die Angabe von [V] im PIP-Befehl bewirkt eine Verifizierung, d.h. einen zusätzlichen Vergleich der neugeschriebenen Datei mit der Originaldatei. Befindet sich schon eine Datei auf der Diskette, so wird eine andere Datei mit demselben Namen über die alte Datei geschrieben, so daß die alte Datei verlorengeht.

Während des Kopierens erscheint in der letzten Zeile des Bildschirms die Aufforderung, daß die virtuelle Diskette E, d.h. die zu beschreibende Diskette, in das Laufwerk A gelegt werden soll.

```
Insert Disk E in Drive A
```

Diese Aufforderung wird je nach Länge der zu kopierenden Dateien mehrmals für Diskette A und E wiederholt. Der Kopiervorgang wird jedesmal nach Einlegen der entsprechenden Diskette durch Drücken der RETURN-Taste fortgesetzt. Mit CTRL-C (Warmstart) kann das Kopierprogramm beendet werden.

Alternativ kann das Kopierprogramm PIP.COM auch direkt mit dem Kopieren beginnen:

```
A>PIP E:=A:*.*[V]<CR>
COPYING
CPM.SYS
CCP.COM
usw.
```

Auch hierbei wird man, wie oben beschrieben, zum mehrmaligen Wechseln der Disketten aufgefordert.

3.4.5 Kopieren mit zwei Laufwerken

Mit zwei Diskettenlaufwerken ist das Kopieren wesentlich bequemer und schneller als mit nur einem Laufwerk. Die Laufwerke haben in der Regel die Bezeichnungen A und B. Das Dienstprogramm PIP.COM wird wieder wie folgt gestartet:

```
A>PIP<CR>
CP/M3 PIP VERSION 3.0
*
```

Zum Kopieren aller Dateien der Systemdiskette im Laufwerk A auf eine neuformatierte Diskette in Laufwerk B gibt man ein:

```
*B:=A:*.*[V]<CR>
```

Während des Kopierens erscheinen wieder nacheinander die kopierten Dateien.

```
COPYING-
CPM+.SYS
CCP.COM
USW.
```

Mit CTRL-C (Warmstart) wird das Kopierprogramm PIP beendet. Auch hier kann wieder wie bei einem Laufwerk der Kopiervorgang direkt mit einem Befehl gestartet werden:

```
A>PIP B:=A:*.*[V]<CR>
```

4 Konsole und Drucker

Im folgenden werden die Steuerung der Konsoleneingabe und -ausgabe, einschließlich der Konsolen-Editierbefehle und der Druckerausgabe, sowie das Prinzip der logischen Geräte unter CP/M 3.0 beschrieben.

4.1 Steuerung der Konsolenausgabe

Oft erfolgt die Bildschirmanzeige zu schnell, um mitlesen zu können, z.B. wird ein langer Text schnell über den oberen Bildschirmrand gerollt. Mit CTRL-S, d.h. Drücken der CTRL-Taste und gleichzeitig der Taste S, kann die Bildschirmausgabe angehalten werden. Die gleiche Wirkung hat das Drücken der »NO SCROLL«-Taste. In der untersten Zeile (Zeile 25) erscheint die Meldung PAUSE in inverser Darstellung.

Mit CTRL-Q bzw. durch nochmaliges Drücken der »NO SCROLL«-Taste wird die Ausgabe fortgesetzt. Wird während der Pause eine andere Taste betätigt, ertönt ein akustisches Signal.

Einige CP/M-Dienstprogramme, z.B. DIR oder TYPE, erzeugen eine automatische seitenweise Konsolenausgabe (paging), d.h. bei einem längeren Text hält die Ausgabe nach einem vollständig gefüllten Bildschirm an. Das seitenweise Umschalten erfolgt entsprechend der Aufforderung in der letzten Zeile des Bildschirms

```
Press RETURN to continue
```

durch Drücken der RETURN-Taste.

Dieses seitenweise Umschalten kann mit dem CP/M-Befehl

```
SETDEF [NO PAGE]
```

unterdrückt werden.

4.2 Steuerung der Druckerausgabe

Will man eine Konsolenausgabe sowohl auf dem Bildschirm als auch auf einem Drucker ausgeben, muß man vorher den Drucker mit

CTRL-P

aktivieren. Jedes Zeichen wird auf dem Bildschirm und dem Drucker ausgedruckt. Die Ausgabe auf dem Drucker wird durch nochmalige Eingabe von »CTRL-P« wieder ausgeschaltet. Eine Ausgabe kann auch hierbei mit

CTRL-S

angehalten und mit

CTRL-Q

bzw. mit der »NO SCROLL«-Taste fortgesetzt werden, z.B. zum Erstellen einer *Hardcopy* des Disketten-Inhaltsverzeichnisses (DIR-Befehl) oder einer Programmdatei (TYPE-Befehl). Hierbei müssen die Control-Befehle vor Übergabe des CP/M-Befehls mit RETURN eingegeben werden.

4.3 Zeileneditierung auf der Konsole

Einfache Tippfehler können sofort mit der INST/DEL-Taste bzw. CTRL-H korrigiert werden. CP/M stellt zusätzliche Zeileneditierfunktionen als CTRL-Zeichen zur Verfügung. Mit diesen CTRL-Zeichen können CP/M-Befehlszeilen und Programmaufruf-Zeilen editiert werden, ohne daß die anderen Zeichen in der Befehlszeile gelöscht werden.

Mit den CTRL-Zeichen nach Tabelle 4.1 kann man den Cursor nach links und rechts bewegen, um ein Zeichen in der Befehlszeile einzufügen oder zu löschen. Rechts von der Korrekturstelle muß dabei kein Zeichen nochmals neu eingegeben werden.

Bei dieser CP/M-Version kann ein CP/M-Befehl durch Drücken der RETURN-Taste bei jeder beliebigen Cursorstellung in der Befehlszeile abgeschlossen werden; CP/M liest die vollständige Befehlszeile. Ebenso kann ein Befehl nochmals editiert und ausgeführt werden. Der RETURN-Befehl (bzw. CTRL-I oder CTRL-M) veranlaßt nicht nur die Ausführung der Befehlszeile, sondern speichert die Zeile auch in einem Puffer, so daß diese mit CTRL-W wieder zurückgerufen werden kann.

Wenn ein Zeichen in der Zeilenmitte eingefügt wird, werden alle Zeichen rechts vom Cursor weiter nach rechts bewegt. Wird die Zeile länger als die Bildschirmbreite, verschwinden die Zeichen auf der rechten Bildschirmseite. Diese Zeichen sind aber nicht gelöscht. Sie werden wieder angezeigt, wenn Zeichen in der Zeile wieder gelöscht werden oder wenn man CTRL-E drückt. CTRL-E schiebt alle Zeichen rechts des Cursors in die nächste Bildschirmzeile.

Werden CTRL-Zeichen auf dem Bildschirm angezeigt, wird ein »^« vor das Zeichen gesetzt, z.B. bei CTRL-C erscheint ^C.

Beispiel

`A>POP A:=B:*.*_`	PIP falsch eingegeben,
`POP?`	Fehlermeldung von CP/M,
`A>POP A:=B:*.*`	CTRL-W wiederholt die letzte Befehlszeile,
`A>POP A:=B:*.*`	CTRL-B setzt den Cursor an den Anfang der Befehlszeile,
`A>POP A:=B:*.*`	CTRL-F setzt den Cursor eine Stelle nach rechts,
`A>PP A:=B:*.*`	CTRL-G löscht den Fehler,
`A>PIP A:=B:*.*`	Korrektur durch Eingaben von I.

Tabelle 4.1: Zeileneditier-Steuerzeichen

Zeichen	Bedeutung
CTRL-A	Bewegt den Cursor ein Zeichen nach links.
CTRL-B	Bewegt den Cursor an den Anfang der Befehlszeile ohne Veränderung des Zeileninhalts. Wenn der Cursor sich am Zeilenanfang befindet, springt er an das Zeilenende.
CTRL-C	Unterbricht ein Programm und setzt die Laufwerksparameter auf die Anfangswerte (Warmstart).
CTRL-E	Bewirkt einen Zeilenvorschub ohne Übergabe des Befehls an CP/M (kein RETURN) und setzt den Cursor an den Anfang der nächsten Zeile, ohne die bisherige Eingabe zu löschen.
CTRL-F	Bewegt den Cursor eine Stelle nach rechts.
CTRL-G	Löscht das Zeichen unter dem Cursor, ohne den Cursor zu bewegen. Die Zeichen rechts vom Cursor werden eine Stelle nach links bewegt.
CTRL-H	Löscht das Zeichen links vom Cursor, indem der Cursor und alle Zeichen links von ihm um eine Stelle nach links gerückt werden. Die Funktion ist dieselbe wie die DEL-Taste.
CTRL-I	Bewegt den Cursor zur nächsten Tabulator-(TAB-)Marke. Tabulatormarken sind automatisch an jeder achten Stelle gesetzt. Die Funktion ist dieselbe wie die der TAB-Taste.
CTRL-J	Übergibt die Befehlszeile an CP/M und setzt den Cursor an den Anfang der nächsten Zeile. Die Funktion ist die gleiche wie die der RETURN-Taste oder CTRL-M.
CTRL-K	Löscht alle Zeichen rechts vom Cursor bis zum Zeilenende.
CTRL-M	Wie CTRL-J oder RETURN-Taste.
CTRL-P	Aktiviert die Druckerausgabe parallel zur Bildschirmausgabe. Nach der ersten Eingabe wird der Drucker aktiviert, eine nochmalige Ausgabe schaltet die Druckerausgabe wieder aus.

Zeichen	Bedeutung
CTRL-Q	Setzt die Bildschirmausgabe bei einer Unterbrechung mit CTRL-S fort.
CTRL-R	Zeigt die aktuelle Befehlszeile nochmals an, wobei ein Doppelkreuz »#« an die derzeitige Cursorposition gesetzt, der Cursor an derselben Stelle in die nächste Zeile bewegt und der bisher eingegebene Befehlsteil nochmals angezeigt wird.
CTRL-S	Unterbricht die Bildschirmanzeige während einer Ausgabe. Die Ausgabe kann mit CTRL-Q fortgesetzt werden.
CTRL-U	Löscht im Befehlsspeicher alle Zeichen einer Befehlszeile, setzt ein Doppelkreuz »#« an die aktuelle Cursorposition und bewegt den Cursor in die nächste Zeile. Mit CTRL-W können alle Zeichen, die in der alten Zeile links vom Cursor standen, in der neuen Zeile wiederholt werden.
CTRL-W	Zeigt die zuletzt eingegebene Befehlszeile erneut an, wenn CTRL-W die erste Eingabe nach der CP/M-Bereitschaftsmeldung »A>« ist. Wiederholt wird die letzte mit CTRL-J, CTRL-M, CTRL-U oder RETURN abgeschlossene Befehlszeile. Enthält die Befehlszeile bereits Zeichen, wird mit CTRL-W der Cursor ans Zeilenende gesetzt. Mit RETURN wird die wiederholte Zeile an CP/M übergeben und ausgeführt.
CTRL-X	Löscht alle Zeichen links vom Cursor und setzt den Cursor an den Zeilenanfang. Alle Zeichen rechts vom Cursor bleiben unverändert erhalten.
CTRL-Z	Dateiende-Zeichen.
INST/DEL	Wie CTRL-H
RETURN	Wie CTRL-J oder CTRL-M
TAB	Wie CTRL-I

4.4 Umleiten von Eingaben und Ausgaben

Der CP/M-Befehl »PUT« ermöglicht, die Ausgabe auf die Konsole oder einen Drucker in eine Diskettendatei umzuleiten. Der CP/M-Befehl »GET« dient dazu, eine Befehlseingabe für CP/M oder für ein Dienstprogramm aus einer Diskettendatei durchzuführen. Die nachfolgenden Beispiele sollen eine Möglichkeit von »GET« oder »PUT« illustrieren.

Man kann mit einem PUT-Befehl eine Konsolenausgabe in eine Diskettendatei umleiten, als ob es die Konsole wäre. Mit »PUT« kann man eine Diskettendatei erzeugen, die ein Disketten-Inhaltsverzeichnis aller Dateien der Diskette enthält. Ebenso kann mit einem PUT-Befehl eine direkte Druckerausgabe in einer Diskettendatei abgelegt werden, als ob es ein Drucker wäre.

Beispiele

a) Die Konsolenausgabe wird in der Datei DIR.PRN abgelegt

```
A>PUT CONSOLE OUTPUT TO FILE DIR.PRN
Putting console output to file: DIR.PRN

A>DIR <CR>
A: FILE     TEX : FRONT    TEX : FRONT    BAK
A: ONE      TEX : ONE      BAK : TWO      TEX
```

b) Die Konsolenausgabe wird nun wieder auf die Konsole geleitet.

```
A>PUT CONSOLE OUTPUT TO CONSOLE
Putting console output to console

A>TYPE DIR.PRN
A: FILE     TEX : FRONT    TEX : FRONT    BAK
A: ONE      TEX : ONE      BAK : TWO      TEX
```

Ein GET-Befehl ermöglicht CP/M oder einem Programm das Lesen einer Diskettendatei anstelle einer Konsoleneingabe über die Tastatur. Beim Lesen dieser Datei von CP/M müssen normale CP/M-Befehlszeilen in der Datei enthalten sein. Beim Lesen der Datei von einem Dienstprogramm müssen geeignete Eingabedaten für dieses Programm vorhanden sein. Eine solche Datei kann sowohl CP/M-Befehlszeilen als auch Programmeingabedaten enthalten, wenn das Programm vorher mit einem Befehl gestartet wurde.

Man kann die SYSTEM-Option in einer GET-Befehlszeile hinzufügen oder weglassen, je nachdem, ob CP/M oder ein Dienstprogramm die Datei lesen soll. Wenn man die SYSTEM-Option wegläßt, meldet sich CP/M mit der

Bereitschaftsmeldung, so daß das Programm gestartet werden kann, das die Eingabedaten aus der angegebenen Datei liest. Wenn man die SYSTEM-Option angibt, entnimmt CP/M die Eingabedaten aus der bezeichneten Datei.

Beispiel

```
3A>TYPE PIP.DAT
B:=FRONT.TEX
B:=ONE.TEX
B:=TWO.TEX

3A>GET CONSOLE INPUT FROM FILE PIP.DAT
Getting console input from file: PIP.DAT

3A>PIP
CP/M 3 PIP VERSION 3.0
*B:=FRONT.TEX
*B:=ONE.TEX
*B:=TWO.TEX
*^<CR>
```

4.5 Zuordnen von logischen Einheiten

Die minimale Hardware-Konfiguration des Commodore 128 besteht aus dem Rechner, einem Bildschirm und einem Diskettenlaufwerk. Man kann nun weitere physikalische Einheiten an das System anschließen, z.B. einen Drucker, ein Modem oder sogar ein *Joystick*. Um diese verschiedenen physikalischen Ein- und Ausgabegeräte anschließen zu können, verbindet CP/M diese verschiedenen physikalischen Einheiten mit logischen Einheiten. Die Zuordnung der logischen zu den physikalischen Einheiten beim C 128 ist in Tabelle 4.2 wiedergegeben.

Mit dem CP/M-Befehl DEVICE kann die Zuordnung geändert werden. Die logischen Einheiten AUXIN und AUXOUT können z.B. einer seriellen Schnittstelle zugeordnet werden, an die auch ein Modem angeschlossen werden kann. Beim C 128 kann diese physikalische Einheit der IC-Baustein 6551 sein, der einen ACIA-IC-Baustein emuliert und als serielle Schnittstelle verwendet werden kann.

Tabelle 4.2: Logische Einheiten beim C-128 - CP/M3

Logische Einheit	Gerätetyp	Physikalische Einheit
CONIN:	Konsoleneingabe	Tastatur (KEYS)
CONOUT:	Konsolenausgabe	Bildschirm (80COL oder 40COL)
AUXIN:	Externe Eingabe	-
AUXOUT:	Externe Ausgabe	-
LST:	List-Gerät	Drucker

4.6 Erweiterungen des CP/M 3.0 auf dem C 128

Beim Commodore 128 sind dem Betriebssystem einige sonst nicht im CP/M 3.0 enthaltene Erweiterungen hinzugefügt.

4.6.1 Änderung der Tastaturbelegung

Die Funktion der meisten Tasten der C-128-Tastatur kann geändert werden. Hiervon ausgenommen sind folgende Tasten:

- Linke SHIFT-Taste,
- Rechte SHIFT-Taste,
- COMMODORE-Taste C=,
- CONTROL-Taste CTRL,
- RESTORE-Taste,
- 40/80-Taste,
- ASCII/DIN-Taste (CAPS LOCK)
- Großschrift-Feststell-Taste (SHIFT LOCK).

Neben den normalen C-128-Tasten können auch die Funktionstasten mit Zeichenketten (Strings) belegt werden.

4.6.2 Belegen einer Taste mit einem Einzeichen-Code

Der bei einem Tastendruck erzeugte Zeichen-Code kann vom Benutzer umdefiniert werden. Jeder Taste sind vier mögliche Definitionen zugeordnet, je nachdem, welche Modus-Taste gerade gedrückt ist:

- Normal-Modus,
- Alpha-Shift-Modus,
- Shift-Modus
- Control-Modus.

Beim Normal-Modus wird nur eine Taste allein gedrückt; der Alpha-Shift-Modus wird mit der Commodore-Taste C= ein- bzw. ausgeschaltet; beim Shift-Modus muß eine Taste zusammen mit der SHIFT-Taste und beim Control-Modus zusammen mit der CTRL-Taste gedrückt werden.

Um diese Änderung durchzuführen, müssen insgesamt drei Tasten gleichzeitig gedrückt werden:

- zuerst die CTRL-Taste,
- dann die rechte SHIFT-Taste und
- zuletzt die Taste »CRSR links« (dunkelgraue Taste).

Danach erscheint ein inverses Feld in der untersten Bildschirmzeile. Die erste Taste, die jetzt gedrückt wird, ist die neu zu definierende Taste. Die bisherige Zuordnung wird in dem inversen Feld als Hexadezimalzahl angezeigt. Durch Eingabe einer anderen Hexadezimalzahl kann die Tastenfunktion geändert werden. Nach Eingabe der neuen Hexadezimalzahl wird diese automatisch übernommen. Soll die bisherige Hexadezimalzahl nicht geändert werden, kann mit der RETURN-Taste abgebrochen werden.

Tabelle 4.3: Übersicht über den Hexadezimal-Code für die Tastaturbelegung

Hex-Code	Bedeutung
00H	Null (keine Wirkung)
01H-7FH	Normaler ASCII-Code
80H-9FH	Zuweisung von Zeichenketten
A0H-AFH	Zeichenfarbe (80-Zeichen-Bildschirm)
B0H-BFH	Hintergrundfarbe (80-Zeichen-Bildschirm)
C0H-CFH	Zeichenfarbe (40-Zeichen-Bildschirm)
D0H-DFH	Hintergrundfarbe (40-Zeichen-Bildschirm)
E0H-EFH	Rahmenfarbe (40-Zeichen-Bildschirm)
F0H	Disketten-Status-Anzeige umschalten
F1H	XON/XOFF-Bildschirmanzeige umschalten
F2H	nicht definiert
F3H	linke obere Fensterecke definieren (40-Zeichen-Bildschirm)
F4H	Rechte untere Fensterecke definieren (40-Zeichen-Bildschirm)
F5H-FFH	Nicht definiert

4.6.3 Belegen einer Taste mit einer Zeichenkette

Mit dieser Änderung ist es möglich, einer einzigen Taste (auch einer Funktionstaste) nicht nur ein Zeichen, sondern eine Zeichenkette (String) zuzuordnen. Um diese Funktion durchzuführen, müssen insgesamt drei Tasten gleichzeitig gedrückt werden:

- zuerst die CTRL-Taste
- dann die rechte SHIFT-Taste und
- zuletzt die Taste »CRSR rechts« (dunkelgraue Taste).

Es erscheint ein langes inverses Feld in der untersten Bildschirmzeile. Die erste Taste bzw. Funktionstaste, die jetzt gedrückt wird, ist die neu zu definierende Taste. Die bisherige Zuordnung wird in dem inversen Feld als Zeichenkette angezeigt. Durch Eingabe einer neuen Zeichenkette kann die Tastenfunktion geändert werden. Dabei wird jede Taste mit ihrer normalen Bedeutung in die Zeichenkette übernommen, d.h. auch CTRL-Zeichen und andere Steuerzeichen. Zum Editieren in dieser invers angezeigten Zeile sind wieder insgesamt drei Tasten zu betätigen:

- zuerst die CTRL-Taste,
- dann die rechte SHIFT-Taste und
- zuletzt eine der folgenden Tasten mit nachfolgender Bedeutung zur Editierung:

Taste	Bedeutung
CRSR links	bewegt den Cursor eine Stelle nach links.
CRSR rechts	bewegt den Cursor eine Stelle nach rechts.
+ (obere Reihe)	fügt eine Leerstelle ein.
- (obere Reihe)	löscht das Zeichen unter dem Cursor.
RETURN	beendet die Definition der Zeichenkette.

Nach dem Beenden der Definition der Zeichenkette mit »CTRL - SHIFT rechts - RETURN« wird die zuerst gedrückte Taste mit einer neuen Bedeutung bzw. Funktion belegt.

4.6.4 Umschalten des ALT-Modus

Der ALT-Modus ermöglicht das Aussenden von 8-Bit-Zeichen entsprechend der Belegung der gedrückten Taste, wobei der ASCII-Code von 80H bis FFH nicht unterdrückt wird. Nach einem CP/M-Kaltstart ist dieser Modus ausgeschaltet. Zum Umschalten des ALT-Modus müssen insgesamt drei Tasten gleichzeitig gedrückt werden:

- zuerst die CTRL-Taste,
- dann die rechte SHIFT-Taste und
- zuletzt die ALT-Taste.

5 Zur Struktur der CP/M3-Befehle

5.1 Einleitung

Das CP/M-Betriebssystem enthält ein Bedienungsprogramm CCP (console command prozessor), das zur Kommunikation des Benutzers mit dem System über die Konsole (Tastatur und Bildschirm) dient. CCP liest und interpretiert die über die Tastatur eingegebenen Befehle.

Eine CP/M-Befehlszeile besteht aus

- einem CP/M-Befehl,
- eventuellen Optionen und
- dem Abschluß mit RETURN (CR = carriage return, ENTER, Wagenrücklauf).

Format: **A>befehl [o]<CR>**

Es bedeuten:

befehl	**CP/M-Befehl im Format einer Dateibezeichnung z.B .d:dateiname.typ;paßwort**
o	**Option**
<CR>	**RETURN-Taste**

Im CP/M-Betriebssystem werden zwei verschiedene Befehlsarten verwendet:

- interne bzw. residente Befehle (built-in commands) und
- externe bzw. transiente Dienstprogramme (transient utility commands).

Ein CP/M-Befehlswort wird zur Identifikation eines Programms verwendet, das sich entweder als Teil von CP/M im Speicher oder als Programmdatei auf einer Diskette befindet. Befehle, die im Speicher abgelegt sind, werden als interne bzw. residente CP/M-Befehle bezeichnet. Befehle, die als Programmdateien auf einer Diskette abgelegt sind, bezeichnet man als transiente Befehle bzw. Dienstprogramme.

CP/M3 besitzt sechs interne Befehle und über zwanzig transiente Dienstprogramme. Zusätzliche Dienstprogramme können jederzeit hinzugefügt werden, die man sich auch selbst schreiben kann.

5.2 Interne bzw. residente Befehle

Interne Befehle sind als Teil von CP/M fest im Speicher abgelegt und jederzeit verfügbar, d.h., die Befehle können jederzeit ausgeführt werden, da dazu keine Diskettendateien notwendig sind. Die Ausführung ist deshalb schneller als bei transienten Dienstprogrammen. Einige interne Befehle besitzen zusätzliche Optionen, die von transienten Dienstprogrammen unterstützt werden. Diese transienten Dienstprogramme haben denselben Namen wie die internen Befehle und sind vom Dateityp COM. Sie werden nur dann geladen, wenn ein interner Befehl eine Option enthält, der nicht vom internen Befehl verarbeitet werden kann.

Wenn man verschiedene Optionen an einen internen Befehl anfügt, kann CP/M dann die Meldung ».COM Requires« (COM-Datei notwendig) ausgeben, wenn das zugehörige transiente Dienstprogramm nicht gefunden werden kann. Um alle Funktionen dieser Befehle zu unterstützen, müssen folgende Dateien auf der Diskette verfügbar sein:

ERASE.COM, RENAME.COM, TYPE.COM und DIR.COM.

Tabelle 5.1 zeigt eine Übersicht über alle internen Befehle, in Kapitel 6 sind die Befehle ausführlich erklärt.

Tabelle 5.1: Übersicht über die internen bzw. residenten CP/M-Befehle

Befehl	Befehlsfunktion
DIR	Anzeigen der Dateinamen aller Dateien des Disketten-Inhaltsverzeichnisses, außer den mit dem Systemattribut SYS gekennzeichneten.
DIRSYS	Anzeigen der Dateinamen aller mit dem Systemattribut SYS gekennzeichneten Dateien des Disketten-Inhaltsverzeichnisses (Abkürzung DIRS).
ERASE	Löschen eines Dateinamens aus dem Disketten-Inhaltsverzeichnis und Freigeben des von der Datei belegten Speicherplatzes auf der Diskette (Abkürzung ERA).
RENAME	Umbenennen einer Diskettendatei (Abkürzung REN).
TYPE	Anzeigen des Inhalts einer ASCII-Text-Datei auf dem Bildschirm (Abkürzung TYP).
USER	Ändern des aktuellen Benutzerbereichs (Abkürzung USE).

5.3 Transiente bzw. externe Dienstprogramme

Nach Eingabe eines als transientes Dienstprogramm identifizierten CP/M-Befehls lädt CP/M die Programmdatei von der Diskette und übergibt alle angefügten Dateinamen, Daten oder Parameter, bevor sie ausgeführt werden.

Tabelle 5.2 zeigt eine Übersicht über alle transienten Dienstprogramme, im Kapitel 6 sind diese CP/M-Befehle näher erklärt. Die transienten Dienstprogramme sind vom Dateityp COM und werden wie die internen Programme nur mit ihrem Dateinamen ohne die Dateikennung COM nach der Bereitschaftsmeldung aufgerufen.

Tabelle 5.2: Übersicht über die transienten CP/M3-Dienstprogramme

Befehl	Funktion
COPYSYS	Erzeugen einer neuen ladefähigen Systemdiskette.
DATE	Setzen und Anzeigen von Datum und Uhrzeit.
DEVICE	Zuordnen der logischen CP/M-Einheiten zu einem oder mehreren physikalischen Geräten, Ändern des Übertragungsprotokolls, der Baud-Raten oder der Konsolen-Bildschirmgröße.
DUMP	Anzeigen einer Datei im ASCII- oder Hexadezimalformat.
ED	Erzeugen und Ändern von Textdateien.
FORMAT	Formatieren von Disketten einschließlich des Ladesektors.
GET	Vorübergehende Eingabe aus einer Diskettendatei statt von der Tastatur.
HELP	Anzeigen von Informationen über die Anwendung der CP/M3-Befehle.
HEXCOM	Erzeugen einer Programmdatei aus einer MAC-Ausgabedatei.

Befehl	Funktion
INITDIR	Initialisieren eines Disketten-Inhaltsverzeichnisses zur Aufnahme von Zeit- und Datumseinträgen.
LIB	Bearbeiten von Bibliotheksdateien mit der Typbezeichnung LIB.
LINK	Verbinden von verschiebbaren REL-Programmodulen (erzeugt mit dem Makroassembler RMAC) zu lauffähigen Programmdateien.
MAC	Übersetzen von Assembler-Quelldateien in Maschinencode-Dateien.
PATCH	Ändern von CP/M-Systemdateien.
PIP	Kopieren und Verbinden von Dateien.
PUT	Vorübergehendes Umleiten einer Drucker- oder Konsolenausgabe in eine Diskettendatei.
RMAC	Übersetzen von Assembler-Quelldateien in verschiebbare Programmodule.
SAVE	Kopieren des Speicherinhalts in eine Diskettendatei.
SET	Setzen von Dateioptionen, Disketten-Labels, Dateiattributen, Zeit- und Datumseinträgen und des Paßwortschutzes.
SETDEF	Setzen der System-Optionen, einschließlich einer Laufwerkssuchverkettung.
SHOW	Anzeigen der Disketten- und Laufwerkseigenschaften.
SID	Überprüfen von Programmen und Korrigieren von Programmfehlern.
SUBMIT	Automatisches Ausführen von mehreren CP/M-Befehlen.
XREF	Erstellen einer Referenzliste von Variablen in Assemblerprogrammen.

5.4 Suchen von Programm- und Datendateien

Wenn eine in einer Befehlszeile eingegebene Datei von CP/M nicht gefunden wird, kann sich die Datei nicht auf der Diskette im Bezugslaufwerk, sondern in einem anderen Laufwerk befinden. Deshalb wird im folgenden der CP/M-Suchvorgang für Programm- und Datendateien näher erläutert.

5.4.1 Suchen von Datendateien

Bei Eingabe einer Befehlszeile übergibt CP/M die Befehlsoptionen bzw. -zusätze an das CP/M-Dienstprogramm bzw. den CP/M-Befehl. Wenn der Befehlszusatz eine Dateibezeichnung enthält, sucht das Dienstprogramm eine Datendatei. Wenn CP/M keine Datendatei finden kann, gibt das Dienstprogramm eine Fehlermeldung auf der Konsole aus, z.B.

`»File not found«`	Datei nicht gefunden oder
`»No File«`	keine Datei,

abhängig vom aufgerufenen CP/M-Dienstprogramm oder -Befehl.

Bei einer Dateibezeichnung ohne Laufwerksangabe im Befehlszusatz durchsucht CP/M das Disketten-Inhaltsverzeichnis des aktuellen Benutzerbereichs auf der Bezugsdiskette. Ist die Datei nicht zu finden, sucht CP/M nach einer Datei mit dem Systemattribut SYS im Disketten-Inhaltsverzeichnis des Benutzerbereichs 0 auf der Diskette im Bezugslaufwerk. Im Benutzerbereich 0 kann das Programm in einer schreibgeschützten Datei (R/O) stehen.

Beispiel

```
3A>TYPE DATEI.TXT
```

Zuerst sucht CP/M im Disketten-Inhaltsverzeichnis des Benutzerbereichs 3 im Laufwerk A. Wenn DATEI.TXT nicht vorhanden ist, wird das Disketten-Inhaltsverzeichnis des Benutzerbereichs 0 im Laufwerk A nach DATEI.TXT mit dem SYS-Attribut durchsucht. Kann die Datei nicht gefunden werden, übergibt CP/M die Kontrolle an TYPE, das wiederum auf der Konsole ausgibt:

```
No File.
```

Einige CP/M-Dienstprogramme, z.B. PIP und DIR, suchen Dateien nur im aktuellen Benutzerbereich. Weil CP/M keinen Schreib-Lese-Zusatz (R/W) bei einer Systemdatei zuläßt, beschränken ERASE und RENAME ihre Suche auf den aktuellen Benutzerbereich.

Der Suchvorgang ist grundsätzlich der gleiche, wenn dem Dateinamen eine Laufwerksangabe hinzugefügt wird. CP/M sucht im Disketten-Inhaltsverzeichnis des aktuellen Benutzerbereichs im angegebenen Laufwerk. Wenn die Datei nicht gefunden wurde, sucht CP/M im Disketten-Inhaltsverzeichnis des Benutzerbereichs 0 im angegebenen Laufwerk nach der Datei mit dem Systemattribut. Kann die Datei hierbei nicht gefunden werden, wird eine Fehlermeldung angezeigt.

5.4.2 Suchen einer Programmdatei

Der Suchvorgang nach einer Programmdatei kann sehr unterschiedlich gegenüber dem nach einer Datendatei sein, da man mit dem SETDEF-Befehl (siehe Abschnitt 6.25) eigene Suchvorgänge definieren kann. Mit SETDEF kann man CP/M veranlassen, bis zu 16 Suchvorgänge durchzuführen, wenn man keine Laufwerksangabe im CP/M-Befehl angibt.

Zuerst soll der Suchvorgang nach einer Programmdatei ohne vorherige Eingabe eines SETDEF-Befehls beschrieben werden.

a) Wenn ein CP/M-Befehl ein transientes Dienstprogramm aufruft, sucht CP/M nach der Programmdatei auf dem aktuellen oder dem angegebenen Laufwerk, zuerst im aktuellen Benutzerbereich und dann im Benutzerbereich 0 nach der gleichen Datei mit dem Systemattribut SYS. Der Vorgang wird sofort abgebrochen, wenn die Datei gefunden wurde. Dann lädt CP/M das Programm in den Speicher und führt es aus. Nach Beendigung des Programms wird wieder die CP/M-Bereitschaftsmeldung zur Eingabe neuer Befehle angezeigt.

b) Wenn CP/M jedoch nicht das Dienstprogramm findet, wird die Befehlszeile mit einem nachfolgenden Fragezeichen nochmals ausgegeben.

c) Wenn man vor dem CP/M-Befehl eine Laufwerksangabe hinzufügt, sucht CP/M die Programmdatei entweder im Disketten-Inhaltsverzeichnis des aktuellen Benutzerbereichs oder dann des Benutzerbereichs 0 auf dem angegebenen Laufwerk, ehe die Befehlszeile mit einem Fragezeichen wiederholt wird, falls die Programmdatei nicht vorhanden ist.

Beispiel

```
4C>A:SHOW SPACE
```

CP/M sucht auf Laufwerk A, im Benutzerbereich 4 und dann im Benutzerbereich 0 nach der Datei SHOW.COM.

d) Bei einem CP/M-Befehl ohne vorangestellte Laufwerksangabe durchsucht CP/M die Disketten-Inhaltsverzeichnisse **aller** Laufwerke in ihrer Reihenfolge, bei einem Laufwerk natürlich nur das aktuelle Laufwerk. Ohne Änderung der Reihenfolge mit einem SETDEF-Befehl, sucht CP/M an **zwei** Stellen nach der Programmdatei.

Beispiel

```
7D>SHOW SPACE
```

CP/M sucht die Datei SHOW.COM auf Laufwerk D erst im Benutzerbereich 7 und dann im Benutzerbereich 0, aber hier mit dem Systemattribut SYS.

Mit dem SET-Befehl kann man die Programmdatei im Benutzerbereich 0 mit dem Systemattribut markieren, um sie dann automatisch für alle Benutzerbereiche verfügbar zu machen. Dadurch muß eine Programmdatei nicht in alle Benutzerbereiche auf allen Laufwerken vervielfältigt werden.

e) Wenn man mit dem SETDEF-Befehl eine eigene Laufwerksreihenfolge bestimmt, sollte das aktuelle und das am meisten verwendete Laufwerk enthalten sein. Die Reihenfolge bleibt bis zu einem neuen Kaltstart des CP/M-Systems erhalten.

Beispiel

Die festgelegte Reihenfolge sei das aktuelle und dann das Laufwerk A.

```
2D>SHOW SPACE
```

CP/M sucht nach SHOW.COM in der Reihenfolge:

- Laufwerk D, Benutzerbereich 2
- Laufwerk D, Benutzerbereich 0
- Laufwerk A, Benutzerbereich 2
- Laufwerk A, Benutzerbereich 0.

5.5 Ausführen von Mehrfachbefehlen

In den bisherigen Beispielen wurde immer nur ein Befehl ausgeführt. CP/M kann aber auch eine Folge von Befehlen ausführen. Die Befehlsfolge kann entweder nach der Bereitschaftsmeldung in einer Befehlszeile eingegeben oder in einer Diskettendatei gespeichert werden. Die in einer Diskettendatei abgespeicherte Befehlsfolge kann mit einem SUBMIT-Befehl abgearbeitet werden.

a) Um mehrere CP/M-Befehle nach der System-Bereitschaftsmeldung einzugeben, wird jeder vollständige CP/M-Befehl mit einem Ausrufezeichen »!« vom nächsten getrennt. Die gesamte Befehlsfolge wird wieder mit RETURN an CP/M übergeben.

Beispiel

```
3A>DIRSYS!DIR TEST*.*!SHOW SPACE<CR>

NON SYSTEM FILE(S) EXIST
3A>DIR TEST*.*
A: TEST1    TXT : TEST2    TXT
3A>SHOW SPACE
A:RW,Space: 100 K
```

b) Wenn man dieselbe Befehlsfolge mehrmals ausführen will, kann diese in einer Diskettendatei gespeichert werden. Mit einem Editierprogramm (z.B. ED) kann man die Datei erstellen. Die Datei muß vom Dateityp SUB sein. Jeder Befehl in der Datei muß in einer neuen Zeile beginnen.

Beispiel

Die Datei: UPDATE.SUB sieht folgendermaßen aus:

```
DIR A:*.COM
ERA B:*.COM
PIP B:=A:*.COM
```

Zum Starten der obigen Datei muß folgender Befehl eingegeben werden:

```
A>SUBMIT UPDATE
```

Das SUBMIT-Dienstprogramm übergibt nacheinander jeden CP/M-Befehl an CP/M zur getrennten Ausführung. Jeder Befehl und alle Programmeldungen werden auf dem Bildschirm angezeigt. Wenn ein Befehl

beendet wurde, erscheinen die CP/M-Bereitschaftsmeldung und der nächste Befehl der SUB-Datei. Ist die SUB-Datei abgearbeitet, wird natürlich nur die CP/M-Bereitschaftsmeldung zur Eingabe neuer Befehle über die Tastatur angezeigt.

c) Mit einem SETDEF-Befehl ist eine automatische Befehlsfolge möglich, da SETDEF im CP/M-Betriebssystem in derselben Weise nach einer SUB-Datei sucht, wie beim Suchen einer Programmdatei nach einer COM-Datei (siehe Abschnitt 5.4.2). Dadurch wird die Anzahl der Suchvorgänge in CP/M verdoppelt, ehe eine Fehlermeldung ausgegeben wird. Der Vorteil einer automatischen Befehlsfolge ist, daß man einen Dateinamen einer SUB-Datei wie einen CP/M-Befehl in der Befehlszeile eingeben kann.

   Wenn CP/M eine SUB-Datei findet, werden die darin enthaltenen Befehle automatisch ausgeführt (siehe auch SETDEF, Abschnitt 6.25 und SUBMIT, Abschnitt 6.28).

5.6 Unterbrechen eines Programmlaufs

Ein Programmlauf kann meistens mit einem Warmstart

CTRL-C

abgebrochen werden, gleichzeitig wird das CP/M-Betriebssystem zurückgesetzt.

Nicht alle CP/M-Anwenderprogramme können mit CTRL-C beendet werden, jedoch die meisten transienten CP/M-Dienstprogramme. Erfolgt gerade eine Bildschirmausgabe, muß vor einer Eingabe von CTRL-C diese mit CTRL-S unterbrochen werden.

Zusätzlich wird mit CTRL-C das Diskettensystem in den Anfangszustand zurückgesetzt. Dieser ganze Vorgang wird mit Warmstart bezeichnet.

Befindet sich der Cursor bei einer Eingabe von CTRL-C direkt hinter der CP/M-Bereitschaftsmeldung, werden alle aktiven Laufwerke von CP/M ausgetragen und anschließend der Bezug zum vorbestimmten Laufwerk (im allgemeinen A) hergestellt. Die aktiven Laufwerke sind alle Laufwerke, auf die nach dem letzten Kalt- oder Warmstart zugegriffen wurde.

Mit dem Befehl »SHOW SPACE« wird die freie Speicherkapazität aller aktiven Laufwerke angezeigt.

Beispiel

Das Beispiel zeigt, daß drei aktive Laufwerke vorhanden sind.

```
A>SHOW SPACE
A: RW,  Space:    160 k
B: RO,  Space:    120 k
C: RO,  Space:    100 k
```

Mit einem Warmstart wird das System zurückgesetzt, so daß nur noch das vorbestimmte Laufwerk A aktiv ist.

```
A>^C
A>SHOW SPACE
A: RW   Space:   160 k
```

5.7 Dienstprogramm HELP

Mit dem transienten Dienstprogramm HELP kann man Informationen über die meisten CP/M-Befehle und -Dienstprogramme, deren Eingaben und deren Bedienung erhalten. Zum Aufrufen genügt die Eingabe:

```
A>HELP<CR>
```

Man kann auch die HELP-Taste und anschließend die RETURN-Taste verwenden.

Auf dem Bildschirm erscheint ein Verzeichnis der verfügbaren Begriffe für Informationen über die CP/M3-Befehle. Nach der anschließend ausgegebenen HELP-Bereitschaftsmeldung »HELP>« kann eine der angezeigten Informationen über einen CP/M-Befehl abgerufen werden. Zuerst wird eine Zusammenfassung des eingegebenen Befehls in englischer Sprache angezeigt, danach eine Liste von Unterbegriffen, die nun getrennt abgerufen werden können, indem nach der HELP-Bereitschaftsmeldung erst ein Punkt und danach der gewünschte Unterbegriff eingegeben werden.

Es kann auch direkt von CP/M aus die Beschreibung eines Begriffs und/oder die Unterbegriffe abgerufen werden.

Beispiele

```
A>HELP
COMMANDS  CNTRCHARS  COPYSYS  DATE      DEVICE  DIR
DUMP      ED         ERASE    FILESPEX  GENCOM  GET
PATCH     PIP(COPY)  PUT      RENAME    RMAC    SAVE
SET       SETDEF     SHOW     ID        SUBMIT  TYPE
USER      XREF
HELP>SHOW
HELP>.OPTIONS
```

Aufruf von HELP mit Ausgabe der verfügbaren Informationen, über den SHOW-Befehl und anschließend über dessen Optionen.

```
A>HELP SHOW            Anzeige der Informationen über den SHOW-Befehl.
A>HELP SHOW OPTIONS    Anzeige der Informationen über die Optionen des SHOW-Befehls.
```

6 Beschreibung der CP/M-Befehle und -Dienstprogramme

In diesem Abschnitt soll eine kurze Beschreibung der auf dem Commodore 128 implementierten CP/M3-Befehle gegeben werden. Kurze Beispiele dienen zur Erläuterung.

- In rechteckigen Klammern »[ ]« können aus einer Liste frei wählbare Optionen als Argument bzw. Parameter nach dem CP/M-Befehl angefügt werden.

- Weiterhin werden folgende Abkürzungen verwendet:

d	für ein Laufwerk (A, B, C, D oder E beim C 128)
n	für eine Zahl
o	für eine Option oder eine Liste von Optionen
s	für eine Zeichenkette (string)
RO	Schreibschutz (read only = nur lesen)
RW	ohne Schreibschutz (read-write = lesen-schreiben)
DIR	DIR-Attribut (directory = Disketten-Inhaltsverzeichnis)
SYS	SYS-Attribut (system = System)

- Alle in Großbuchstaben geschriebenen Formatangaben in einem CP/M-Befehl sind obligatorisch.

- Alle in Kleinbuchstaben geschriebenen Formatangaben in einem CP/M-Befehl sind Platzhalter für eine Option, eine Zahl oder eine Zeichenkette, entweder aus einem vorgegebenen Wertebereich oder einer vorgegebenen Liste oder frei wählbar.

- Jeder Befehl muß durch Drücken der RETURN-Taste logisch abgeschlossen und an das Betriebssystem übergeben werden (carriage return <CR> = Wagenrücklauf).

6.1 DATEINAME (Laden und Ausführen eines Programms)

Format: **nd:dateiname <CR>**

Es bedeuten:

n	ein Benutzerbereich
d	ein Diskettenlaufwerk
dateiname	der Name der Datei »dateiname.COM«, die das zu ladende und auszuführende Programm, z.B. ein transientes CP/M-Dienstprogramm, enthält

Beschreibung

Direkt ausführbare CP/M-Programme werden in Dateien mit der Bezeichnung »dateiname.COM« gespeichert. Nach Eingabe des Dateinamens und Abschluß mit <CR> läuft unter CP/M folgendes ab:

- CP/M sucht auf der Diskette »d«, d.h. auf der im Laufwerk »d« eingelegten Diskette, die Programmdatei »dateiname.COM« im Benutzerbereich »n«.
- Diese Programmdatei wird in den Arbeitsspeicher TPA geladen.
- Mit der Ausführung der Programminstruktionen wird gestartet.

Ist diese Datei auf der bezeichneten Diskette nicht vorhanden, gibt CP/M die Meldung

```
dateiname?
```

aus. Das Programm liegt dann nicht im bezeichneten Benutzerbereich als COM-Datei vor.

6.2 nd: (Wechsel des Bezugslaufwerkes und des Benutzerbereiches)

Format: **nd:**

Es bedeuten:

n	ein Benutzerbereich (0 bis 15)
d	ein Diskettenlaufwerk (Großbuchstabe A, B, C, D oder E)

Beschreibung

Unter CP/M stehen Sie immer mit einem Diskettenlaufwerk in Verbindung (Bezugslaufwerk). Sie können an der Bereitschaftsmeldung erkennen, mit welchem Laufwerk und in welchem Benutzerbereich Sie gerade verbunden sind, bei »A>« mit dem Laufwerk A (Benutzerbereich 0), bei »1B>« mit dem Laufwerk B (Benutzerbereich 1) usw.

Mit dem Commodore 128 sind unter CP/M3 nur vier Laufwerke (A, B, C, D) anschließbar, zusätzlich ist ein virtuelles Laufwerk E vorhanden. Man kann nun das Bezugslaufwerk bzw. den Benutzerbereich durch Eingabe der neuen Bezeichnungen ändern.

Arbeitet man nur mit einem Einzellaufwerk (1541, 1570 bzw. 1571), so ist es sehr praktisch, daß ein zweites Laufwerk E simuliert werden kann. Bei Eingabe des Laufwerkes E wird man zum Wechseln der Diskette aufgefordert (neue Bezugsdiskette); d.h., das neue Bezugslaufwerk ist nur virtuell. Im folgenden wird deshalb auch der Ausdruck »Bezugsdiskette« verwendet. Die Disketten selbst werden nicht mit der Bezugsdisketten-Angabe markiert.

Beispiel

Bei einem Einzel-Diskettenlaufwerk wollen Sie von der Bezugsdiskette A auf die Bezugsdiskette E wechseln. Dann geben Sie nach der Bereitschaftsmeldung ein:

```
A>  E: <CR>
```

In der letzten Bildschirmzeile erscheint die Meldung

```
Insert Disk E in Drive A
```

Nach dem Einlegen einer neuen Diskette und Drücken der RETURN-Taste wird Bezug zu dieser Diskette hergestellt, d.h., CP/M meldet sich mit

```
E>
```

6.3 COPYSYS

Format: **COPYSYS**

Beschreibung

Der COPYSYS-Befehl kopiert das Betriebssystem CP/M3 von einer CP/M3-Systemdiskette auf eine andere Diskette. Die neue Diskette muß im selben Format wie die Originaldiskette formatiert sein.

Dieses COPYSYS-Dienstprogramm kopiert nur die Systemspuren auf die neue Diskette. Um die neue Diskette als CP/M3-Systemdiskette zu verwenden, muß also danach noch die Systemdatei CPM+.SYS auf die neue Diskette kopiert werden, wozu COPYSYS eine Option hat. Zum Kopieren anderer Dateien muß der PIP-Befehl verwendet werden.

Beispiel

```
A> COPYSYS

Copysys Ver 3.0
Source drive name (or return for default)  <CR>

Source on A then type return
```

COPYSYS fragt nach dem Quellenlaufwerk; mit <CR> kann das Bezugslaufwerk gewählt werden.

6.4 DATE

Format: **DATE**
DATE CONTINUOUS
DATE mm/dd/yy hh:mm:ss
DATE SET

Beschreibung

Der DATE-Befehl ist ein transientes Dienstprogramm, mit dem Datum und Tageszeit angezeigt und eingestellt werden können. Bei einem CP/M-Kaltstart werden Datum und Uhrzeit auf das Erstellungsdatum des CP/M3-Systems eingestellt. DATE verwendet man nun zum Ändern dieser Initialwerte in das aktuelle Datum und die aktuelle Uhrzeit.

6.4.1 Anzeigen des aktuellen Datums und der aktuellen Uhrzeit

Format: **DATE**
DATE CONTINUOUS

Beschreibung

Diese Form des DATE-Befehls zeigt das aktuelle Datum und die aktuelle Uhrzeit an. Mit »CONTINUOUS« können fortlaufend Datum und Uhrzeit angezeigt werden; es kann mit »C« abgekürzt werden. Durch Drücken irgendeiner Taste kann die Anzeige unterbrochen werden.

Beispiele

`A> DATE`	Anzeige des aktuellen Datums und der aktuellen Uhrzeit.
`A> DATE C`	Fortlaufende Anzeige von Datum und Uhrzeit.

6.4.2 Einstellen von Datum und Uhrzeit

Format: **DATE mm/dd/yy hh:mm:ss**
DATE SET

Es bedeuten fortlaufend:

mm	Monat mit Januar (=1), Februar (=2) ....Dezember (=12),
dd	Tag (1 bis 31),
yy	Jahr (relativ zu 1900),
hh	Stunde (0 bis 23),
mm	Minute (0 bis 59) und
ss	Sekunde (0 bis 59).

Beschreibung

Mit der ersten Form können Datum und Uhrzeit im Befehl direkt eingegeben werden. Die Gültigkeit der Eingaben wird geprüft.

Mit der zweiten Form werden Datum und Uhrzeit nacheinander abgefragt. Das Überspringen der Datums- und Uhrzeit-Eingabe geschieht mit einer direkten Eingabe von <CR> nach dem Fragetext.

Beispiele

```
A> DATE 09/15/85 12:30:0
Press any key to set time
Sun 09/15/85 12:30:00
```

Eingabe von Datum und Uhrzeit,
Einstellen der Zeit durch Drücken irgendeiner Taste.

```
A> DATE SET
Enter today's date - (MM/DD/YY):
Enter the time - (HH:MM:SS):
Press any key to set time
```

Eingabe des Datums,
Eingabe der Uhrzeit,
Einstellen der Zeit durch Drücken irgendeiner Taste.

6.5 DEVICE

Format:

DEVICE o
DEVICE ld = pd [o]
DEVICE ld = NULL
DEVICE pd [o]
DEVICE CONSOLE [o]

Es bedeuten:

o	Option
ld	logische Einheit (logical device), ein symbolischer Gerätename für eine Gruppe von Einheiten, die alle vom System bedient werden können
pd	physikalische Einheit (physical device), ein aktueller Gerätename einer vom System ansprechbaren Geräteeinheit

Beschreibung

Der DEVICE-Befehl ist ein transientes Dienstprogramm, das die aktuelle Zuweisung der logischen Einheit und Namen der physikalischen Einheiten anzeigt. Außerdem können mit DEVICE logische CP/M-Einheiten den Peripheriegeräten des Rechners zugeordnet werden. Mit DEVICE können auch das Übertragungsprotokoll und die Geschwindigkeit der Peripheriegeräte festgelegt und die aktuelle Bildschirmgröße der Konsole angezeigt und festgelegt werden.

CP/M3 unterstützt die folgenden fünf logischen Einheiten:

CONIN: (Konsoleneingabe)
CONOUT: (Konsolenausgabe)
AUXIN: (Hilfseingabe)
AUXOUT: (Hilfsausgabe)
LST: (Druckerausgabe)

oder auch

CON: (für CONIN: und CONOUT:)

Die Namen der physikalischen Einheiten sind bei verschiedenen Rechnern jeweils andere. Mit dem DEVICE-Befehl können die Namen und Eigenschaften der physikalischen Einheiten des verwendeten Systems angezeigt werden.

6.5.1 Anzeige der Eigenschaften und Zuordnungen der Einheiten

Format: **DEVICE o**

mit o für NAMES oder VALUES oder pd oder ld

Beschreibung

Diese Befehlsform dient zur Anzeige der Namen und Eigenschaften der physikalischen Einheiten und der derzeitigen Zuordnung der logischen Einheiten im System.

Beispiele

```
A> DEVICE
```

Anzeige der physikalischen Einheiten und der aktuellen Zuordnung der logischen Einheiten.

```
Physical Devices:
I=Input, O=Output, S=Serial, X=Xon-Xoff
KEYS NONE I 80COL NONE O 40COL NONE O
PRT1 NONE O PRT2  NONE O 6551.19200 IOSX

Current Assignments:
CONIN:          = KEYS
CONOUT:         = 80COL
AUXIN:          = NullDevice
AUXOUT:         = NullDevice
LST:            = PRT1

Enter new assignment or hit RETURN
```

Nun können neue zugelassene Zuweisungen von Einheiten eingegeben werden oder mit <CR> der angezeigte bisherige Zustand übernommen werden.

```
A> DEVICE NAMES
```

Ausgabe der physikalischen Einheiten (siehe oben)

```
A> DEVICE VALUES
```

Ausgabe der aktuellen Zuordnung der logischen Einheiten (siehe oben)

```
A> DEVICE CON
```

Ausgabe der Zuordnung der logischen Einheit Konsole.

```
CONIN: = KEYS
CONOUT: = 80COL
```

6.5.2 Zuweisen einer logischen Einheit

Format: **DEVICE ld = pd [o]**
DEVICE ld = NULL

Beschreibung

Mit der ersten Befehlsform wird einer logischen Einheit eine oder mehrere physikalische Einheiten zugewiesen; mit der zweiten werden die logischen Einheiten von allen physikalischen Einheiten getrennt.

In Tabelle 6.1 sind die möglichen Optionen zusammengestellt.

Tabelle 6.1: DEVICE-Optionen für physikalische Einheiten

Option	Bedeutung
XON	legt ein XON/XOFF-Übertragungsprotokoll fest. Dieses Protokoll verwendet die beiden Sonderzeichen XON und XOFF des ASCII-Zeichensatzes. XON schaltet die Übertragung ein und XOFF aus. Vor der Übertragung eines Zeichens vom Rechner zu einem Peripheriegerät prüft der Rechner, ob Daten von den Peripheriegeräten kommen. Bei einem ankommenden XOFF-Zeichen unterbricht der Rechner jede weitere Ausgabe, bis ein XON die Bereitschaft der Einheit zum Empfang weiterer Daten anzeigt.
NOXON	bedeutet, daß kein XON/XOFF-Protokoll verwendet wird, d.h. der Rechner sendet unabhängig davon, ob das Empfangsgerät bereit ist oder nicht, Daten an das Gerät.
Baudrate	Die Baudrate legt die Übertragungsgeschwindigkeit des Geräts fest. CP/M verwendet folgende Baudraten: 50 75 110 134 150 300 600 1200 1800 2400 3600 4800 7200 9600 19200

Beispiele

`A> DEVICE CONOUT:= 80COL,PRT1`	Zuweisen des Konsolenausgangs CONOUT: dem 80-Zeichen-Bildschirm und dem Drucker PRT1.
`A> DEVICE AUXIN:= 6551 [XON,300]`	Zuweisen des logischen Hilfseingangs AUXIN: der physikalischen Einheit 6551 (serielle Schnittstelle im C 128) mit XON/XOFF-Protokoll und einer Übertragungsrate von 300 Baud.
`A> DEVICE LST:= NULL`	Unterbrechen der List-Ausgabe der logischen Einheit LST.

6.5.3 Einstellen der Eigenschaften einer physikalischen Einheit

Format: **DEVICE pd [o]**

Beschreibung

Diese DEVICE-Befehlsart stellt die Eigenschaften der im Befehl angegebenen physikalischen Einheit ein.

Beispiel

`A> DEVICE 6551 [XON,300]`	Einschalten des XON/XOFF-Protokolls für die physikalische Einheit 6551 und Einstellen einer Übertragungsrate von 300 Baud.

6.5.4 Anzeigen und Einstellen der Konsolen-Bildschirmgröße

Format: **DEVICE CONSOLE [o]**
DEVICE CON: [o]

Beschreibung

Mit dieser DEVICE-Befehlsform kann die aktuelle Bildschirmgröße der Konsole angezeigt und eingestellt werden.

Beispiele

`A> DEVICE CON: [PAGE]`	Anzeigen der Breite in Spalten und der Länge in Zeilen des Bildschirms.
`A> DEVICE CON: [COLUMNS=40, LINES=16]`	Einstellen der Bildschirmgröße auf 40 Spalten und 16 Zeilen.

6.6 DIR

Format: **DIR d:**
DIR datei
DIRSYS d:
DIRSYS datei
DIR d: [o]
DIR datei [o]

Es bedeuten:

d	ein Diskettenlaufwerk (A bis E)
datei	eine Dateibezeichnung, z.B. »d:dateiname.typ;paßwort«

Beschreibung

Mit dem DIR-Befehl kann das Disketten-Inhaltsverzeichnis (directory) angezeigt werden, d.h. die Dateinamen der bezeichneten Diskette und ggf. deren Eigenschaften.

DIR und DIRSYS sind residente Befehle, und DIR mit Optionen ist ein transientes Dienstprogramm.

6.6.1 Anzeigen des Disketten-Inhaltsverzeichnisses

Format: **DIR d:**
DIR datei
DIRSYS d:
DIRSYS datei

Beschreibung

Die residenten Befehle DIR und DIRSYS bewirken die Anzeige und die Ausgabe der im Disketten-Inhaltsverzeichnis der Bezugsdiskette eingetragenen Dateinamen.

Der DIR-Befehl gibt nur die Dateinamen im aktuellen Benutzerbereich aus, die das DIR-Attribut besitzen, d.h., Dateien mit dem Systemattribut SYS werden nicht angezeigt. Der DIRSYS-Befehl zeigt nur die Dateinamen mit dem Systemattribut SYS im aktuellen Benutzerbereich an. DIR und DIRSYS akzeptieren die Platzhalterzeichen »*« und »?« in der Dateibezeichnung. DIRSYS kann mit DIRS abgekürzt werden.

Ohne eine Laufwerks- und Dateibezeichnung, werden alle Dateinamen im Bezugslaufwerk und im aktuellen Benutzerbereich angezeigt, mit DIR jedoch nur die mit dem DIR-Attribut und mit DIRSYS nur alle Systemdateien mit dem Systemattribut SYS.

Mit einer Laufwerksangabe, aber ohne Dateibezeichnung, zeigt der DIR-Befehl alle Dateinamen mit dem DIR-Attribut im aktuellen Benutzerbereich und im angegebenen Laufwerk an, DIRSYS entsprechend die Systemdateien.

Erfüllt kein Dateiname die angegebenen Merkmale bzw. ist kein Dateiname im Disketten-Inhaltsverzeichnis der angesprochenen Diskette eingetragen, wird die Meldung

```
No File
```

ausgegeben.

Wenn eine Systemdatei die angegebenen Merkmale erfüllt, erhält man beim DIR-Befehl die Meldung

```
SYSTEM FILE(S) EXIST
```

Wenn eine Datei mit dem DIR-Attribut die angegebenen Merkmale erfüllt, erhält man bei DIRSYS die Meldung

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

Ist der Bildschirm bei einer DIR-Angabe vollgeschrieben, unterbricht DIR die Ausgabe. Sie kann durch Drücken einer beliebigen Taste fortgesetzt werden.

Beispiele

`A> DIR`	Anzeigen aller Dateien des Benutzerbereichs 0 im Laufwerk A mit dem DIR-Attribut.
`A> DIR B:`	Anzeigen aller Dateien des Benutzerbereichs 0 im Laufwerk B mit dem DIR-Attribut.
`A> DIR B:X.BAS`	Anzeigen der Datei X.BAS auf der Diskette im Laufwerk B.

`4A> DIR *.BAS`	Anzeigen aller Dateien mit dem Dateityp BAS und dem Attribut DIR im Benutzerbereich 4 der Bezugsdiskette A.
`B> DIR A:X*.C?D`	Anzeigen aller Dateien mit dem DIR-Attribut im Benutzerbereich 0 auf der Diskette im Laufwerk A, deren Dateinamen mit dem Buchstaben X beginnen und deren 3stelliger Dateitypname als erstes Zeichen C und als letztes Zeichen D enthält.
`A> DIRS`	Anzeigen aller Dateien mit dem Systemattribut SYS im Benutzerbereich 0 auf der Bezugsdiskette A.
`3A> DIRS *.COM`	Anzeigen aller Systemdateien des Typs COM auf der Bezugsdiskette A im Benutzerbereich 3.

6.6.2 Externer DIR-Befehl mit Optionen

Format: **DIR d: [o]**
DIR datei [o]

Beschreibung

Dieser externe DIR-Befehl mit Optionen ist eine erweiterte Version des residenten DIR-Befehls und zeigt die CP/M-Dateien auf verschiedene Weisen an. DIR kann Dateien auf einem oder allen Laufwerken in einem oder allen Benutzerbereichen suchen. Die in rechteckigen Klammern an den DIR-Befehl angehängten Optionen modifizieren die Ausführung und sind in der Tabelle 6.2 zusammengefaßt. Es kann nur eine Option oder mehrere durch Kommas oder Leerstellen getrennte Optionen verwendet werden. Auch hier wird die Ausgabe bei vollem Bildschirm unterbrochen und kann durch Drücken einer beliebigen Taste fortgesetzt werden.

Tabelle 6.2: DIR-Optionen

Option	Funktion
ATT	Anzeigen der Dateiattribute.
DATE	Anzeigen der Dateien mit ihren Datums- und Zeiteinträgen. Ohne aktive Einträge gibt DIR die Meldung aus: `Date and Time Stamping Inactive`
DIR	Anzeigen nur der Dateien mit einem DIR-Attribut.
DRIVE=ALL	Anzeigen der Dateien aller angeschlossenen Laufwerke. Statt DRIVE kann auch DISK eingegeben werden, auch bei den beiden nächsten Laufwerksoptionen.
DRIVE=(A,..,D)	Anzeigen der Dateien der angegebenen Laufwerke.
DRIVE=d	Anzeigen der Dateien im Laufwerk d.
EXCLUDE	Anzeigen der Dateien im voreingestellten Laufwerk (A) und Benutzerbereich (0), die nicht in der Befehlszeile als Datei angesprochen werden.
FF	Senden eines Anfangszeichens »Seitenvorschub« (form feed) an die Druckereinheit, wenn sie mit CTRL-P aktiviert worden ist. Bei einer Option »LENGTH=n« erfolgt alle n Zeilen ein Seitenvorschub, sonst wird die voreingestellte seitenweise Ausgabe abgeschaltet.
FULL	Anzeigen des Namens, der Größe, der Anzahl der 128-Byte-Records und der Attribute der Dateien. Bei einem vorhandenen DIR-Attribut auf der Diskette zeigt DIR den Paßwortschutz und die Zeiteinträge an, sonst entfallen diese Einträge. Die Dateinamen werden alphabetisch sortiert. FULL ist das vorgegebene Ausgabeformat bei dem DIR-Befehl mit Optionen. Im SET-Befehl findet man eine Beschreibung über die Dateiattribute und den Paßwortschutz.

Option	Funktion
LENGTH=n	Anzeigen von n Zeilen pro Seite. Die Kopfzeile wird auf jeder Seite wiederholt und bei n nicht mitgezählt. n ist eine Zahl zwischen 5 und 65536. Voreingestellt ist die Bildschirmlänge.
MESSAGE	Anzeigen der Laufwerke und Benutzerbereiche, die gerade durchsucht werden. Sind keine Dateien vorhanden, wird angezeigt »File not found«.
NOPAGE	Ausschalten der seitenweise Ausgabe. Die Information scrollt kontinuierlich über den Bildschirm.
NOSORT	Anzeigen der Dateien in der Reihenfolge, in der sie auf der Diskette stehen.
RO	Anzeigen nur der Dateien mit Schreibschutz-Attribut (read only).
RW	Anzeigen nur der Dateien ohne Schreibschutz-Attribut (read-write).
SIZE	Anzeigen der Dateien und der Dateigröße in Kbytes.
SYS	Anzeigen nur der Dateien mit dem Systemattribut SYS.
USER=ALL	Anzeigen aller Dateien aller Benutzerbereiche im Bezugslaufwerk.
USER=n	Anzeigen der Dateien des Benutzerbereichs n.
USER=(0,1,..,15)	Anzeigen der Dateien der bezeichneten Benutzerbereiche.

Beispiele

`A> DIR C: [FULL]`	Anzeigen aller Eigenschaften aller Dateien des Benutzerbereichs 0 von Laufwerk C.
`A> DIR C: [SIZE]`	Anzeigen der Größe aller Dateien des Benutzerbereichs 0 von Laufwerk C.
`A> DIR [DRIVE=C,FF]`	Anzeigen der Dateien von Laufwerk C; vor der Anzeige wird ein Seitenvorschub-Zeichen an den Drucker gesendet.
`A> DIR D: [RW,SYS]`	Anzeigen aller Dateien des Laufwerkes D ohne Schreibschutz, aber mit dem Systemattribut.
`A> DIR C: [USER=ALL]`	Anzeigen aller Dateien aus allen Benutzerbereichen von Laufwerk C.
`A> DIR [USER=2]`	Anzeigen aller Dateien des Benutzerbereichs 2 von Bezugslaufwerk A.
`A> DIR [USER=(3,5)]`	Anzeigen aller Dateien der Benutzerbereiche 3 und 5 auf der Bezugsdiskette.
`A> DIR [DRIVE=ALL]`	Anzeigen aller Dateien des Benutzerbereichs 0 von allen Laufwerken in der Suchkette (siehe auch den SETDEF-Befehl).
`4A> DIR [DRIVE=C]`	Anzeigen aller Dateien des Benutzerbereichs 4 auf dem Laufwerk C.
`A> DIR [DRIVE=(B,D)]`	Anzeigen aller Dateien des Benutzerbereichs 0 auf den Laufwerken B und D.
`A> DIR [EXCLUDE] *.COM`	Ausgabe aller Dateien der Bezugsdiskette im Benutzerbereich 0, die nicht den Dateityp COM besitzen.
`A> DIR [USER=ALL, DRIVE=ALL,SYS] *.COM *.ASM`	Anzeigen aller Systemdateien mit den Dateitypen COM und ASM von der Bezugsdiskette aus allen Benutzerbereichen.
`A> DIR X.SUB [MESSAGE,USER=ALL, DRIVE=ALL]`	Suchen der Datei X.SUB auf allen Laufwerken und in allen Benutzerbereichen unter Anzeige der durchsuchten Laufwerke und Benutzerbereiche.
`A> DIR [DRIVE=ALL USER=ALL] TEST.BAS`	Anzeigen der Datei TEST.BAS, wenn sie in einem angeschlossenen Laufwerk in einem Benutzerbereich zu finden ist.
`A> DIR [SIZE,RW] D:`	Anzeigen aller Dateien ohne Schreibschutz im Laufwerk D mit ihren Dateigrößen in Kbytes. D: hat dieselbe Wirkung wie D:*.*.

6.7 DUMP

Format: **DUMP datei**

Es bedeutet:

datei	eine Dateibezeichnung, z.B. »d:dateiname.typ;paßwort«

Beschreibung

Mit dem Befehl DUMP wird der Inhalt der bezeichneten Datei im Hexadezimal- und ASCII-Format auf dem Bildschirm ausgegeben. Die Platzhalterzeichen »*« und »?« in der Dateibezeichnung sind nicht erlaubt.

Beispiele

`A> DUMP DATA.DTA`	Anzeigen des Inhalts der Datei DATA.DTA vom Bezugslaufwerk auf dem Bildschirm.
`A> DUMP B:TEXT.TXT`	Anzeigen des Inhalts der Textdatei TEXT.TXT von der Diskette im Laufwerk B auf dem Bildschirm.

6.8 ED

Format: **ED quelldatei zieldatei**

Es bedeuten:

quelldatei	die Dateibezeichnung der zu editierenden Quelldatei im Format »d:dateiname.typ;paßwort«.
zieldatei	die Dateibezeichnung der editierten Zieldatei im Format »d:dateiname.typ;paßwort«, wenn man den Namen vom Namen der ursprünglichen Quelldatei unterscheiden möchte.

Beschreibung

Das transiente ED-Dienstprogramm verwendet man zum Erstellen und Editieren von CP/M-Quellprogrammen, Dateien und Texten als Diskettendateien. Es ist ein zeilenorientiertes Texteditierprogramm (Texteditor), d.h. man erstellt und ändert Zeichendateien zeilenweise oder durch Ansprechen einzelner Zeichen innerhalb einer Zeile.

Das ED-Dienstprogramm benutzt einen Teil des Benutzerspeichers als aktiven Textpuffer, in dem die in einer Datei enthaltenen Zeichen gelöscht, geändert oder neue Zeichen eingefügt werden können. Zum Einlesen aller oder eines Teils der Daten einer Datei in den Puffer wird das A-Kommando verwendet. Das W- oder E-Kommando dient zum Zurückschreiben aller oder eines Teils der Zeichen vom Puffer in die Diskettendatei.

Ein imaginärer Zeichenzeiger CP (character pointer) steht entweder am Pufferanfang, zwischen zwei Zeichen innerhalb des Puffers oder am Pufferende.

Der Dialog zwischen dem Benutzer und ED läuft entweder im Kommando- oder Einfügemodus ab. Im Kommandomodus zeigt ED einen Stern »*« als Bereitschaftszeichen auf dem Bildschirm an. In diesem Modus können Kommandos in Form von Einzelbuchstaben eingegeben werden, z.B. zum Lesen von Text in den Puffer, zum Bewegen des Zeichenzeigers oder zum Ändern des ED-Modus. Im Kommandomodus sind die in Tabelle 6.3 aufgeführten Zeileneditierzeichen verwendbar; im Einfügemodus sind nicht alle diese Editierzeichen zulässig. In Tabelle 6.4 sind alle ED-Kommandos zusammengestellt.

Falls die mit ED aufgerufene Datei existiert, wird diese Datei zur Bearbeitung vorbereitet und eine Zwischendatei mit dem Namen

»quelldatei.$$$« eröffnet. Falls die zu editierende Datei nicht existiert, wird eine neue Datei mit dem eingegebenen Namen »quelldatei.typ« und eine Zwischendatei »quelldatei.$$$« eröffnet. Nach Abschluß des Editiervorgangs wird die Originaldatei in »quelldatei.BAK« und die editierte Datei mit dem Namen »quelldatei.typ« auf die Diskette »d« oder eventuell als »zieldatei.typ2« auf die Diskette »d2« geschrieben. Wenn man das Rückschreiben vergißt, geht die editierte Datei verloren, d.h., jeder Arbeitsschritt muß ED mitgeteilt werden.

Wenn man den ED-Befehl ohne Argumente eingibt, erscheinen nacheinander die Aufforderungen zur Eingabe einer Quelldatei und einer Zieldatei. Für die Zieldatei kann ein Dateiname oder Laufwerk angegeben werden, wenn diese von der Quelldatei abweichen sollen. Betätigt man direkt nach der Eingabeaufforderung für die Zieldatei die RETURN-Taste, dann wird die Originaldatei durch die editierte Ausgabedatei ersetzt und die Originaldatei mit dem Dateityp BAK umbenannt. Wenn die zweite Dateibezeichnung lediglich die Laufwerksangabe »d2« enthält, sind der Dateiname und -typ dieselben wie bei der Originaldatei.

Wenn die in der ersten Dateibezeichnung genannte Datei »d: quelldatei.typ« bereits existiert, muß die Datei mit dem A-Kommando in den Puffer gelesen werden.

Mit dem I(insert)-Kommando wird ED in den Einfügemodus gesetzt. In diesem Modus werden alle eingegebenen Zeichen nacheinander im Puffer gespeichert, beginnend bei der Position des CP. Alle im Einfügemodus getippten Einbuchstaben-Kommandos werden nicht als Kommandos ausgelegt, sondern einfach im Puffer gespeichert. Mit CTRL-Z kann man vom Einfügemodus in den Kommandomodus zurückkehren.

Die Einbuchstaben-Kommandos werden gewöhnlich klein geschrieben. Kommandos, denen eine Zeichenfolge folgt, enden mit CTRL-Z, wenn ein weiterer Kommandobuchstabe folgen soll.

Jedes als Großbuchstabe eingegebene Einbuchstaben-Kommando veranlaßt ED, bis zu dem CTRL-Z, das das Kommando abschließt, alle Zeichen intern in Großbuchstaben zu übertragen.

Bei der Aktivierung von ED nehmen die links auf dem Bildschirm erscheinenden Zeilennummern die Form:

```
nnnnn:
```

an, wobei nnnnn eine Zahl von 1 bis 65535 ist. Die Zeilennummern werden nur zur Orientierung angezeigt und sind weder im Puffer noch in der Datei enthalten. Die Bildschirmzeile beginnt mit »:*«, wenn der Zeichenzeiger am Pufferanfang oder -ende steht.

Beispiele

`A> ED PGM.ASM`
`:*`
`:*#A`
`:*OP`
`:*E`
	Einrichten der Datei PGM.ASM zum Editieren,
	ED-Bereitschaftsmeldung zur Kommandoeingabe,
	Einlesen der Datei PGM.ASM in den Puffer,
	Füllen des Bildschirms mit den ersten Zeilen des Puffers,
	Beenden des Editiervorgangs: die Originaldatei wird als PGM.BAK und die editierte Datei als PGM.ASM im* Laufwerk A gespeichert.

`A> ED PGM.PAS B:NEU.PAS`	Die Originaldatei PGM.PAS im Laufwerk A bleibt dort unverändert, während die editierte Datei NEU.PAS auf dem Laufwerk B gespeichert wird.
`A> B:ED PGM.PAS B:`	Die Datei ED.COM befindet sich im Laufwerk B. Die Originaldatei PGM.PAS im Laufwerk A bleibt unverändert, während die editierte Datei auch als PGM.PAS auf dem Laufwerk B gespeichert wird.

Tabelle 6.3: ED-Zeileneditierzeichen

Befehl	Funktion
CTRL-C	Rücksprung ins CP/M-Betriebssystem (Warmstart, nur im Kommandomodus).
CTRL-E	Bewegt den Zeichenzeiger CP an den Anfang der nächsten Zeile, ohne die bisherigen Eingaben zu löschen (nur im Kommandomodus).
CTRL-H	Löscht das zuletzt in der Zeile eingegebene Zeichen (im Kommando- und Einfügemodus).
CTRL-U	Löscht die gesamte gerade eingegebene Zeile (im Kommando- und Einfügemodus).
CTRL-X	wie CTRL-U
CTRL-Z	Springt aus dem Einfügemodus zurück in den Kommandomodus.
DEL	wie CTRL-H

Tabelle 6.4: ED-Kommandos

Kommando	Funktion
nA	Übernahme von n Zeilen aus der Originaldatei in den Pufferspeicher.
0A	Übernahme so vieler Zeilen aus der Datei, bis der Puffer zur Hälfte gefüllt ist.
#A	Übernahme aus der Datei, bis der Puffer voll oder das Dateiende erreicht ist.
B,-B	Verschieben des Zeichenzeigers an den Anfang (+B oder B) oder an das Ende des Pufferspeichers (-B).
nC, -nC	Verschieben des Zeichenzeigers um n Zeichen vorwärts (+nC oder nC) oder rückwärts (-nC).
nD, -nD	Löschen von n Zeichen vor (+nD oder nD) oder hinter (-nD) dem Zeichenzeiger.
E	Sichern der neuen Datei und Rücksprung ins CP/M.
nFs^Z	Suchen des n-ten Auftretens der Zeichenkette »s«. Ohne eine Angabe für n wird die erste Zeichenkette »s« gesucht. CTRL-Z (^Z) dient als Abschluß der Zeichenkette, wenn noch ein weiteres ED-Kommando in der Zeile folgen soll. Dieses Kommando »F« wandelt alle Zeichen in den Großbuchstabenmodus um. Zum Suchen im Kleinschriftmodus dient das Kommando »f«.
H	Beenden der Editierung, Speichern der neu editierten Datei, Fortsetzen der Editierung mit der neuen Datei als neuer Quelldatei.
I	Aufrufen des Einfügemodus zur Texteingabe im Großschriftmodus nach der Position des Zeichenzeigers CP. Mit dem Kommando »i« kann Text im Kleinschriftmodus eingegeben werden. Der Einfügemodus wird mit CTRL-Z beendet, der Zeichenzeiger wird dabei ans Ende des eingefügten Textes verschoben.

Kommando	Funktion
Is^Z	Einfügen der Zeichenkette »s« im Großschriftmodus, beginnend an der Position des Zeichenzeigers. CTRL-Z kann als logischer Zeichenkettenabschluß dienen. Der Zeichenzeiger wird an das Ende der eingefügten Zeichenkette verschoben.
i	Aufrufen des Einfügemodus im Kleinschriftmodus, sonst wie »I«.
is^Z	Einfügen der Zeichenkette »s« im Kleinschriftmodus, sonst wie »Is^Z«.
Js1^Zs2 ^Zs3^Z	Aneinanderfügen von Zeichenketten, indem die erste Zeichenkette »s1« gesucht, die zweite Zeichenkette »s2« an die erste angefügt wird und alle Zeichen bis zur dritten Zeichenkette »s3« gelöscht werden. Der logische Abschluß nach »s3« mit CTRL-Z muß nur dann erfolgen, wenn ein weiteres ED-Kommando in derselben Zeile eingegeben werden soll.
nk, -nk	Löschen von n Zeilen nach (nk oder +nk) bzw. vor (-nk) dem Zeichenzeiger.
nL, -nL, 0L	Verschieben des Zeichenzeigers um n Zeilen vorwärts (nL oder +nL) bzw. rückwärts (-nL). Mit »0L« verschiebt sich der Zeichenzeiger an den Anfang der aktuellen Zeile.
nMk^Z	n-maliges Ausführen der ED-Kommandos »k«. Bei »0M« oder »M« werden die Kommandos »k« so lange wiederholt, bis das Textende im Puffer erreicht ist oder ein Fehler auftritt. Den logischen Kommandoabschluß CTRL-Z benötigt man nur dann, wenn ein weiteres ED-Kommando in derselben Zeile eingegeben werden soll, das nicht Teil von »M« ist und deshalb nur einmal ausgeführt werden soll.
n:	Verschieben des Zeichenzeigers an den Anfang der Zeile n.
n, -n	Verschieben des Zeichenzeigers um n Zeilen vorwärts (n oder +n) oder rückwärts (-n) und Ausgeben der Zeile.

Kommando	Funktion
:nk	Ausführen des Kommandos »k« bis zur Zeile n.
nNs^Z	Suchen des n-ten Auftretens der Zeichenkette »s« im Großschriftmodus. Zum Suchen im Kleinschriftmodus dient das Kommando »n«. CTRL-Z verwendet man zu einer weiteren Kommandoeingabe in derselben Zeile.
O	Beenden der ED-Editierung und Rücksprung in die Originaldatei »quelldatei« ohne Übernahme einer Änderung.
nP, -nP, 0P	Anzeigen von n Bildschirmseiten vorwärts (nP oder +nP) oder rückwärts (-nP), von der Position des Zeichenzeigers ausgehend. Bei »0P« wird nur eine Bildschirmseite ausgegeben. Die Länge einer Bildschirmseite beträgt 23 Zeilen.
Q	Beenden der ED-Editierung und Rücksprung ins CP/M. Die neue (editierte) Datei wird nicht gespeichert.
R^Z	Lesen der Datei »X$$$$$$$.LIB« in den Textpuffer.
R dateiname^Z	Lesen und Einfügen der angegebenen Datei »dateiname« in den Textpuffer.
nSs1^Zs2^Z	Suchen der Zeichenkette »s1« und Ersetzen durch »s2«, insgesamt n-mal wiederholen. Mit »Ss1^Zs2« wird der Austausch nur einmal ausgeführt. Das letzte CTRL-Z kann zum Abtrennen bei der Eingabe weiterer Kommandos in derselben Zeile dienen. Das Kommando »S« bewirkt eine automatische Zeichenumwandlung in den Großschriftmodus. Das Kommando »s« dient zur Zeichenverarbeitung im Kleinschriftmodus.
nT, -nT, 0T	Bildschirmausgabe von n Zeilen vorwärts (nT oder +nT) oder rückwärts (-nT), von der Position des Zeichenzeigers ausgehend. Mit »0T« oder »T« wird die Zeile ausgegeben, in der der Zeichenzeiger gerade steht. »B#T« listet den ganzen Pufferspeicher aus.

Kommando	Funktion
U, -U	Umwandeln aller Zeichen im Pufferspeicher in den Großschriftmodus. Mit »+U« bzw. »U« wird die Umwandlung ein-, mit »-U« wieder ausgeschaltet.
V, -V	Anzeige der Zeilennummern mit »+V« bzw. »V« einschalten und mit »-V« ausschalten.
0V	Anzeigen der verfügbaren und der gesamten Größe des Textpuffers in Bytes (dezimal).
nW	Schreiben der n ersten Zeilen in die Zwischendatei »quelldatei.$$$«. Mit »W« wird nur die aktuelle Zeile abgespeichert.
OW	Schreiben so vieler Zeilen in die Zwischendatei, bis der Puffer halb leer ist.
nX	Schreiben oder Anfügen der n folgenden Zeilen in die Zwischendatei »X$$$$$$$.LIB«. Diese n Zeilen können mit »R« wieder zurückgeholt werden, so daß eine Zeile leicht verschoben werden kann.
0X	Löschen der Datei »X$$$$$$$.LIB«.
nXf^Z	Schreiben oder Anfügen der n folgenden Zeilen in die Datei »f«.
0Xf	Löschen der Datei »f«.
nZ	Unterbrechen der laufenden ED-Kommandos von n Sekunden.

Anmerkungen zu Tabelle 6.4

1. Man kann für den Operanden »n« in den Kommandos dieser Tabelle auch die Form »n1::n2« verwenden. Hierbei arbeitet das ED-Kommando von Zeile n1 bis zur Zeile n2. Benutzt man diese Operandenform entweder ohne »n1:« oder »:n2«, wird für den fehlenden Operanden die Zeile eingesetzt, in der der Zeichenzeiger gerade steht.

2. Man kann in den ED-Kommandos für »n« ein »#« verwenden; hierfür wird dann der größtmögliche Wert, d.h. 65535, eingesetzt.

3. Die Kommandos »F«, »I«, »N« und »S« erzeugen eine automatische Umwandlung der Zeichen in den Großschriftmodus. Will man Zeichen im Kleinschriftmodus eingeben, sind die Kommandos »f«, »i«, »n« und »s« zu verwenden.

4. Mit CTRL-Z (^Z) kann man verschiedene ED-Kommandos in derselben Zeile voneinander trennen.

6.9 ERASE

Format: **ERASE datei [CONFIRM]**

Es bedeutet:

datei eine Dateibezeichnung »d:dateiname.typ;paßwort«

Beschreibung

Der Befehl »ERASE« löscht eine oder mehrere Dateien aus dem Disketten-Inhaltsverzeichnis (directory) einer Diskette in Laufwerk »d«. Ohne Angabe des Laufwerkparameters »d« werden die Dateien der Diskette im Bezugslaufwerk gelöscht. Das Disketten-Inhaltsverzeichnis und der Speicherplatz werden auf diese Weise für spätere Benutzung durch andere Dateien freigegeben.

Die Platzhalterzeichen »*« und »?« können verwendet werden. Mit »*« für die Parameter »filename« und »typ« können eine Gruppe von Dateien gelöscht werden. Um wichtige Daten nicht zu zerstören, sollte man den Platzhalter »*« nur mit äußerster Vorsicht verwenden.

ERASE kann mit ERA abgekürzt werden.

Option

CONFIRM Diese frei wählbare Option bewirkt, daß vor dem Löschen einer Datei eine Bestätigung bzw. Quittierung verlangt wird. CONFIRM kann mit C abgekürzt werden.

Beispiele

`A> ERASE PGM.TXT`	Löschen der Datei PGM.TXT auf der Diskette im Bezugslaufwerk A.
`A> ERA B:PGM.TXT`	Löschen der Datei PGM.TXT auf der Diskette im Laufwerk B.
`A> ERASE *.TXT`	Löschen aller Dateien des Dateityps TXT auf der Diskette im Bezugslaufwerk A.
`A> ERA PGM.*`	Löschen aller Dateien mit dem Dateinamen PGM und beliebigen Dateityps.
`A> ERA B:*.*` `ERASE B:*.*(Y/N)? Y`	Löschen aller Dateien auf der Diskette im Laufwerk B.

```
A> ERA B:*.* [C]
Confirm (Y/N)? Y
```

Löschen aller Dateien auf der Diskette im Laufwerk B. Jede Datei im Laufwerk B wird zwecks Bestätigung mit einem Fragezeichen angegeben.

6.10 GENCOM

Format: **GENCOM comdatei rsxdatei [o]**
mit o für LOADER, NULL und SCB=n,v

Es bedeuten:

comdatei	eine Dateibezeichnung im Format »d:dateiname.COM«
rsxdatei	eine Dateibezeichnung im Format »d2:dateiname2.RSX«
o	eine Option
n	ein Versatz (offset)
v	eine Hexadezimalzahl

Beschreibung

Der GENCOM-Befehl ist ein nichtresidentes Dienstprogramm, das eine spezielle COM-Datei mit angegliederten RSX-Dateien erzeugt. RSX-Dateien werden als residente Systemerweiterungen (RSX = resident system extension) verwendet. GENCOM setzt einen besonderen Kennsatz (header) an den Anfang der Ausgabe-Programmdatei, um ein notwendiges RSX-Laden anzuzeigen. Es kann auch ein Kennzeichen-Bit setzen, um das Ladeprogramm aktiv zu halten.

Mit GENCOM kann auch eine mittels GENCOM bereits verarbeitete Datei wieder zur Original-COM-Datei ohne Kennsatz und RSX-Dateien zurückverwandelt werden.

Optionen

LOADER — Setzen eines Kennzeichnungs-Bits, um das Ladeprogramm aktiv zu halten. Diese Option wird nur dann eingesetzt, wenn der COM-Datei keine RSX-Dateien angefügt werden. Es wird ein besonderer Kennsatz (header) von 256 Byte Länge hinzugefügt.

NULL — Es werden nur RSX-Dateien eingegeben. GENCOM richtet eine leere COM-Hilfsdatei für die RSX-Dateien ein. Der Name der erzeugten COM-Datei wird aus der ersten RSX-Dateibezeichnung gebildet.

SCB=(n,v) — Es wird der Systemsteuerblock (SCB = system control block) unter Verwendung der mit (n,v) angegebenen Hexadezimalwerte im Programm gebildet.

6.10.1 Anfügen von RSX-Dateien an eine COM-Datei

Format: **GENCOM comdatei rsxdatei [o]**
mit o für LOADER und SCB = (n,v)

Beschreibung

Es wird eine COM-Datei mit einem Anfangskennsatz und angefügten RSX-Dateien (maximal 15) erstellt. GENCOM geht davon aus, daß der erste Dateiname eine COM-Datei und die folgenden Namen RSX-Dateien bezeichnen. Die ursprüngliche COM-Datei wird durch die neu erstellte COM-Datei ersetzt.

Beispiel

```
A> GENCOM PGM PROG1 PROG2
```

Erzeugen der neuen COM-Datei PGM.COM mit den angefügten RSX-Dateien PROG1 und PROG2.

6.10.2 Erstellen einer COM-Datei nur aus RSX-Dateien

Format: **GENCOM rsxdatei [NULL]**

Beschreibung

Die RSX-Dateien werden an eine leere COM-Datei angefügt, deren Name aus der ersten RSX-Dateibezeichnung erstellt wird. Mit diesem Befehlsformat können direkt RSX-Dateien geladen werden.

Beispiel

```
A> GENCOM PROG1 PROG2 [NULL]
```

6.10.3 Umspeichern einer Datei mit angefügten RSX-Dateien in die ursprüngliche COM-Datei

Format: **GENCOM dateiname**

Beschreibung

GENCOM lädt die bereits mit GENCOM bearbeitete COM-Datei »dateiname«, entfernt den Anfangskennsatz, löscht alle angefügten RSX-

Dateien und speichert diese wieder im ursprünglichen Format als COM-Datei zurück.

Beispiel:

```
A> GENCOM PGM
```

6.10.4 Verändern (Hinzufügen oder Ersetzen) von RSX-Dateien

Format: **GENCOM comdatei rsxdatei [o]**
mit o für LOADER und SCB=(n,o).

Beschreibung

Hierbei werden RSX-Dateien einer bereits mit GENCOM verarbeiteten Datei verändert, indem die Liste der RSX-Dateien überprüft und ggf. eine bereits vorhandene Datei ausgetauscht bzw. eine noch nicht vorhandene hinzugefügt wird.

Beispiel

```
A> GENCOM PGM PROG1 PROG2
```

6.10.5 Anfügen eines Anfangskennsatzes

Format: **GENCOM dateiname [o]**
mit o für SCB=(n,v) oder [LOADER]

Beschreibung

Einer COM-Datei ohne RSX-Dateien wird bei gesetztem SCB- oder Ladekennzeichen-Bit ein GENCOM-Anfangskennsatz (header record) hinzugefügt. Mit dieser Befehlsform werden keine RSX-Dateien der COM-Datei angefügt.

Beispiel

`A> GENCOM PGM [LOADER]`	**Anfügen des Anfangskennsatzes und Setzen des Ladekennzeichen-Bits.**
`A> GENCOM PGM [SCB=(2,1)]`	**Byte 2 des SCB auf 1 setzen.**

6.11 GET

Format: **GET FILE datei [o]**
GET CONSOLE INPUT FROM FILE datei [o]
GET CONSOLE
GET CONSOLE INPUT FROM CONSOLE
mit o für ECHO oder NO ECHO bzw. SYSTEM

Es bedeutet:

datei — eine Dateibezeichnung im Format »d:dateiname.typ;paßwort«

Beschreibung

Das transiente Dienstprogramm GET steuert das Betriebssystem in der Weise, daß Konsoleneingaben aus einer Datei erfolgen. Die Datei kann CP/M-Befehle und/oder Eingaben für ein Benutzerprogramm enthalten.

Die Konsoleneingabe wird einer Datei entnommen, bis das Programm endet. Wenn vor Beendigung einer Programmeingabe die Datei abgearbeitet wurde, erwartet das Programm weitere Eingaben von der Konsole. Wenn das Programm endet, ehe die gesamte Eingabedatei gelesen wurde, erwartet das System weitere Eingaben von der Konsole.

Mit der SYSTEM-Option holt sich das System die nächste Anweisung unmittelbar aus der angegebenen Datei. Ohne SYSTEM-Option kann ein Systembefehl zum Aufruf eines Benutzerprogramms eingegeben werden, dessen Konsoleneingabe aus der mit GET bezeichneten Datei entnommen wird. Das System verlangt eine Konsoleneingabe, wenn das Dateiende erreicht wurde. Mit dem Befehl GET CONSOLE INPUT FROM CONSOLE als Befehlszeile in der Eingabedatei verlangt das System ebenfalls wieder eine direkte Konsoleneingabe.

6.11.1 Abrufen einer Konsoleneingabe aus einer Datei

Format: **GET FILE datei [o]**
GET CONSOLE INPUT FROM FILE datei [0]

Beschreibung

Das CP/M-System wird angewiesen, weitere Konsoleneingaben aus einer Datei abzurufen.

Tabelle 6.5: GET-Optionen

Option	Bedeutung
ECHO	Die Eingaben werden auf der Konsole (Bildschirm) angezeigt. Diese Option ist voreingestellt.
NO ECHO	Die Dateieingaben werden nicht auf der Konsole (Bildschirm) angezeigt. Eine Programmausgabe und die CP/M-Bereitschaftsmeldung werden nicht von dieser Option berührt und deshalb auf der Konsole angezeigt.
SYSTEM	Alle CP/M-Systemeingaben werden aus der im GET-Befehl bezeichneten Datei entnommen. GET entnimmt System- und Programmeingaben der Datei, bis die Datei abgearbeitet oder bis ein GET CONSOLE-Befehl aus der Datei gelesen wird.

Beispiele

```
A> GET FILE XINPUT
A> PROG
```

Aktivieren des GET-Dienstprogrammes.
Da die Systemoption nicht angegeben wurde, liest CP/M die nächste Befehlszeile von der Konsole und PROG aus.Wenn PROG eine Konsoleneingabe fordert, wird diese aus der Datei XINPUT genommen. Wenn PROG endet, kehrt CP/M für eine Konsoleneingabe zurück zur Konsole.

```
A> GET FILE XIN
[SYSTEM]
```

Da die SYSTEM-Option im Befehl enthalten ist, wird eine unmittelbar nachfolgende Konsoleneingabe aus der Datei XIN gelesen. CP/M verlangt wieder eine Konsoleneingabe, wenn das Ende der Datei XIN erreicht wird. Alternativ kann XIN das System zurück zur Konsole steuern, wenn XIN einen Befehl GET CONSOLE enthält.

6.11.2 Beenden der Konsoleneingabe aus einer Datei

Format: GET CONSOLE
GET CONSOLE INPUT FROM CONSOLE

Beschreibung

Das CP/M-System wird angewiesen, Konsoleneingaben von der Konsole (Tastatur) abzurufen.

Beispiel

```
A> GET CONSOLE
```

CP/M holt eine Konsoleneingabe von der Tastatur. Dieser Befehl kann in einer Datei benutzt werden, die vorher in einem Befehl GET FILE bezeichnet wurde, und die bereits von CP/M für eine Konsoleneingabe gelesen wurde. Der Befehl wird also benutzt, um die Konsoleneingabe zurück zur Konsole zu legen, ehe das Dateiende erreicht wird.

6.12 HELP

Format: **HELP t u [o]**
mit o für NOPAGE und LIST

Es bedeuten:

t	ein Begriff (topic)
u	Unterbegriffe (subtopic)

Beschreibung

Das transiente HELP-Dienstprogramm stellt eine Zusammenfassung von Informationen zu sämtlichen in diesem Buch beschriebenen CP/M3-Befehlen bereit. HELP gibt allgemeine Erklärungen zu einem CP/M-Befehl als Begriff (topic) und eingehende Beschreibungen zu einem Befehl als Unterbegriff (subtopic).

Mit HELP ohne Befehlsargumente wird eine Aufstellung aller verfügbaren Begriffe ausgegeben.

HELP mit einem Begriff als Befehlsargument gibt Erklärungen zu diesem Begriff aus, gefolgt von allen zu diesem Begriff zur Verfügung stehenden Unterbegriffen.

HELP mit einem Begriff und einem Unterbegriff als Befehlsargumente gibt Informationen zu diesem bestimmten Unterbegriff aus.

Nach jeder Informationsangabe erscheint eine spezielle Bereitschaftsmeldung »HELP>« auf dem Bildschirm.

- Der Zugriff zu Unterbegriffen wird jetzt durch die Eingabe eines Unterbegriffs mit vorangestelltem Punkt. Der Punkt bewirkt, daß die Suche nach dem Unterbegriff auf der letzten Begriffsebene durchgeführt wird.

  ```
  HELP>.u<CR>
  ```

- Die Eingabe eines einzelnen Punktes nach der Bereitschaftsmeldung wiederholt die letzte Ausgabe.

  ```
  HELP>.<CR>
  ```

- Die Eingabe eines Fragezeichens »?« nach der Bereitschaftsmeldung gibt die Liste der Begriffe aus.

```
HELP>?<CR>
```

- Das Betätigen der RETURN-Taste nach der Bereitschaftsmeldung bewirkt einen Rücksprung ins CP/M3.

```
HELP> <CR>
```

Die Namen der Begriffe und Unterbegriffe können abgekürzt werden. Gewöhnlich genügen die ersten zwei Buchstaben zur Identifizierung.

Tabelle 6.6: HELP-Optionen

Option	Bedeutung
NOPAGE	Ausschalten der voreingestellten Ausgabe je einer Bildschirmseite (die Anzahl der Zeilen pro Seite kann z.B. mit DEVICE eingestellt werden; beim C 128 sind 24 Zeilen voreingestellt.), d.h., die Anzeige scrollt automatisch bis zum Ende der Information. Mit CTRL-S kann die Ausgabe aufgehalten und mit CTRL-Q wieder fortgesetzt werden. NOPAGE kann mit N abgekürzt werden.
LIST	Wie NOPAGE, zusätzliche Zeilen zwischen den Rubriken sind jedoch unterdrückt. Mit CTRL-P kann man die Informationen auf einem Drucker ausgeben.

Beispiele

`A> HELP`	Liste der verfügbaren Begriffe.
`A> HELP DATE`	Allgemeine Angaben über den DATE-Befehl.
`A> HELP DIR OPTIONS [N]`	Erklärungen zu den Optionen des DIR-Befehls; kein Seitenmodus.
`A> HELP ED`	Allgemeine Angaben zum ED-Befehl.
`A> HELP ED COMMANDS`	Erklärungen zu den ED-Kommandos.
`A> HELP ED` `HELP>.COMMANDS`	Erklärungen zu den ED-Kommandos.

6.13 HEXCOM

Format: **HEXCOM dateiname**

Es bedeutet:

dateiname — eine HEX-Datei im Format »d:dateiname.HEX«

Beschreibung

Das transiente Dienstprogramm HEXCOM erzeugt aus einer Eingabedatei vom Dateityp HEX (HEX-Datei) eine direkt ablauffähige Befehlsdatei vom Dateityp COM. Der Name der übersetzten Ausgabedatei ist derselbe wie der der HEX-Eingabedatei, jedoch mit dem Dateityp COM. HEXCOM verlangt immer eine Eingabedatei mit dem Dateityp HEX.

Beispiel

```
A> HEXCOM B:PGM
```

Erzeugen der COM-Ausgabedatei PGM.COM aus der HEX-Eingabedatei PGM.HEX.

6.14 INITDIR

Format: **INITDIR d:**

Beschreibung

Mit einem INITDIR-Befehl kann ein Disketten-Inhaltsverzeichnis (directory) so initialisiert werden, daß die Dateieinträge auf der betreffenden Diskette mit Datums- und Zeitmarken versehen oder solche Markierungen entfernt werden.

Bei einer leeren Diskette bereitet INITDIR das Inhaltsverzeichnis für den Eintrag von Datums- und Zeitmarken vor. Bei einer Diskette, auf der schon Dateien eingetragen sind, prüft INITDIR die im Inhaltsverzeichnis für die Markierungen noch verbleibende Kapazität. Steht nicht mehr ausreichend Kapazität zur Verfügung, führt INITDIR keine Initialisierung des Inhaltsverzeichnisses für die Markierungen durch und antwortet mit einer Fehlermeldung.

Nach einer Initialisierung der Datums- und Zeitmarken müssen die Zeitmarken auf der Diskette mit einem SET-Befehl aktiviert werden.

Beispiel

```
A> INITDIR
ERROR: Unrecognized drive.
Drive:x

Enter drive:A

INITDIR WILL ACTIVATE TIME STAMPS FOR SPECIFIED DRIVE

Do you want to re-format the
directory on drive: A(Y/N)? Y
```

INITDIR aktiviert das Disketten-Inhaltsverzeichnis für Datums- und Zeiteinträge.

6.15 LIB

Format: **LIB datei [o1]**
LIB datei [o2] = datei m,datei m...

Es bedeuten:

datei	eine Dateibezeichnung im Format »d:dateiname.typ;paßwort«
o1	eine Option (I bzw. M bzw. P bzw. D)
o2	eine Option (I bzw. M bzw. P)
m	ein LIB-Parameter (modifier)

Beschreibung

Eine Bibliotheksdatei (Dateityp LIB) ist eine Zusammenstellung von Objektmodulen. Mit dem LIB-Dienstprogramm können Bibliotheksdateien angelegt, in bereits vorhandenen Bibliotheken Module angefügt, ausgetauscht, ausgewählt und gelöscht oder Kenntnis über den Inhalt von Bibliotheksdateien erhalten werden.

Mit LIB erzeugt und pflegt (verwaltet) man diese Bibliotheksdateien, die Objektmodule in Microsoft-REL-Dateiformat enthalten. Diese verschiebbaren Objektmodule können mit dem Makroassembler RMAC von Digital Research oder einem beliebigen anderen Übersetzungsprogramm zum Erzeugen von Modulen im Microsoft-REL-Format erzeugt werden.

Mit dem LINK-80-Dienstprogramm kann man nun Objektmodule aus einer Bibliotheksdatei mit anderen Objektdateien verbinden. LINK-80 wählt selbsttätig nur solche Module aus der Bibliotheksdatei aus, die für das zu bindende Programm benötigt werden, und bildet daraus eine ablauffähige Datei mit dem Dateityp COM.

Optionen

Die Bibliotheksdatei ist vom Dateityp REL oder IRL, je nach der gewählten Option. Module einer REL-Bibliothek dürfen keine Rückverweise auf bereits vorher vorkommende Module enthalten, weil LINK-80 normalerweise eine Bibliothek nur einmal durchläuft.

Tabelle 6.7: LIB-Optionen

Option	Bedeutung
I	Die INDEX-Option erzeugt eine indizierte Bibliotheksdatei vom Typ IRL. LINK-80 durchsucht diese schneller als nicht indizierte.
M	Mit der MODUL-Option werden die Modulnamen angezeigt.
P	Mit der PUBLICS-Option werden die Modulnamen und die globalen Variablen für die neue Bibliotheksdatei angezeigt.
D	Mit der DUMP-Option wird der Inhalt der Objektmodule im ASCII-Format angezeigt.

LIB-Parameter

Mit den LIB-Parametern in der Befehlszeile können Module in der Bibliotheksdatei gelöscht, ersetzt oder ausgewählt werden. Module, die gelöscht oder ausgetauscht werden sollen, stehen in spitzen Klammern »<...>«, und Module, die ausgewählt werden sollen, in runden Klammern »(...)«.

Werden keine Angaben gemacht, setzt LIB für alle Quelldateien den Dateityp REL voraus. Folgt auf einen Dateinamen eine in Klammern stehende Gruppe von Modulnamen, werden diese Module in die neue Bibliotheksdatei aufgenommen. Werden keine Module ausgegeben, übernimmt LIB sämtliche Module aus der Quelldatei in die neue Bibliotheksdatei.

Tabelle 6.8: LIB-Parameter

Parameter	Bedeutung
<modul=>	Löschen von »modul«
<modul=dateiname.REL>	Austauschen von »modul« gegen »dateiname.REL«. Kurzform, wenn Modul- und Dateiname identisch sind: <dateiname>
(mA-mX,m1,m4)	Auswählen der Module »mA« bis »mX« und »m1« und »m4«.

Beispiele

`A> LIB TEST[P]`	Anzeigen aller Module und globalen Variablen von TEST.REL.
`A> LIB TEST[P]=F1,F2`	Erzeugen von TEST.REL aus F1.REL und F2.REL.
`A> LIB TEST=T1(M1,M4), T2(C1-C4,C6)`	Erstellen von TEST.REL aus zwei Quelldateien, M1 und M4 aus T1.REL und C1 bis C4 und C6 aus T2.REL.
`A> LIB F2=F3<MA=>`	Erzeugen von F2.REL aus F3.REL unter Auslassung von MA in F3.REL.
`A> LIB F6=F5<MA=FILE.REL>`	Erzeugen von F6.REL aus F5.REL, wobei MA durch FILE.REL ersetzt wird.
`A> LIB F6=F5<FILE>`	Erzeugen von F6.REL aus F5.REL und Ersetzen des Moduls FILE (in F5.REL) durch das gleichnamige Modul FILE.REL.
`A> LIB F1[I]=B:F2(X1,X2,X4-X6)`	Erstellen von F1 im Laufwerk A aus den Modulen X1,X2,X4 bis X6 aus F2.REL im Laufwerk B.

6.16 LINK

Format: **LINK datei [o]=datei [o],...**

Es bedeutet:

datei	eine Dateibezeichnung im Format »d:dateiname.REL«

Beschreibung

Mit dem LINK-Befehl werden verschiebbare Objektmodule, die z.B. mit RMAC oder PL/I-80 erzeugt wurden, zu einer direkt ausführbaren COM-Datei verbunden (link = verbinden). Verschiebbare Dateien können externe Referenzmodule und globale Variable enthalten und Bezug auf Module in Bibliotheksdateien nehmen.

LINK durchsucht die Bibliotheksdateien und fügt die Referenzmodule in die Ausgabedatei ein. Der LINK-Befehl ist identisch mit dem bekannten Dienstprogramm LINK-80 von Digital Research.

Optionen

Mit den LINK-Optionen kann man den Ablauf von LINK-80 beeinflussen. Die LINK-Optionen stehen in rechteckigen Klammern hinter der Dateibezeichnung. Mehrere Optionen werden durch Kommas getrennt (Tabelle 6.9).

Tabelle 6.9: LINK-Optionen

Option	Bedeutung
A	Zusätzlicher Speicher. Der Pufferspeicher wird reduziert und Zwischendaten auf die Diskette geschrieben.
B	Verbinden eines BIOS in einem »gebankten« CP/M-System: - Anfügen eines Datenabschnitts an die Speicherseitengrenze, - Ablegen der Länge des Codesegments im Kopfdatensatz (header), - Ausgabedatei ist vom Dateityp SPR.
Dhhhh	Anfangsadresse der Daten. Die Anfangsadresse für den gemeinsamen Bereich und den Datenbereich wird festgelegt.
Gn	Startadresse (go). Die Startadresse wird auf das Label n gelegt.
Lhhhh	Laden. Die vorgegebene Ladeadresse eines Moduls kann von 0100H (Voreinstellung) auf den Wert hhhhH (hexadezimal) gelegt werden.
Mhhhh	Speichergröße. Ein zusätzlicher freier Speicherbereich kann für MP/M-Module vereinbart werden.
NL	Kein Auflisten der Symboltabelle auf der Konsole.
NR	Keine Datei für die Symboltabelle.
OC	Ausgabe einer Objektdatei vom Dateityp COM. Dies ist der voreingestellte Normalfall.
OP	Ausgabe einer verschiebbaren Objektdatei vom Dateityp PRL in verschiebbaren Segmenten zum Ablauf unter MP/M.
OR	Ausgabe einer RSP-Datei für residente Systemprozesse zum Ablauf unter MP/M.
OS	Ausgabe einer verschiebbaren Systemseitendatei vom Dateityp SPR zum Ablauf unter CP/M.

Option	Bedeutung
Phhhh	Programm-Anfangsadresse. Die Anfangsadresse eines Programms kann von 0100H (Voreinstellung) auf den Wert hhhhH (hexadezimal) gelegt werden.
Q	Auslisten von Symbolen mit voranstehenden Fragezeichen.
S	Suchen einer Bibliotheksdatei. Es werden nur die notwendigen Teile der vorstehenden Datei geladen; nur bei Bibliotheksdateien.
$Cd	Die Konsolenausgabe wird für d=X zur Konsole, für d=Y zum Drucker geleitet oder für d=Z unterdrückt. Der voreingestellte Normalfall ist X.
$Id	Die Zwischendateien werden im Laufwerk d (A bis E) gesucht. Der voreingestellte Normalfall ist das Bezugslaufwerk.
$Ld	Die Bibliotheksdateien werden im Laufwerk d (A bis E) gesucht. Der voreingestellte Normalfall ist das Bezugslaufwerk.
$Od	Die Objektdatei wird auf das Laufwerk d (A bis E) geschrieben. Für d=Z wird die Ausgabe unterdrückt. Die Voreinstellung ist das Laufwerk der ersten Datei im LINK-80-Befehl.
$Sd	Die Symboltabelle wird auf das Laufwerk d (A bis E) geschrieben. Für d=Y geht die Ausgabe zum Drucker und für d=Z wird sie unterdrückt.

Beispiele

`A> LINK B:PRG[NR]`	LINK-80 im Laufwerk A erzeugt aus PRG.REL im Laufwerk B die ausführbare Maschinencode-Datei PRG.COM im Laufwerk B. Eine Symboltabelle wird nicht erstellt.
`A> LINK M1,M2,M3`	LINK-80 verbindet die getrennt kompilierten Dateien M1, M2 und M3 zur ausführbaren Maschinencode-Datei M1.COM und berechnet ihre externen Referenzmodule.
`A> LINK M=M1,M2,M3`	LINK-80 verbindet die getrennt kompilierten Dateien M1, M2 und M3 zur ausführbaren Maschinencode-Datei M.COM.
`A> LINK PRG,FILE[S]`	Mit der S-Option durchsucht LINK-80 die Datei FILE als Bibliotheksdatei. LINK-80 verbindet PRG.REL mit den aufgerufenen Unterprogrammen aus FILE.REL zur Datei PRG.COM.

6.17 MAC

Format: **MAC dateiname $od**

Es bedeuten:

dateiname	eine Datei im Format »dateiname.ASM«
o	eine MAC-Option
d	ein Parameter für ein Ausgabegerät

Beschreibung

Das transiente MAC-Dienstprogramm, der CP/M-Makroassembler MAC, liest Anweisungen in Assemblersprache aus einer Disketten-Eingabedatei vom Dateityp ASM, assembliert diese Anweisungen und erstellt drei Ausgabedateien mit dem Eingabedateinamen und den Dateitypen HEX, PRN und SYM.

- Die Ausgabedatei »dateiname.HEX« enthält den Objektcode im Intel-Hexadezimal-Format. Mit einem Test- und Prüfprogramm, einem sogenannten Debugger, kann die HEX-Datei auf Fehler geprüft und mit dem HEXCOM-Befehl eine COM-Datei erstellt und diese dann ausgeführt werden.

- Die Ausgabedatei »dateiname.PRN« enthält das mit Kommentaren versehene ursprüngliche Quellprogramm, das entweder ausgedruckt oder auf der Konsole angezeigt werden kann. Die PRN-Datei enthält in der linken Seitenhälfte eine 16 Spalten breite Aufstellung, in der die Werte von Konstanten, Maschinencode-Adressen und der erzeugte Maschinencode aufgelistet sind. Konstante Adressen werden mit einem Gleichheitszeichen gekennzeichnet, um sie nicht mit Maschinencode-Adressen zu verwechseln.

- Die Ausgabedatei »dateiname.SYM« enthält eine sortierte Liste der im Programm definierten Symbole.

Vor dem Aufruf von MAC muß eine Quelldatei des Dateityps ASM mit den Assemblersprache-Anweisungen erstellt werden.

Optionen

Die Eingabe und Ausgabe von MAC kann mit den in den Tabellen 6.10 und 6.11 aufgeführten Optionen gesteuert werden. Nach der MAC-Option »o« kann mit einem Parameter »d« für ein Ausgabegerät das Ziellaufwerk, eine Ausgabe auf die Konsole, auf den Drucker oder eine Ausgabenunterdrückung festgelegt werden. In Tabelle 6.12 sind zusätzliche Parameter für das Ausgabeformat der Ausgabedateien angegeben.

Tabelle 6.10: MAC-Optionen

Option o	Bedeutung
A	Quellaufwerk der ASM-Datei (d= A-E).
H	Ziellaufwerk der HEX-Datei (d= A-E,Z).
L	Quellaufwerk der Makrobibliothek-LIB-Dateien, die mit der MACLIB-Anweisung aufgerufen werden.
P	Ziellaufwerk der PRN-Datei (d= A-E,X,P,Z).
S	Ziellaufwerk der SYM-Datei (d= A-E,X,P,Z).

Tabelle 6.11: Ausgabegerät

Parameter d	Bedeutung
A bis E	Laufwerk A-E, soweit vorhanden.
X	Konsolenausgabe (Bildschirm).
P	Druckerausgabe.
Z	keine Ausgabedatei.

Tabelle 6.12: Ausgabedatei-Parameter

Parameter	Bedeutung
+L	Auflisten der aus den Makrobibliotheks-LIB-Dateien gelesenen Eingabezeilen.
-L	Kein Auflisten (voreingestellter Normalfall).
+M	Auflisten aller Makrozeilen, wie sie während des Assembliervorgangs verarbeitet werden.
-M	Keine Ausgabe von Makrozeilen (voreingestellter Normalfall).
*M	Auflisten nur des hexadezimalen Maschinencodes, der beim Einsetzen von Makrozeilen erzeugt wurde.
+Q	Auflisten aller LOCAL-Symbole in der Symbolliste.
-Q	Keine Ausgabe aller LOCAL-Symbole in der Symbolliste (voreingesteller Normalfall).
+S	Anfügen der Symboldatei an die PRN-Datei.
-S	Kein Erstellen der Symboldatei.
+1	Erstellen einer Auflistung im Durchlauf 1 zur Makro-Fehlersuche in der PRN-Datei.
-1	Kein Erstellen einer Auflistung.

Beispiele

```
A> MAC PGM
```

Aufrufen von MAC im Laufwerk A und Bearbeiten der Datei PGM.ASM ebenfalls im Laufwerk A.

```
A> MAC PGM $PB AA HB SX
```

Aufrufen von MAC im Laufwerk A; Ausgabe der Datei PGM.PRN an Laufwerk B, Eingabe der Datei PGM.ASM von Laufwerk A, Ausgabe der Datei PGM.HEX an Laufwerk B und Ausgabe der Datei PGM.SYM an die Konsole.

6.18 PATCH

Format: **PATCH dateiname typ n**

Es bedeuten:

dateiname	ein Dateiname
typ	eine Typ-Option (COM, PRL oder SPR)
n	die Änderungsnummer (eine Zahl von 1 bis 32).

Beschreibung

Mit dem PATCH-Befehl (patch = ändern) kann die Änderungsnummer n im CP/M3-Betriebssystem oder in einer CP/M3-Befehlsdatei angezeigt oder eingefügt werden.

Nur CP/M3-Systemdateien des Dateityps COM, PRL oder SPR können mit dem PATCH-Befehl geändert werden. Ohne Angabe einer Typ-Option sucht das PATCH-Dienstprogramm nach der Datei »dateiname.COM«.

Bei einer erfolgreichen Änderung der Änderungsnummer erfolgt die Meldung:

```
Patch Installed.
```

Konnte die Änderung nicht durchgeführt werden, erfolgt die Meldung:

```
Patch not Installed
```

und eine der folgenden möglichen Fehlermeldungen:

`*.ERROR: Patch requires CP/M3.`	Änderung benötigt CP/M3.
`*.ERROR: Invalid filetype typ.`	Ungültiger Dateityp »typ«.
`*.ERROR: Serial Number mismatch.`	Falsche Seriennummer.
`*.ERROR: Invalid patch number n.`	Ungültige Änderungsnummer.

Beispiel

`A> PATCH SHOW 2`	Ändern der Systemdatei SHOW.COM mit der Änderungsnummer 2.

6.19 PIP

Format: **PIP d: [Gn]= quelldatei [o],...**
PIP zieldatei[Gn]=quelldatei [o],...
PIP zieldatei [Gn]=d:[o]

Es bedeuten:

d	ein Laufwerk
quelldatei	die Dateibezeichnung der Quelldatei im Format »d:dateiname.typ;paßwort«
zieldatei	die Dateibezeichnung der Zieldatei im Format »d:dateiname.typ;paßwort«
Gn	Benutzerbereich n der Zieldatei (einzige für eine Zieldatei angebbare Option)
o	Optionen für die Quelldatei

Beschreibung

Mit dem transienten PIP-Dienstprogramm können eine oder mehrere Dateien von einer Diskette (Laufwerk) bzw. einem Benutzerbereich auf andere kopiert werden. Die kopierte Datei kann danach noch umbenannt werden. Mit PIP können eine oder mehrere Dateien zu einer zusammengefaßt werden. PIP kann auch eine Textdatei von der Diskette auf einen Drucker oder eine andere zusätzliche logische Ausgabeeinheit übertragen werden. Mit Eingaben über die Konsole oder andere logische Eingabeeinheiten kann PIP eine Datei auf einer Diskette erstellen. Mit PIP können Daten von einer logischen Eingabeeinheit zu einer logischen Ausgabeeinheit übertragen werden, daher die Bezeichnung PIP (peripheral interchange program = peripheres Austauschprogramm).

Jeder PIP-Vorgang kann mit CTRL-C abgebrochen werden.

Attribute

PIP kopiert folgende Dateiattribute zusammen mit der Datei:

- RW ohne Schreibschutz,
- RO mit Schreibschutz,
- SYS Systemdatei,
- DIR Datei im Disketten-Inhaltsverzeichnis und
- F1 bis F4 vom Benutzer definierbare Attribute.

Paßwort

Bei einer mit einem Paßwort geschützten Datei muß das Paßwort in der Befehlszeile nach dem Dateinamen und/oder Dateityp eingegeben werden. Bei Eingabe eines falschen Paßwortes wird die Datei übergangen und eine Fehlermeldung ausgegeben.

- Wenn eine Zieldatei mit einem Paßwort gekennzeichnet wird, wird automatisch für diese Datei der Paßwortschutz-Modus auf READ eingestellt.

- Bei Angabe einer Zieldatei ohne Paßwort weist PIP dieser Datei kein Paßwort zu.

- Bei alleiniger Angabe eines Ziellaufwerkes weist PIP der Zieldatei denselben Paßwortschutz der Quelldatei zu.

Einzellaufwerk

Wenn nur ein Einzellaufwerk zur Verfügung steht, wird als Ziellaufwerk das virtuelle Laufwerk E angegeben. Während des Kopierens erscheinen dann Aufforderungen zum Wechseln der Quell- und Zieldisketten.

6.19.1 Kopieren einer einzelnen Datei

Format: **PIP d:[Gn]=quelldatei [o]**
PIP zieldatei [Gn]=d: [o]
PIP zieldatei [Gn]=quelldatei [o]

Beschreibung

PIP sucht im angegebenen Laufwerk nach der Quelldatei und kopiert sie in die Zieldatei auf demselben oder dem angegebenen Laufwerk. Vor dem Kopiervorgang sollte sichergestellt sein, daß zur Aufnahme der Zieldatei ausreichend freie Speicherkapazität vorhanden ist; z.B. die Dateilänge mit dem DIR-Befehl und die freie Disketten-Speicherkapazität mit dem SHOW-Befehl. Mit einem ERASE-Befehl können alte Dateien gelöscht werden.

Dateien werden erst in eine Zwischendatei »zieldatei.$$$« kopiert, um sicherzustellen, daß die Dateien auf der Diskette ausreichend Platz haben.

Kann der Kopiervorgang erfolgreich abgeschlossen werden, ändert PIP den zwischenzeitlichen Dateityp »$$$« in den für die Zieldatei festgelegten Dateityp um. Existiert bereits eine alte Datei mit dem Namen der Zieldatei, so wird diese alte Datei gelöscht.

Beispiele

`A> PIP B:=A:FILE.DAT` `A> PIP B:FILE.DAT=A:`	Kopieren der Datei FILE.DAT von Laufwerk A auf Laufwerk B.
`A> PIP B:NEU.DAT=A:ALT.DAT`	Kopieren von ALT.DAT auf Laufwerk A in NEU.DAT auf Laufwerk B.
`A> PIP NEU.DAT=ALT.DAT`	Kopieren von ALT.DAT in NEU.DAT auf derselben Diskette in Laufwerk A.
`A> PIP B:PGM.BAK=A:PGM.DAT[G1]`	Kopieren von PGM.DAT des Benutzerbereichs 1 im Laufwerk A in PGM. BAK im Laufwerk B unter der angegebenen Benutzernummer 1.

6.19.2 Kopieren mehrerer Dateien

Format: **PIP d:[Gn]=d:quelldatei [o]**
mit »quelldatei« für Quelldateibezeichnungen mit Platzhalterzeichen

Beschreibung

Mit den Platzhalterzeichen »*« und »?« in den Dateibezeichnungen einer Quelldatei können mehrere Dateien mit einem PIP-Befehl kopiert werden. Hierbei werden die zutreffenden Dateien eine nach der anderen vom Quellaufwerk auf das Ziellaufwerk unter Beibehalten der ursprünglichen Namen kopiert.

Beispiele

`A> PIP B:=A:*.COM`	Kopieren aller COM-Dateien vom Laufwerk A auf das Laufwerk B.
`A> PIP B:=A:*.*`	Kopieren aller Dateien vom Laufwerk A auf das Laufwerk B. Hiermit kann eine Sicherungskopie einer Originaldiskette erstellt werden, jedoch nicht die CP/M3-Systemspuren (siehe COPYSYS).
`A> PIP B:=A:PGM???.*`	Kopieren aller Dateien mit einem 3- bis 6stelligen Dateinamen, der mit PGM beginnt, von Laufwerk A auf das Laufwerk B.
`A> PIP B:[G1]=A:*.BAS`	Kopieren aller BAS-Dateien unter dem Benutzerbereich 0 im Laufwerk A auf das Laufwerk B unter dem Benutzerbereich 1.

6.19.3 Verknüpfen von Dateien

Format: **PIP zieldatei [Gn]=quelldatei1 [o],quelldatei2 [o],...**

Beschreibung

Mit PIP kann der Inhalt von zwei und mehreren Dateien in eine Datei kopiert werden (Dateiverkettung), dabei werden die Quelldateien aus dem PIP-Befehl von links nach rechts verknüpft, d.h. zuerst »quelldatei1«, dann »quelldatei2« usw. Für jede Quelldatei können eine oder mehrere Optionen angegeben werden.

Optionen

Alle PIP-Optionen erzwingen eine zeichenweise Übertragung, mit Ausnahme der folgenden:

A, C, Gn, K, O, R, V und W (Tabelle 6.14).

Bei der zeichenweisen Übertragung sucht PIP nach einem CTRL-Z-Zeichen, dem Dateiende-Zeichen (end-of-file-marker), wobei PIP dann bei einem CTRL-Z-Zeichen die Zeichenübertragung beendet.

Bei einer normalen Verknüpfung von Dateien sucht PIP lediglich den letzten Datensatz einer Datei nach dem CTRL-Z-Dateiende-Zeichen ab. Zur Verknüpfung von Maschinencode-Dateien werden die eingefügten CTRL-Z-Zeichen mit der O-Option ignoriert.

Beispiele

`A> PIP NEU=ALT1,ALT2,ALT3`	Verknüpfen der Dateien ALT1, ALT2 und ALT3 zur Datei NEU.
`A> PIP B:X.BAS=Y.BAS,B:Z.BAS`	Verknüpfen von Y.BAS in Laufwerk A und Z.BAS in Laufwerk B zur Datei X.BAS in Laufwerk B.

6.19.4 Kopieren von Dateien auf und von Zusatzeinheiten

Format: **PIP ziel-ld=quell-ld [o]**
PIP zieldatei [Gn]=quell-ld [o]
PIP ziel-ld=quelldatei [o]

Es bedeuten:

ziel-ld	logische Zieleinheit/Gerät (AUX:, CON:, PRN: und LST:)
quell-ld	logische Quelleinheit/Gerät (AUX: [o], CON: [o], NUL: und EOF:)
zieldatei	Dateibezeichnung der Zieldatei
quelldatei	Dateibezeichnung der Quelldatei

Beschreibung

Mit PIP kann eine Datei auch von einer Diskette auf ein Gerät, von einem Gerät auf eine Diskette oder von einem Gerät auf ein anderes kopiert werden. Jedes Peripheriegerät wird einer logischen Einheit zugewiesen, d.h. einer logischen Quelleinheit und einer logischen Zieleinheit (siehe DEVICE-Befehl). Jeder Name einer logischen Einheit schließt mit einem Doppelpunkt ab, um ihn nicht mit Dateinamen zu verwechseln. Die Dateien müssen druckbare Zeichen enthalten.

Tabelle 6.13: Logische PIP-Einheiten

Name	Logische Einheit
CON:	Konsoleneingabe- und Ausgabeeinheit: - Quelleinheit (Tastatureingabe), - Zieleinheit (Bildschirmausgabe)
AUX:	Eingabe- und Ausgabe-Zusatzeinheit.
LST:	Ausgabe auf einer Druckereinheit (Drucker).
NUL:	Quelleinheit, die 40 hexadezimale Null-Zeichen erzeugt.
EOF:	Quelleinheit, die ein einziges CTRL-Z-Zeichen, das Dateiende-Zeichen (ASCII-Wert 1AH) erzeugt.
PRN:	spezielle Zieleinheit (Ausgabe auf einer Druckereinheit), mit Tabulatorstops in jeder achten Spalte, Zeilennumerierung und Seitenvorschub nach jeder 60. Zeile.

Beispiele

`B> PIP PRN:=CON:,FILE.DAT`	Zunächst Übertragen von der Konsole (Tastatur) auf eine Druckereinheit, bis zur Eingabe von CTRL-Z. Danach Ausgabe von FILE.DAT im Laufwerk B auf die Druckereinheit.
`A> PIP B:PGM.BAS=CON:`	Jede Konsoleneingabe (Tastatur) wird in die Datei PGM.BAS im Laufwerk B geschrieben.
`A> PIP LST:=CON:`	Jede Konsoleneingabe (Tastatur) wird auf der Druckereinheit ausgedruckt. Übertragung wird mit CTRL-Z beendet.
`A> PIP PRN:=B:PGM.BAS`	Die Datei PGM.BAS im Laufwerk B wird auf der Druckereinheit ausgedruckt.

6.19.5 Kommandomodus

Format: **PIP <CR>**

Beschreibung

Das PIP-Dienstprogramm kann auch im sogenannten Kommandomodus verwendet werden. Nach dem Laden von PIP meldet es sich mit einem Stern »*«. Nach dieser PIP-Bereitschaftsmeldung können nun beliebige PIP-Kommandos, wie unter den oben erwähnten PIP-Befehlen beschrieben, eingegeben werden.

Man unterbricht das PIP-Dienstprogramm durch Drücken der RETURN-Taste <CR> nach der PIP-Bereitschaftsmeldung oder durch CTRL-C.

Beispiel

```
A> PIP
CP/M 3 PIP VERSION 3.0
* NEU = ALT1,ALT2,ALT3
* A:=B:X.BAS
* B:=*.*
* ^M
A>
```

6.19.6 Verwenden von PIP-Optionen

Erklärung

Mit den PIP-Optionen können die Quelldateien auf verschiedene Arten verarbeitet werden (Tabelle 6.14). Eine Option muß unmittelbar auf die angesprochene Datei oder Einheit folgen. Sie muß in rechteckige Klammern »[ ]« gesetzt werden. Zwischen einem Optionszeichen und einer nachfolgenden Zahl dürfen keine Leerzeichen stehen.

Mehrere Optionen in einem Klammernpaar können unmittelbar hintereinander stehen oder durch Leerzeichen getrennt werden.

Beispiele

`A> PIP PGM.BAS=CODE.BAS[L],DAT.BAS[U]`	Kopieren und Verknüpfen von CODE.BAS in Kleinbuchstaben und DAT.BAS in Großbuchstaben in der Datei PGM.BAS.
`A> PIP CON:=PGM.BAS[D80]`	Konsolenausgabe von PGM.BAS; sämtliche Zeichen hinter der 80. Spalte werden gelöscht.
`A> PIP B:=BRIEF.TXT[E]`	Kopieren BRIEF.TXT aus Laufwerk A auf das Laufwerk B, gleichzeitige Bildschirmausgabe.
`A> PIP B:=A:*.COM[VWR]`	Kopieren aller COM-Dateien von Laufwerk A auf Laufwerk B; Verifizieren (V) der Quell- und Zieldatei; Überschreiben (W) von schreibgeschützten Zieldateien; Kopieren von Systemdateien (R).

Tabelle 6.14: PIP-Optionen

Option	Bedeutung
A	Archiv. Kopieren nur der Dateien, die nach dem letzten Kopieren verändert wurden, z.B. zum Aktualisieren einer Sicherungskopie. Das Archivattribut der Datei wird eingeschaltet (siehe SET-Befehl).
C	Kopiervorgang bestätigen (confirm). Vor Durchführen eines Kopiervorgangs wird jedesmal eine Bestätigung verlangt, z.B. Kopieren von ausgewählten Dateien.
Dn	Ausblenden aller Zeichen nach der n-ten Spalte in einer Zeile, z.B. Begrenzen der Zeilenlänge bei einer Druckerausgabe auf die druckbare Zeilenlänge (delete after column n).
E	Echo (echo). Wiederholen aller Zeichen während des Kopierens auf der Konsole (Bildschirm).
F	Herausfiltern von Seitenvorschüben (filter form feed). Alle Seitenvorschubzeichen (ASCII-Wert 0CH oder CTRL-L) einer Quelldatei werden beim Kopieren gelöscht.
Gn	Kopieren in oder aus dem Benutzerbereich n; n zwischen 0 und 15.
H	Hexadezimale Datenübertragung. Überprüfen der Daten auf korrekte Intel-Hexadezimal-Dateiformate. Fehler werden auf der Konsole (Bildschirm) angezeigt.
I	Unterdrücken (ignore) von :00-Datensätzen bei der Übertragung einer Datei im Intel-Hexadezimal-Format. Mit der I-Option wird automatisch die H-Option aktiviert.
L	Umwandeln aller Großbuchstaben (A-Z) der Quelldatei in Kleinbuchstaben (a-z) der Zieldatei. Alle anderen Zeichen bleiben erhalten.

Option	Bedeutung
N	Zeilennummern an den Anfang jeder Zeile schreiben (line numbers). Die Zeilennummern beginnen mit 1 und werden für jede Zeile um 1 weitergezählt. Auf die Zeilennummer folgt ein Doppelpunkt. Mit »N2« werden die Zeilennummern mit vorstehenden Nullen aufgefüllt und hinter jeder Nummer ein Tabulator eingefügt. Mit »T« kann der Tabulator vergrößert werden.
O	Übertragen von Objektdateien im Maschinencode, d.h. von nicht druckbaren Nicht-ASCII-Dateien (object file transfer). Das Dateiende-Zeichen CTRL-Z wird wie ein normales Zeichen übertragen und beendet die Übertragung nicht.
Pn	Einstellen der Seitenlänge (set page length). Nach n Zeilen wird ein Seitenvorschub-Zeichen (ASCII-Wert 0CH oder CTRL-L) eingefügt. Der voreingestellte Normalfall (n=1) fügt PIP nach jeder 60. Zeile einen Seitenvorschub ein.
Qs^Z	Beenden des Kopiervorgangs nach der Zeichenkette s (quit copying). CTRL-Z dient als Begrenzung der Zeichenkette s.
R	Kopieren von Systemdateien mit dem Attribut SYS (read system files). Systemdateien können nur mit dieser Option kopiert werden.
Ss^Z	Beginn des Kopiervorgangs ab der Zeichenkette s (start copying). CTRL-Z dient als Begrenzung der Zeichenkette s.
Tn	Erweitern der Tabulatorzeichen (expand tabs). Die Tabulatorzeichen CTRL-I (ASCII-Wert 09H) werden durch so viele Leerzeichen erweitert, daß das nächste Zeichen in eine durch n dividierbare Spalte gesetzt wird.
U	Umwandeln der Kleinbuchstaben der Quelldatei in Großbuchstaben in der Zieldatei. Alle anderen Zeichen bleiben unverändert.
V	Überprüfen (verify = verifizieren) auf fehlerfreie Datenübertragung zwischen Quell- und Zieldatei. Die Zieldatei muß eine Diskettendatei sein.

Option	Bedeutung
W	Direktes Überschreiben von schreibgeschützten Dateien mit dem R/O-Attribut, d.h., es erfolgt keine Rückfrage auf dem Bildschirm.
Z	Paritätsbit (8. Bit) in ASCII-Zeichen auf Null setzen (zero the parity bit); nur bei Quelldateien mit Zeichen möglich.

6.20 PUT

Format: **PUT CONSOLE OUTPUT TO FILE datei [o]**
PUT PRINTER OUTPUT TO FILE datei [o]
PUT CONSOLE OUTPUT TO CONSOLE
PUT PRINTER OUTPUT TO PRINTER

Es bedeutet:

datei eine Dateibezeichnung im Format »d:dateiname.typ;paßwort«

Beschreibung

Mit dem transienten PUT-Dienstprogramm kann eine Konsolen- oder Druckerausgabe in eine Datei umgeleitet werden. Es kann damit also auch eine Ausgabe eines Benutzerprogramms statt auf die Konsole oder den Drucker in einer Datei abgelegt werden. Eine Ausgabe wird so lange in eine Datei geleitet, bis das Programm endet; dann wird die Ausgabe wieder zurückgegeben.

Mit der SYSTEM-Option werden alle nachfolgenden Konsolen-/Druckerausgaben in die angegebene Datei geschickt; diese Ausgabe muß mit dem entsprechenden PUT-Befehl beendet werden.

Tabelle 6.15: PUT-Optionen

Option	Bedeutung
ECHO	Ausgabe zusätzlich auf die Konsole (Bildschirm). Voreingestellt bei Konsolenausgabe.
NO ECHO	Keine Ausgabe auf die Konsole (Bildschirm).
FILTER	Umwandeln von CTRL-Steuerzeichen in druckbare Zeichen, z.B. ESCAPE-Zeichen in »^[«.
NO FILTER	Kein Umwandeln von CTRL-Steuerzeichen (voreingestellter Normalfall).
SYSTEM	Betriebssystem- und Programmausgaben werden in die mit »datei« bezeichnete Datei geschrieben, bis ein PUT-CONSOLE-Befehl die Ausgabe an die Konsole zurückgibt.

6.20.1 Konsolenausgabe in eine Datei

Format: **PUT CONSOLE OUTPUT TO FILE datei [o]**

Beschreibung

Hiermit wird eine Konsolenausgabe in eine Datei geschickt.

Beispiel

```
A> PUT CONSOLE OUTPUT TO FILE AUSG [ECHO]
```

Umleiten der Konsolenausgabe in die Datei AUSG und Anzeigen auf dem Bildschirm.

6.20.2 Druckerausgabe in eine Datei

Format: **PUT PRINTER OUTPUT TO FILE datei [o]**

Beschreibung

Eine Druckerausgabe wird in eine Datei geschickt. Der voreingestellte Normalfall ist die Option »NO ECHO«. Mit »ECHO« wird diese Druckerausgabe auch auf den Drucker geleitet.

Beispiel

```
A> PUT PRINTER OUTPUT TO FILE AUSG
A> PGM
```

Die Druckerausgabe des Programms PGM wird in die Datei AUSG geschickt.

6.20.3 Konsolenausgabe in eine Datei beenden

Format: **PUT CONSOLE OUTPUT TO CONSOLE**

Beschreibung

Die Konsolenausgabe wird zurück an die Konsole gegeben.

6.20.4 Druckerausgabe in eine Datei beenden

Format: **PUT PRINTER OUTPUT TO PRINTER**

Beschreibung

Die Druckerausgabe wird zurück an den Drucker gegeben.

6.21 RENAME

Format: **RENAME neudatei=altdatei**

Es bedeuten:

neudatei	neue Dateibezeichnung im Format »d:dateiname.typ;paßwort«
altdatei	alte Dateibezeichnung im Format »d:dateiname.typ;paßwort«

Beschreibung

Mit dem RENAME-Befehl kann der Name einer im Disketten-Inhaltsverzeichnis aufgeführten CP/M-Datei vom bisherigen Namen »altdatei« in den neuen Namen »neudatei« geändert werden. Es wird keine Kopie hergestellt. RENAME kann mit REN abgekürzt werden. Ist der neue Name mit dem Namen einer bereits auf der Diskette gespeicherten Datei identisch, wird vor einer Umbenennung erst gefragt, ob letztere gelöscht werden kann.

In der neuen Dateibezeichnung kann ein Laufwerk angegeben werden, in dem dann die umzubenennende Datei zu finden sein sollte. Wird in beiden Dateibezeichnungen ein Laufwerk genannt, muß es das gleiche sein.

Bei Verwendung der Platzhalterzeichen »*« und »?« in den Dateibezeichnungen können gleichzeitig mehrere Dateinamen geändert werden. Gleichzeitig müssen die Platzhalter in der alten und neuen Dateibezeichnung genau übereinstimmen.

Beispiele

`A> REN NEU.BAS=ALT.BAK`	Umbenennen der Datei ALT.BAK in NEU.BAS.
`A> RENAME B:X.PAS=Y.PLI` oder `A> RENAME X.PAS=B:Y.PLI`	Umbenennen der Datei Y.PLI in X.PAS auf dem Laufwerk B.
`A> REN B:NEU=B:ALT`	Umbenennen der Datei ALT in NEU auf dem Laufwerk B.
`A> REN *.TEX=*.TXT`	Umbenennen aller Dateien mit dem Dateityp TXT in solche mit dem Dateityp TEX.
`A> REN A*.T*=S*.T*`	

6.22 RMAC

Format: **RMAC datei $od**

Es bedeuten:

datei	eine Dateibezeichnung im Format »d:dateiname.ASM;paßwort«
o	eine RMAC-Option (R, S oder P)
d	eine Ausgabeoption für eine Laufwerksangabe

Beschreibung

Der Makroassembler RMAC übersetzt eine ASM-Datei in eine verschiebbare REL-Datei, mit der dann in einem LINK-Befehl in eine COM-Datei erzeugt werden kann.

Optionen

Die RMAC-Optionen legen das Ziel der Ausgabedateien fest. Mit dem zusätzlichen Ausgabeparameter »d« wird das Ziellaufwerk für die Ausgabedateien gewählt.

Im MAC-Befehl bestimmt die H-Option das Ziel der HEX-Datei, während im RMAC-Befehl die R-Option das Ziel der REL-Datei festlegt. Da REL-Dateien keine ASCII-Dateien sind, ist eine Konsolenausgabe (mit $RX) und eine Druckerausgabe (mit $RP) nicht möglich.

Tabelle 6.16: RMAC-Optionen

Option	Bedeutung
R	Erzeugen einer REL-Datei (d=A-E, Z)
S	Erzeugen einer SYM-Datei (d=A-E, X,P,Z)
P	Erzeugen einer PRN-Datei (d=A-E, X,P,Z)

Tabelle 6.17: Ausgabegeräte

Parameter d	Bedeutung
A bis E	Laufwerk A bis E
X	Konsolenausgabe (Bildschirm)
P	Druckerausgabe
Z	keine Ausgabe

Beispiel

```
A> RMAC TEST $PX SB RB
```

RMAC assembliert die Datei TEST.ASM aus Laufwerk A, schickt die Druckdatei TEST.PRN an die Konsole (Bildschirm), die Symboldatei TEST.SYM und die verschiebbare Objektdatei TEST.REL ins Laufwerk B.

6.23 SAVE

Format: SAVE

Beschreibung

Das SAVE-Dienstprogramm kopiert den Inhalt eines Teils des Speichers in eine Datei. Zur Verwendung von SAVE wird dieses zunächst gestartet und anschließend ein Programm gestartet, das eine Datei in den Speicher einliest. Nach dem Programmende meldet sich das SAVE-Dienstprogramm zurück und fragt nach der Datei, in die der Speicher kopiert werden soll, sowie nach Anfangs- und Endadresse des zu sichernden Speicherbereichs.

Beispiel

`A> SAVE`	Aktivieren des SAVE-Dienstprogramms,
`A> SID datei.com`	Programm (hier SID), das eine Datei (hier datei.com) in den Speicher lädt,
`#go`	Ausführen des Programms,
`SAVE Ver 3.0`	nach Programmende meldet sich SAVE,
`File (or RETURN to exit)? datei2.com`	Eingabe einer neuen Dateibezeichnung,
`Delete datei2.com?`	Y
`From? 100`	Anfangsadresse (100H),
`TO? 400`	Endadresse (400H) des zu sichernden Speicherbereichs,
`A>`	Rücksprung ins CP/M.

6.24 SET

Format: **SET [o]**
SET d:[o]
SET datei [o]

Beschreibung

Mit dem SET-Befehl werden

- im Disketten-Inhaltsverzeichnis der Paßwortschutz und Zeitmarken,
- im Laufwerk der Schreibschutz und
- bei Dateien die Dateiattribute (z.B. DIR, SYS, RO, RW usw.) initialisiert.

6.24.1 Setzen der Dateiattribute

Format: **SET datei [o]**

Beschreibung

Mit dem SET-Befehl werden die in Tabelle 6.18 aufgeführten Attribute für eine oder mehrere Dateien gesetzt.

Beispiel

`A> SET PGM.TEX [RO,SYS]`	Setzen der Datei PGM.TEX auf Schreibschutz und System SYS.
`A> SET PGM.TEX [RW,DIR]`	Setzen der Datei PGM.TEX auf Schreib-/Lesezugriff und Disketten-Inhaltsverzeichnis DIR.

Tabelle 6.18: SET-Dateiattribute

Option	Bedeutung
DIR	Ändern des SYS-Attributs in das DIR-Attribut, d.h., die Datei wird aus dem SYS- in das normale DIR-Inhaltsverzeichnis gesetzt.
SYS	Setzen des System-(SYS-)Attributs für die Datei.
RO	Schreibschutz (nur Lesezugriff).
RW	Lese- und Schreibzugriff.
ARCHIVE=OFF	Entfernen des Archivattributs. Eine Datei ohne Archivattribut ist nicht gesichert (archiviert), d.h. mit der PIP-Option [A] kann eine Datei mit ausgeschaltetem Archivattribut kopiert werden.
ARCHIVE=ON	Einschalten des Archivattributs. Eine Datei mit Archivattribut ist gesichert (archiviert). Die PIP-Option [A] schaltet ebenfalls das Archivattribut ein. Das Archivattribut wird mit DIR angezeigt.
F1=ON oder OFF F2=ON oder OFF F3=ON oder OFF F4=ON oder OFF	Ein- bzw. Ausschalten eines vom Benutzer definierbaren Dateiattributs F1, F2, F3 oder F4.

6.24.2 Setzen der Laufwerkattribute

Format: **SET d: [RO]**
SET d: [RW]

Erklärung

Diese SET-Befehle setzen für das angegebene Laufwerk einen Schreibschutz RO (Nur-Lesezugriff) oder einen Lese-/Schreibzugriff RW. Bei einem schreibgeschützten Laufwerk kann keine Datei

- mit PIP hineinkopiert,
- mit ERASE gelöscht oder
- mit RENAME umbenannt werden,

d.h., es kann mit diesem Laufwerk keine Schreiboperation auf einer Diskette erfolgen.

Mit CTRL-C nach der CP/M-Bereitschaftsmeldung werden alle Laufwerke auf Lese-/Schreibzugriff gesetzt.

Beispiel

```
A> SET B: [RO]          Schreibschutz für Laufwerk B.
```

6.24.3 Diskettenname (Disketten-Label)

Format: **SET d: [NAME=labelname.typ]**

Es bedeuten:

labelname	Diskettenname aus bis zu 8 Zeichen
typ	Diskettentyp aus bis zu 3 Zeichen

Beschreibung

Die Diskette im angegebenen Laufwerk wird mit einem Namen (Label) gekennzeichnet, um z.B. Disketten einfacher katalogisieren zu können. Der Diskettenname besteht wie ein Dateiname aus einem bis zu acht Zeichen langen Namen und ggf. aus einem bis zu drei Zeichen langen Typ.

Beispiel

```
A> SET [NAME=DISK 100]          Benennen der Diskette im Laufwerk A mit DISK 100.
```

6.24.4 Zuweisen eines Paßwortes zu einer Diskette

Format: **SET [PASSWORD=paßwort]**
SET [PASSWORD=<CR>

Beschreibung

Die erste Form des SET-Befehls weist dem Diskettennamen ein Paßwort zu, mit der zweiten wird der Paßwortschutz aufgehoben. Bei einer Diskette mit Paßwortschutz muß das Paßwort bei CP/M-Befehlen, die den Diskettennamen verwenden, immer angegeben werden.

Beispiel

```
A> SET [PASSWORD=GEHEIM]
A> SET [PASSWORD=<CR>
```

6.24.5 Aktivieren des Paßwortschutzes für Dateien

Format: **SET [PROTECT=ON]**
SET [PROTECT=OFF]

Beschreibung

Die erste Form des SET-Befehls aktiviert den Paßwortschutz für sämtliche Dateien auf einer Diskette. Dieser Schutz muß vor einer Zuordnung von Paßworten aktiviert werden.

Mit dem zweiten SET-Befehl wird der Paßwortschutz wieder aufgehoben.

6.24.6 Zuweisen eines Paßwortes zu Dateien

Format: **SET-datei [PASSWORD=paßwort]**

Beschreibung

Mit diesem SET-Befehl wird der Datei (mit der Dateibezeichnung »datei«) das im Befehlsargument angegebene Paßwort zugeordnet. Paßworte können bis zu acht Zeichen umfassen. Kleinbuchstaben werden in Großbuchstaben umgewandelt.

In den Dateibezeichnungen können die Platzhalter »*« und »?« verwendet werden; dasselbe Paßwort wird dann allen zutreffenden Dateien zugeordnet.

Ein Zugriff auf eine paßwortgeschützte Datei ist nur mit dem Dateipaßwort möglich, sofern nicht der Paßwortschutz für die Diskette aufgehoben wird. Ohne das Diskettenpaßwort kann jedoch auch dieser Schutz nicht aufgehoben werden.

Beispiel

```
A> SET PGM.BAS [PASSWORD=SCHUTZ]
```

6.24.7 Schutzarten für paßwortgeschützte Dateien

Format: **SET datei [PROTECT=o]**
mit o für READ ,WRITE, DELETE und NONE.

Beschreibung

Es können vier verschiedene Schutzarten eines Paßwortschutzes für eine Diskettendatei gewählt werden (Tabelle 6.19)

Tabelle 6.19: Arten des Paßwortschutzes für Dateien

Option	Schutzart
READ	Paßwort erforderlich zum Lesen, Kopieren, Beschreiben, Löschen oder Umbenennen einer Datei.
WRITE	Paßwort erforderlich zum Beschreiben, Löschen oder Umbenennen der Datei. Kein Paßwort erforderlich zum Lesen der Datei.
DELETE	Paßwort erforderlich zum Löschen oder Umbenennen der Datei. Kein Paßwort erforderlich zum Lesen oder Ändern der Datei.
NONE	Die Datei hat kein Paßwort, bzw. das Paßwort wird gelöscht.

Beispiel

```
B> SET *.TEX [PASSWORD=GEHEIM PROTECT=WRITE]
```

Alle TEX-Dateien im Laufwerk B erhalten das Paßwort GEHEIM und werden schreibgeschützt.

6.24.8 Zuordnen eines voreingestellten Standardpaßwortes

Format: **SET [DEFAULT=paßwort]**

Beschreibung

Mit diesem SET-Befehl wird für die Einschaltdauer des Rechners das angegebene Paßwort »paßwort« als Standardpaßwort verwendet, wenn kein oder ein falsches Paßwort für den Zugriff auf Dateien benutzt wird. Auf alle Dateien, die mit demselben Paßwort wie das Standardpaßwort geschützt sind, kann dann zugegriffen werden.

Einer Datei ohne Paßwortschutz, aber mit der Option »PROTECT=ON«, wird dieses Standardpaßwort zugewiesen, so daß auch auf diese zugegriffen werden kann.

Beispiel

```
A> SET [DEFAULT=SCHUTZ]
```

Standardpaßwort ist SCHUTZ.

6.24.9 Setzen der Option für Datums/Zeit-Marken

Format: **SET [CREATE=ON]**
SET [ACCESS=ON]
SET [UPDATE=ON]

Beschreibung

Mit diesen SET-Optionen wird es ermöglicht, Datums/Zeit-Markierungen über die Erstellung (CREATE), Aktualisierung (UPDATE) und den letzten Zugriff (ACCESS) von Diskettendateien zu speichern.

Um in ein Disketten-Inhaltsverzeichnis Datums/Zeit-Markierungen aufzeichnen zu können, muß diese Diskette zunächst mit dem INITDIR-Befehl vorbereitet werden. Von den drei möglichen Datums/Zeitmarkierungen können nur jeweils zwei gleichzeitig aktiviert sein, da sich die CREATE- (Erstellen der Datei) und die ACCESS-Option (Zugriff) gegenseitig ausschließen, d.h., die zuletzt gesetzte Option ist aktiviert.

Eine Ausgabe einer Datei mit einem DIR-Befehl stellt keinen Zugriff im Sinne dieser SET-Option dar.

Beispiele

```
A> SET [ACCESS=ON]
A> DIR [FULL]

Name      Byte  Recs  Attributes  Prot       Update          Access

DREI.TEX   9K    71      Dir RW    None   09/22/85 16:30
EINS.TEX  12K    95      Dir RW    None   09/20/85 10:45
ZWEI.TEX  10K    76      Dir RW    None   09/21/85 12:10


A> SET [CREATE=ON, UPDATE=ON]
A> DIR [FULL]

Name      Byte  Recs  Attributes  Prot       Update          Access

DREI.TEX   9K    71      Dir RW    None   08/22/85 16:30   09/20/85 15:10
EINS.TEX  12K    95      Dir RW    None   08/20/85 10:45   09/20/85  9:36
ZWEI.TEX  10K    76      Dir RW    None   08/21/85 12:10   09/21/85 10:15
```

Tabelle 6.20: SET-Optionen über Datums/Zeit-Markierungen

Option	Bedeutung
CREATE=ON	Die Dateien werden mit dem Zeitpunkt der Erstellung der Datei im Disketten-Inhaltsverzeichnis markiert, wenn diese Option vorher aktiviert wurde.
ACCESS=ON	Im Disketten-Inhaltsverzeichnis wird der Zeitpunkt des letzten Zugriffs auf die jeweilige Datei festgehalten, wenn es keine schreibgeschützte Datei, also [RW], ist.
UPDATE=ON	Der Zeitpunkt einer Aktualisierung einer Datei, d.h. der letzten Änderung, wird bei aktivierter UPDATE-Option aufgezeichnet.

6.25 SETDEF

Format: **SETDEF d:,d:,d:,d: [TEMPORARY=d:] [ORDER (typ,typ)]**
SETDEF [o]
mit o für DISPLAY oder NO DISPLAY und
für PAGE oder NO PAGE.

Beschreibung

Mit dem SETDEF-Befehl können

- die Disketten-Suchreihenfolge,
- das temporäre Laufwerk für die Ablage von Zwischendateien und
- die Dateitypen-Suchreihenfolge

für das Laden von Programmen und für die Ausführung von SUBMIT-Befehlen (SUB-Dateien) festgelegt werden.

Weiterhin gibt es die Möglichkeit,

- mit der DISPLAY-Funktion (System-Anzeige-Modus) Name und Quelle der geladenen Dateien bzw.
- mit der PAGE-Funktion (System-Seiten-Modus) die Bildschirmausgabe seitenweise anzuzeigen.

6.25.1 Anzeige der bestehenden Systemeinstellungen

Format: **SETDEF**

Beschreibung

Es werden die Disketten-Suchreihenfolge, das temporäre Laufwerk und die Dateitypen-Suchreihenfolge ausgegeben.

Beispiel

```
A> SETDEF
Drive Search Path:
1st Drive           - Default
Search Order        - COM
Temporary Drive     - Default
Console Page Mode   - On
Program Name Display - Off
```

6.25.2 Laufwerk- bzw. Disketten-Suchreihenfolge

Format: **SETDEF d:, d:, d:, d:**

Beschreibung

Die Suchreihenfolge der Laufwerke (Disketten) für COM- und SUB-Dateien wird festgelegt. Mit einem Stern »*« für »d« wird das jeweilige aktuelle Laufwerk bzw. Bezugslaufwerk in die Suchreihenfolge eingefügt.

Beispiel

```
A> SETDEF C:,*
```

Die Suchreihenfolge für Dateien ist erst das Laufwerk C und dann das Bezugslaufwerk.

6.25.3 Laufwerk für Zwischendateien

Format: **SETDEF [TEMPORARY=d:]**

Beschreibung

Alle Zwischendateien werden nicht auf dem (aktuellen) Bezugslaufwerk, sondern auf dem mit SETDEF angegebenen Laufwerk »d« abgelegt.

Beispiel

```
A> SETDEF [TEMPORARY=C:]
```

Zwischendateien nur in das Laufwerk C.

6.25.4 Dateitypen-Suchreihenfolge

Format: **SETDEF [ORDER=(typ,typ)]**
für »typ« nur COM oder SUB

Beschreibung

Für die Suchreihenfolge von Dateien kann die Reihenfolge der Dateitypen festgelegt werden. Als Dateityp können nur COM und SUB eingesetzt werden. Nach einem CP/M-Start werden nur COM-Dateien gesucht.

Beispiel

```
A> SETDEF [ORDER=(SUB,COM)]
```

Erst werden SUB-, dann COM-Dateien gesucht.

6.25.5 System-Anzeigemodus

Format: **SETDEF [o]**
mit o für DISPLAY oder NO DISPLAY

Beschreibung

Bei eingeschaltetem Anzeigemodus (mit DISPLAY) gibt CP/M die folgenden Informationen zu einer Programmdatei aus, bevor sie geladen wird:

```
Laufwerk: Dateiname.Typ (Benutzerbereich n)
```

Der voreingestellte Normalfall ist der ausgeschaltete Anzeigemodus (NO DISPLAY).

Beispiel

```
A> SETDEF [DISPLAY]
A> PIP
A:PIP COM                          PIP.COM auf dem Laufwerk A
CP/M 3 PIP VERSION 3.0
*
4A> PIP
A: PIP COM (User 0)                PIP.COM im Benutzerbereich 0 auf dem Laufwerk A.
CP/M 3 VERSION 3.0
*
```

6.25.6 System-Seitenmodus

Format: **SETDEF [o]**
mit o für PAGE oder NO PAGE

Beschreibung

Bei aktiviertem Seitenmodus (mit PAGE) halten CP/M3-Dienstprogramme an, sobald eine komplette Bildschirmseite der Konsole mit Informationen ausgegeben wurde. Nach dem Drücken einer beliebigen Taste laufen die Dienstprogramme weiter. Der voreingestellte Normalfall ist der eingeschaltete System-Seitenmodus.

Beispiel

```
A> SETDEF [NO PAGE]
```

System-Seitenmodus ausgeschaltet; nach Ausgabe einer kompletten Bildschirmseite scrollt der Bildschirminhalt fortlaufend nach oben.

6.26 SHOW

Format: **SHOW d: [o]**
mit o für SPACE, LABEL, USERS, DIR und DRIVE.

Beschreibung

Mit dem SHOW-Befehl können folgende Informationen über Disketten und Laufwerke ausgegeben werden:

- Zugriffsmodus und noch verfügbare freie Diskettenkapazität,
- Diskettenname (Label),
- aktiver Benutzerbereich,
- Anzahl der Dateien in den einzelnen Benutzerbereichen auf der Diskette,
- Anzahl der noch verfügbaren freien Einträge im Disketten-Inhaltsverzeichnis (directory) und
- Laufwerkseigenschaften.

6.26.1 Zugriffsmodus und noch verfügbare Diskettenkapazität

Format: **SHOW d: [SPACE]**

Beschreibung

Mit diesem Befehl werden das Laufwerk, der Zugriffsmodus (RW = Lese/Schreibzugriff und RO = Schreibschutz) für das Laufwerk und die noch verfügbare freie Diskettenkapazität in Kilobyte ausgegeben. Der SHOW-Befehl ohne weitere Angaben gibt Informationen zu allen im System angemeldeten Laufwerken aus.

Beispiele

```
A> SHOW
A:RW,Space                132 k
B:RO,Space                  4 k

A> SHOW B:
B:RO,Space                  4 k

A> SHOW B: [SPACE]
B:RO,Space                  4 k
```

6.26.2 Diskettenname (Label)

Format: **SHOW d: [LABEL]**

Beschreibung

Mit diesem SHOW-Befehl werden Angaben über Diskettennamen (Label) ausgegeben.

Beispiel

```
A> SHOW B:[LABEL]

Label for Drive b:

Directory   Passwds   Stamp    Stamp
Label       Regd     Access   Update     Label Created      Label Updated

NAME.EIN     on        on       on       08/20/85 10:35     09/21/85 21:06
```

Die einzelnen Spalten zeigen von links nach rechts folgendes an:

- Diskettenname des Disketten-Inhaltsverzeichnisses,
- Paßwortschutz der Diskette aktiviert,
- Zeitmarkenmodus für die Erstellung und die letzte Aktualisierung
- aktiviert,
- Datums-/Zeit-Markierung der Erstellung des Diskettennamens und
- Datums-/Zeit-Markierung der letzten Änderung des Diskettennamens.

6.26.3 Benutzerbereiche der Diskette

Format: **SHOW d: [USERS]**

Beschreibung

Mit diesem Befehl werden der aktuelle Benutzerbereich sowie sämtliche Benutzerbereiche auf der Diskette im angesprochenen Laufwerk und die Anzahl der den Bereichen zugeordneten Dateien angezeigt.

Beispiel

```
A> SHOW [USERS]

A: Active User:          0
A: Active Files:         0        2        3
A: # of files:          24       12        6

A: Number of time/date directory entries:    32
A: Number of free directory entries:         50
```

6.26.4 Anzahl der freien Einträge im Disketten-Inhaltsverzeichnis

Format: **SHOW d: [DIR]**

Beschreibung

Es wird die Anzahl der freien Einträge des Disketten-Inhaltsverzeichnisses (directory) für das angegebene Laufwerk angezeigt.

Beispiel

```
A> SHOW C: [DIR]
C: Number of free directory entries:  50
```

6.26.5 Eigenschaften eines Laufwerks

Format: **SHOW d: [DRIVE]**

Beschreibung:

Mit diesem SHOW-Befehl werden die Eigenschaften des angegebenen Laufwerks angegeben.

Beispiel

```
A> SHOW [DRIVE]
A:Drive Characteristics                  Laufwerkeigenschaften
2,720:128 Byte Record Capacity           Aufzeichnungseinheiten zu je 128 Byte
340:Kilobyte Drive Capacity              Diskettenkapazität in Kbyte
128:32 Byte Directory Entries            Maximale DIR-Einträge zu je 32 Byte
128:Checked Directory Entries            Max. DIR-Einträge mit Schreibschutz (RO)
256:Records/Directory Entry              Aufzeichnungseinheit pro DIR-Eintrag
16:Records/Block                         Aufzeichnungseinheit pro Block
8:Sectors/Track                          Sektoren pro Diskettenspur
0:Reserved Tracks                        reservierte Spuren (für das Betriebssystem)
1024:Bytes/Physical Record               Bytes pro Aufzeichnungseinheit
```

6.27 SID

Format: **SID pd,sd oder SID *,sd SID hd**

Es bedeuten:

pd	eine Programm-Dateibezeichnung (zu testendes Programm)
sd	eine Symbol-Dateibezeichnung. Wird z.B. vom Assembler RMAC erzeugt. Format: »d:dateiname.SYM«
hd	eine Hilfsprogramm-Dateibezeichnung. Format: d:HIST.UTL d:TRACE.UTL

Beschreibung

Das SID-Dienstprogramm (symbolic instruction debugger) ermöglicht die Fehlersuche und das Testen von Programmen, die für den Mikroprozessor 8080 entwickelt wurden.

SID gestattet es, Programme schrittweise mit vollständiger Ablaufsanzeige auszuführen, Programme an jedem Punkt anzuhalten, Programme symbolisch zu disassemblieren, d.h., gewählte Unterprogrammnamen und Variablendefinitionen werden mit angezeigt, Programme zu assemblieren sowie Speicher und Registerinhalte anzuzeigen und zu verändern. Um all dies ausführen zu können, gibt es eine Reihe von SID-Kommandos, die nachfolgend zusammengefaßt sind:

A	Programmeingabe mit Hilfe des Assemblers
C	Aufruf eines Unterprogramms
D	Ausgabe eines Speicherbereichs in HEX-Zahlen und ASCII-Zeichen
F	Speicherbereiche mit einem konstanten Wert füllen
G	Das untersuchte Programm ausführen
H	Hexadezimale Arithmetik
I	Datei-Control-Block setzen
L	Speicherbereich disassemblieren
M	Speicherbereiche verschieben
P	Setzen, Rücksetzen oder Anzeigen von Haltepunkten
R	Programm- oder Symboldatei von Diskette laden
S	Einzelne Speicherstellen anzeigen und verändern
T	Schrittweise Programmausführung mit Anzeige der Registerinhalte
U	Schrittweise Programmausführung ohne Anzeige der Registerinhalte
X	Registerinhalte anzeigen und verändern.

Die aufgelisteten Kommandos werden in Tabelle 6.21 näher erläutert.

6.27.1 Laden und Starten von SID

SID kann auf zwei Arten geladen und gestartet werden.

a) Ohne Angabe einer Dateibezeichnung

Mit dem Befehl

```
A> SID
```

wird nur SID ohne ein zu testendes Programm in den Speicher geladen. Nach dem Ladevorgang erscheint die SID-Bereitschaftsmeldung:

```
#
```

SID wartet jetzt auf die Eingabe eines Kommandos. Mit

```
E pd,sd
```

können ein zu testendes Programm »pd« und wahlweise eine Symboltabelle »sd« geladen und danach bearbeitet werden. Erst nach dem Laden eines Programms können die Kommandos G, T und U verwendet werden.

b) Mit Angabe einer Dateibezeichnung

Mit dem Befehl

```
A> SID pd,sd
```

wird zunächst SID und dann das zu testende Programm »pd« und wahlweise eine Symboltabelle »sd« in den Arbeitsspeicher geladen. Nach dem Ladevorgang meldet sich SID mit der Bereitschaftsmeldung # und wartet auf die Eingabe eines Kommandos.

Wird die Programmdateibezeichnung »pd« ohne Dateierweiterung eingegeben, wird automatisch von SID der Dateityp COM eingesetzt. Ähnliches gilt für die wahlweise hinzugeladene Symboldatei »sd«, die eine Tabelle der im zu testenden Programm verwendeten Unterprogrammnamen und Variablenbezeichnungen enthält. Gibt man keine Dateierweiterung beim Aufruf der Symboldatei an, so sucht SID automatisch nach einer Datei mit dem Typ SYM.

6.27.2 SID-Bildschirm-Steuerzeichen

CTRL-C	Abbruch des Arbeitens unter SID (CP/M-Warmstart)
CTRL-S	Bildschirmausgabe anhalten/weiterführen
CTRL-X	aktuelle Eingabezeile löschen
CTRL-P	Druckerausgabe ein-/ausschalten
beliebige Taste	Abbruch einer überlangen Bildschirmausgabe

6.27.3 Konstanten

SID verarbeitet drei verschiedene Arten von Konstanten, deren Anwendung im folgenden beschrieben wird. Nach dem Laden von SID wird die Eingabe von Hexadezimalzahlen erwartet. Die Zahlen zur Basis 16 sind die Ziffern 0 bis 9 sowie die Buchstaben A bis F.

a) Hexadezimalkonstanten

Eine absolute Hexadezimalzahl für SID besteht aus einer oder mehreren hexadezimalen Ziffern, wobei die am weitesten links stehende Ziffer die höchstwertige ist. Werden mehr als vier Ziffern eingegeben, so akzeptiert SID davon nur die rechten vier.

Im folgenden einige Beispiele für die Eingabe gültiger Hexadezimalzahlen:

HEX-Eingabe	Dezimalzahl	SID-Hex-Zahl
1	1	0001
abcd	43981	ABCD
10000	65536	0000
38001	229377	8001

b) Dezimalkonstanten

Es ist möglich, neben der Standardeingabe von Hexadezimalwerten unter SID auch Dezimalzahlen zu verwenden. Hierzu muß der Zahleneingabe ein #-Zeichen vorangestellt werden. Der gültige Zahlenbereich erstreckt sich wie bei Hexadezimalzahlen von 0 bis 65535 (FFFFH).

c) Zeichenkettenkonstanten

SID akzeptiert neben Zahlen auch die Eingabe von Zeichenketten, wobei zur weiteren Verarbeitung eine Umwandlung in die zugehörige ASCII-Zahl erfolgt. Zum Beispiel entspricht A=41H, a=61H, B=42H, C=43H und Leerzeichen=20H.

Damit SID eine Zeichenkette erkennt, muß diese von Apostrophzeichen eingeschlossen werden. Die resultierende Hexadezimalzahl ist auf vier Ziffern beschränkt.

Eingegebene Zeichenkette	Hexadezimalwert
'A'	0041H
'ABC'	4243H
'aA'	6141H
' '	2020H

6.27.4 Symbolische Ausdrücke

Bearbeitet man große Maschinenprogramme, ist es oft schwierig, die verschiedenen Code- (Hauptprogramm, Unterprogramme) und Datenbereiche zu unterscheiden. SID unterstützt dafür den Anwender durch die Möglichkeit, Symbole für Adressen und Variablen zu verwenden. Einige Compiler und Assembler, z.B. RMAC, erzeugen von sich aus Symboltabellen, die dann von SID verarbeitet werden können. Durch ein einzelnes Symbol kann auf drei verschiedene Arten auf Adressen und Daten zugegriffen werden:

- auf den dem Symbol zugewiesenen Adreßwert,
  Schreibweise: .SYMBOLNAME.

- auf den Inhalt der von .SYMBOLNAME adressierten Speicherstelle (8-Bit-Wert),
  Schreibweise: = SYMBOLNAME.

- auf die Inhalte der über .SYMBOLNAME und .SYMBOLNAME+1 adressierten Speicherzellen (16-Bit-Wert),
  Schreibweise: @ SYMBOLNAME.

Die Wirkung dieser drei Arten des symbolischen Zugriffs soll an einigen Beispielen erläutert werden. Die Symboltabelle TEST.SYM enthält folgende Einträge:

0100 BEGIN 0102 ENDE

Weiterhin sind in den Speicherzellen ab Adresse 100H die nachstehenden Werte enthalten:

0100H : 02
0101H : 3E
0102H : 4D
0103H : 22.

Um die folgende Tabelle besser verstehen zu können, muß an dieser Stelle auf eine Eigenheit der 8080-CPU eingegangen werden.

16-Bit-Werte, also zum Beispiel eine Speicheradresse, werden so abgelegt, daß zuerst in eine Speicherstelle das niederwertige Byte geschrieben wird und in die Speicherstelle +1 das höherwertige Byte.

SID gleicht diese ungewöhnliche Art der Speicherung aus, indem beim Zugriff auf einen 16-Bit-Wert über ein Symbol die Bytes des Ergebnisses gedreht werden.

symbolischer Zugriff	hexadezimales Ergebnis
. BEGIN	0100
. ENDE	0102
@ BEGIN	3E02
@ ENDE	224D
= BEGIN	0002
= ENDE	004D

6.27.5 Symbolische Operatoren

Zahlen, Zeichenketten und symbolische Ausdrücke können durch die Operatoren »+« oder »-« miteinander kombiniert werden. Man muß dabei beachten, daß eine Sequenz von Zeichen und Symbolen nicht durch Leerzeichen unterbrochen wird.

Beispiele

```
DF00+#128
=ENDE+5
.BEGIN-10
```

SID rechnet die Eingabe wie folgt um:

```
DF00+80 = DF80
=ENDE+5 = 4D+5 = 52
.BEGIN-10 = 100-10 = 90
```

Die so bestimmten Adreß- bzw. Datenwerte können dann als Eingabe für SID-Kommandos verwendet werden.

6.27.6 SID-Kommandos

In Tabelle 6.21 sind die SID-Kommandos mit den entsprechenden Argumenten (Parametern und Optionen) zusammengestellt. Der eigentliche Kommandobuchstabe ist groß geschrieben. Die klein geschriebenen Parameter und Optionen folgen direkt den Kommandobuchstaben. Optionen sind zusätzlich eingeklammert. Die Argumente der SID-Kommandos können durch die oben dargestellten symbolischen Ausdrücke ersetzt werden. Werden zwei symbolische Ausdrücke benötigt, berechnet SID den zweiten aus dem ersten, wenn die oben beschriebenen Operatoren »+« und »-« verwendet werden.

Tabelle 6.21: SID-Kommandos

Kommando	Bedeutung
As	Eingabe von Anweisungen in Assemblersprache (assemble), s bedeutet die Anfangsadresse.
Cs {b{,d}}	Aufruf des Speicherinhalts an der Adresse s (call). b steht für den Wert des Registerpaares BC, d für den des Registerpaares DE.
D{W} {s}{,f}	Anzeigen des Speicherinhalts in hexadezimaler und ASCII-Form (display). Mit DW ist es ein 16-Bit-Wort. s ist die Anfangs- und f die Endadresse.
Epd{,sd}	Ein Programm pd und eine Symboltabelle sd zur Ausführung laden (load).
E* sd	Laden einer Symboltabellen-Datei sd.
Fs,f,d	Füllen des Speichers mit einem konstanten Wert (fill). s ist die Anfangsadresse, f die Endadresse und d ein 8-Bit-Datenwert.
G{p}{,a{,b}}	Starten der Programmausführung (go). p ist die Startadresse; a ist der erste und b ein möglicher zweiter Unterbrechungspunkt. G0 erzeugt einen CP/M-Warmstart.
Ha	Anzeigen des Hexadezimal-, Dezimal- und ASCII-Wertes der Hexadezimalzahl a (hex).
H.a	Umwandeln des symbolischen Ausdrucks a in einen Zahlenwert und ein Zeichen.
Ha,b	Summe und Differenz der beiden symbolischen Ausdrücke a und b.
I datei	Eingabe einer Datei »datei« in den Dateikontrollblock FCB.
L{s}{,f}	Auflisten des Speicherinhalts als mnemonische 8080-Anweisungen von der Anfangsadresse s bis zur Endadresse f (list).

Kommando	Bedeutung
Ms,h,d	Kopieren eines Speicherabschnitts (move). s ist die Anfangsadresse, h die Endadresse und d die Zieladresse (Anfangsadresse) der Kopie.
P{p{,c}}	Setzen, Rücksetzen und Anzeigen von Unterbrechungspunkten (pass). p ist die Adresse des Unterbrechungspunktes und c der Startwert eines mitlaufenden Ablaufzählers.
R datei{,d}	Laden einer Datei »datei« in den Speicher (read). Wahlweise kann die Datei mit einem Versatz d geladen werden.
S{W}s	Eingabe von Speicherinhalten an der Adresse s (set). Mit SW ist es ein 16-Bit-Wort.
T{n{,c}}	Schrittweiser Programmablauf (trace). n ist die Anzahl der Programmschritte und c die Anfangsadresse eines Dienstprogramms (mit dem Dateityp UTL). Wichtig: Im Trace-Modus werden BDOS-Funktionen von SID ohne Zustandsanzeige ausgeführt.
TW{n{,c}}	Schrittweiser Ablauf des Hauptprogramms, aber normaler Ablauf von Unterprogrammen (trace without call). n ist die Anzahl der Programmschritte und c die Anfangsadresse eines Dienstprogramms (mit dem Dateityp UTL).
U{W}{n{,c}}	Anzeige des Programmablaufs, jedoch kein schrittweiser Ablauf (untrace). n ist die Anzahl der Programmschritte und c die Anfangsadresse eines Dienstprogramms. Mit UW werden Unterprogramme nicht angezeigt.
V	Anzeige des Inhalts (value) des nächsten verfügbaren Speicherplatzes (NEXT), des nächsten Speicherplatzes nach der längsten eingelesenen Datei (MSZE), des aktuellen Standes des Programmzählers (PC) und der Endadresse des verfügbaren Speichers (END).
W datei{,s,f}	Schreiben des Inhalts eines zusammenhängenden Speicherbereiches in die Datei mit der Dateibezeichnung »datei« (write). s ist die Anfangsadresse und f die Endadresse.

Kommando	Bedeutung
X{f} {r}	Prüfen und Ändern des Prozessor-(CPU)-Zustands (examine). f ist eines der Statusregister-Bits C, E, I, M oder Z; r eines der Register A, B, D, H, P oder S.

Beispiele

```
A> SID
#
```
Laden von SID.
SID-Bereitschaftsmeldung zur Eingabe von SID-Kommandos.

```
A> B:SID BEISPIEL.HEX
NEXT MSZE  PC   END
nnnn mmmm pppp eeee
```
Laden von SID und BEISPIEL.HEX von Laufwerk B und folgender SID-Anzeige:
nnnn ist eine hexadezimale Adresse des nächsten freien Speicherplatzes, mmmm der nächste Speicherplatz nach dem längsten zugeladenen Programm. (Nach dem SID-Aufruf ist nnnn=mmmm). pppp ist der Hexadezimalwert des Programmzählers, eeee die Hexadezimaladresse des logischen TPA-Endes.

```
#
#DFE00+#128,+5
```
Nach der SID-Bereitschaftsmeldung # folgt das SID-Kommando D, mit dem die Werte angezeigt werden, die sich im Speicher von der Anfangsadresse FE80 (FE00+#128≙FE00+80) bis zur Endadresse FE85 befinden.

6.27.7 SID-Dienstprogramme

Die SID-Dienstprogramme HIST.UTL und TRACE.UTL sind Zusatzprogramme, die zusammen mit SID zusätzliche Testhilfe ermöglichen. Genauere Informationen über deren Wirkungsweise sollte man der Spezialliteratur entnehmen.

a) Laden der SID-Dienstprogramme

HIST.UTL und TRACE.UTL werden direkt mit dem SID-Befehl als Parameter eingegeben:

```
A> SID dateiname.UTL
```
mit »dateiname« für HIST und TRACE.

b) Histogramm

Mit HIST wird ein Histogramm (Balkendiagramm, Häufigkeitstabelle) erstellt, aus dem die relative Häufigkeit der Code-Ausführung innerhalb ausgewählter Teile des zu testenden Programms zu ersehen ist. Dies ist für sehr häufig ausgeführte Code-Abschnitte interessant.

Nach dem Laden meldet sich HIST mit

```
TYPE HISTOGRAM BOUNDS
aaaa,bbbb
```

Durch Eingabe der Grenzen von aaaa bis bbbb kann ein Schaubild zwischen diesen Punkten angegeben werden. Die Daten für dieses Schaubild werden mit den SID-Kommandos U oder T gesammelt und dann auf dem Bildschirm ausgegeben.

c) Rückverfolgung

Mit TRACE ist eine Rückverfolgung der Anweisungen möglich, die zu einem bestimmten Unterbrechungspunkt im Testprogramm geführt haben. Es können die Adressen von bis zu 256 Anweisungen zwischen Unterbrechungspunkten mit den SID-Befehlen U und T erfaßt werden.

6.28 SUBMIT

Format: **SUBMIT datei a ... a**

Es bedeuten:

datei	eine Dateibezeichnung im Format »d:dateiname.SUB;paßwort«
a	ein Argument

Beschreibung

Mit dem SUBMIT-Befehl können eine Gruppe oder Stapel von CP/M-Befehlen ausgeführt werden, die in einer SUB-Datei mit dem Dateityp SUB abgelegt sind; d.h. mit einem einzigen SUBMIT-Befehl kann eine beliebige Anzahl von CP/M-Befehlen nacheinander so ausgeführt werden, als ob jeder einzelne Befehl einzeln über die Tastatur eingegeben würde.

Normalerweise wird ein CP/M-Befehl zeilenweise eingegeben. Bei mehrmaliger Eingabe derselben Befehlsfolge ist es einfacher, die Befehle mit SUBMIT in einer Gruppe zusammen aufzurufen.

a) SUB-Datei

Diese Folge von CP/M-Befehlen wird in eine Datei mit dem Dateityp SUB in der gewünschten Reihenfolge eingegeben. Wenn nun diese SUB-Datei mit dem SUBMIT-Befehl aufgerufen wird, gibt SUBMIT den Inhalt jeder Zeile dieser SUB-Datei als CP/M-Befehl an CP/M3 weiter, so als würde jeder Befehl einzeln über die Tastatur eingetippt werden.

b) Erstellen einer SUB-Datei

Die SUB-Datei kann z.B. mit dem ED-Dienstprogramm erstellt werden. Sie kann normale CP/M-Befehle, geschachtelte SUBMIT-Befehle und Eingabedaten für CP/M-Befehle und für ein Programm enthalten.

c) Argumente einer SUB-Datei

Für die Ausführung von SUB-Dateien können zusätzlich Argumente in der SUBMIT-Befehlszeile eingelesen werden. Jedes eingegebene Argument wird als Parameter in der SUB-Datei gekennzeichnet. Diese Parameter dienen als Platzhalter, d.h., sie übergeben das eingegebene Argument an die Stelle, an der der Parameter in der Datei steht. Insgesamt sind neun Parameter in einer SUB-Datei möglich; sie werden in der eingegebenen Reihenfolge der Argumente mit $1, $2 bis $9 gekennzeichnet. Das erste Argument wird also in der Datei durch $1, das zweite durch $2 usw. bis hin zum neunten durch $9 ersetzt.

Wenn weniger Argumente in einen SUBMIT-Befehl eingegeben werden als Parameter in der SUB-Datei, werden die übrigen Parameter ignoriert.

Um ein $-Zeichen als Parameter in die SUB-Datei zu übertragen, muß man $$ als Argument eingeben.

Beispiele

a) Die Datei START.SUB enthält folgende CP/M-Befehle:

```
ERA $1.BAK
DIR $1
PIP $1=A:$2.COM
```

Der folgende SUBMIT-Befehl

```
A> SUBMIT START SAM TEX
```

bewirkt, daß in der Datei START.SUB überall $1 durch SAM und $2 durch TEX ersetzt wird, d.h., die von CP/M ausgeführte Befehlsfolge lautet:

```
ERA SAM.BAK
DIR SAM
PIP SAM=A:TEX.COM
```

b) In der Datei AA.SUB lautet eine Zeile:

```
MAC $1 $$$2
```

Der folgende SUBMIT-Befehl

```
A> SUBMIT AA ZZ SZ
```

bewirkt folgenden Befehl:

```
MAC ZZ $SZ
```

6.28.1 Programmeingabe in einer SUB-Datei

Eine SUB-Datei kann Zeilen mit der Eingabe eines Programms enthalten. Vor jeder Programmeingabezeile muß das Zeichen < (kleiner als) stehen.

Wenn das Programm mehr Eingabedaten benötigt als in der SUB-Datei vorhanden sind, müssen die fehlenden Daten über die Tastatur eingegeben werden.

Soll eine SUB-Datei Steuerzeichen enthalten, muß vor dem als Buchstaben eingegebenen Steuerzeichen das Zeichen ^ stehen. Soll nun das Zeichen ^ als Parameter in eine SUB-Datei eingegeben werden, so muß im SUBMIT-Befehl die Kombination ^^ eingegeben werden. (Diese Kombination ^^ wird auf dieselbe Weise in das Zeichen ^ umgewandelt wie $$ in $.)

Beispiel

Eine SUB-Datei lautet:

```
PIP
<B:=*.ASM
<CON:=DUMP.ASM
<
DIR
```

Die drei nach PIP stehenden Zeilen sind Eingabezeilen für den PIP-Befehl. Die dritte Zeile, die nur das Zeichen < enthält, bewirkt ein <CR> bzw. RETURN, und zwar für PIP, d.h. PIP wird beendet. Dadurch kann der letzte Befehl DIR ausgeführt werden.

6.28.2 SUB-Datei

Eine SUB-Datei kann Zeilen mit folgenden Befehlen enthalten:

- einen beliebigen gültigen CP/M3-Befehl,
- einen beliebigen gültigen CP/M3-Befehl mit SUBMIT-Parametern,
- eine beliebige Dateneingabe und
- eine beliebige Programmeingabe mit Parametern ($1 bis $9).

Zeilen mit CP/M-Befehlen dürfen nicht mehr als 128 Zeichen enthalten.

Beispiel

Eine SUB-Datei mit verschiedenen Eingabezeilen lautet:

```
DIR
DIR *.BAK
MAC $1 $$$4
PIP LST:=$1.PRN [T$2 $3 $5]
DIR *.ASM
PIP
<B:=*.ASM
<CON:=DUMP.ASM
<
DIR B:
```

6.28.3 Ausführen eines SUBMIT-Befehls

Format: **SUBMIT**
SUBMIT datei
SUBMIT datei a a

Beschreibung

Wird SUBMIT ohne eine Dateibezeichnung und Argumente aufgerufen, folgt die Aufforderung, eine Dateibezeichnung und Argumente einzugeben.

Beispiele

```
A> SUBMIT
Enter File to Submit: START B TEX

A> SUBMIT START

A> SUBMIT START B TEX          Aufruf START.SUB, mit B für $1 und TEX für $2.
```

6.28.4 Startdatei PROFILE.SUB

Nach jedem Kalt- oder Warmstart des Rechners sucht CP/M3 selbsttätig eine spezielle SUB-Datei, nämlich die Datei PROFILE.SUB, um sie sofort auszuführen. Ist diese Datei nicht vorhanden, arbeitet CP/M3 normal weiter. Ist jedoch die Datei PROFILE.SUB vorhanden, führt CP/M3 die in dieser Datei gespeicherten Befehle sofort aus. Diese Datei erleichtert also die regelmäßige Ausführung einer Gruppe von CP/M-Befehlen vor dem Beginn der normalen Arbeit am Rechner.

Beispiel

Es sollen nach jedem Start mit der Arbeit am Rechner die jeweilige Uhrzeit und das aktuelle Datum eingegeben werden. Hierzu erzeugt man mit ED die Datei PROFILE.SUB mit folgendem Inhalt:

```
DATE SET
```

6.29 TYPE

Format: **TYPE datei [o]**
mit o für PAGE oder NO PAGE.

Beschreibung

Mit dem TYPE-Befehl wird der Inhalt einer ASCII-Zeichen-Datei auf dem Konsolenbildschirm ausgegeben. Mit der normalerweise voreingestellten Option PAGE erfolgt die Konsolenausgabe seitenweise, d.h., die Ausgabe auf dem Bildschirm wird automatisch nach n Zeilen Text angehalten, wobei n gewöhnlich bei CP/M auf 24 Zeilen pro Seite voreingestellt ist. Mit dem DEVICE-Befehl kann n auf einen anderen Wert eingestellt werden. Zum Anzeigen der nächsten Seiten (also die nächsten n Zeilen) muß nur eine beliebige Taste betätigt werden. Mit der Option NO PAGE erfolgt die Bildschirmausgabe der Konsole fortlaufend ohne Unterbrechung. Tabulatorzeichen in der auszugebenden Datei werden bei der Bildschirmanzeige auf jede achte Spalte erweitert.

Die Ausgabe kann mit einigen CTRL-Zeichen gesteuert werden (Tabelle 6.22).

Der TYPE-Befehl kann auf zwei Arten eingegeben werden:

a) Ohne Dateibezeichnung

Mit

```
A> TYPE
```

fordert CP/M die Eingabe des Dateinamens:

```
Enter filename
```

b) Mit Dateibezeichnung

Nach dem Befehl

```
A> TYPE datei
```

wird nach der Ausgabevorbereitung sofort mit der Bildschirmausgabe begonnen.

Ist eine eingegebene Datei nicht im angegebenen Laufwerk, erfolgt die Meldung:

```
No File
```

Die Platzhalter »*« und »?« sind nicht erlaubt.

Beispiele

`A> TYPE PROG.BAS`	Seitenweise Bildschirmausgabe der Datei PROG.BAS von Laufwerk A.
`A> TYPE B:PGM.FOR [NO PAGE]`	Fortlaufende Bildschirmausgabe der Datei PGM.FOR von Laufwerk B.

Tabelle 6.22: TYPE-Steuerzeichen

CTRL-S	Unterbrechen der Bildschirmausgabe.
CTRL-Q	Fortsetzen der Bildschirmausgabe nach einer Unterbrechung mit CTRL-S.
CTRL-P	Gleichzeitige Ausgabe auf Bildschirm und Drucker, wenn CTRL-P vor der Eingabe der TYPE-Befehlszeile eingegeben wird. Nochmaliges CTRL-P beendet die Druckerausgabe.
CTRL-C	Abbruch des TYPE-Befehls und Rücksprung ins CP/M (Warmstart).

6.30 USER

Format: **USER n**
mit n als Nummer des Benutzerbereichs

Beschreibung

Mit dem USER-Befehl wird der aktuelle Benutzerbereich eingestellt. Bei einem CP/M-Kalt- oder -Warmstart ist der Benutzerbereich 0 aktiviert. Mögliche Benutzerbereiche liegen im Bereich von 0 bis 15.

Jede CP/M-Datei ist einem Benutzerbereich zugeordnet. Es kann im allgemeinen nur auf eine Datei im aktuellen Benutzerbereich zugegriffen werden. Aber auf eine Datei im Benutzerbereich 0 mit dem SYS-Attribut kann aus allen Benutzerbereichen zugegriffen werden.

Beispiele:

```
A> USER                 Aufruf des USER-Befehls,
Enter User#:5           neuer Benutzerbereich 5,
5A>                     CP/M-Bereitschaftsmeldung.
A> USER 5               Aufruf des USER-Befehls mit
5A>                     sofortiger Eingabe des neuen Benutzerbereichs 5.
```

6.31 XREF

Format: **XREF d:dateiname o**
mit o für $P

Beschreibung

Mit dem XREF-Befehl kann eine Übersichtsliste von Querverweisen bei der Verwendung von Variablen in einem Programm angefertigt werden.

Hierzu benötigt XREF die von MAC oder RMAC erzeugten PRN- und SYM-Dateien, wobei diese beiden Dateien denselben Dateinamen haben, wie er im XREF-Befehl angegeben wurde. XREF gibt eine Datei mit dem Dateityp XRF aus.

Die Option $P erzeugt eine Druckerausgabe.

Beispiele

`A> XREF B:PROG`	XREF wird aus Laufwerk A geladen, bearbeitet die Dateien PROG.SYM und PROG.PRN aus Laufwerk B und erzeugt die Datei PROG.XRF auch auf Laufwerk B.
`A> XREF B:PGM $P`	Ausgabe der Datei PGM.XRF auf dem Drucker.

7 Arbeiten mit CP/M

An Hand einiger Beispiele soll im folgenden eine Anleitung zum Arbeiten mit CP/M gegeben werden.

Es ist zu erwarten, daß in Kürze einige CP/M-Software für den Commodore 128 zur Verfügung stehen wird. Da mit den neuen Diskettenlaufwerken 1571 und 1570 auch fremde Diskettenformate gelesen werden können, ist das Problem der Übertragung auf den C 128 nicht so einschränkend, wie das bisher bei den meisten auf dem Markt erschienenen CP/M-Rechnern war.

Es soll gezeigt werden, daß vorhandene Software mit relativ einfachen Mitteln auf dem C 128 installiert werden kann. Wichtig ist ein komfortables Textverarbeitungsprogramm zum Erstellen von Programmen und Dateien. Sehr bekannt ist »WordStar«; hier soll die Installation des recht preiswerten Textverarbeitungsprogramms NEVADA-EDIT besprochen werden. Zusätzlich wird die Installation und eine kurze Einführung in das Arbeiten mit den höheren Programmiersprachen BASIC (Interpreter), Fortran und Pascal gezeigt.

Kopieren von Datendisketten

Im allgemeinen wird eine Diskette mit Anwender-Software ohne das CP/M-Betriebssystem und ohne die CP/M-Systemspuren geliefert. Deshalb kann mit dieser Diskette nicht direkt gearbeitet werden, sondern man muß sich erst eine arbeitsfähige CP/M-Diskette mit den Systemspuren erstellen.

Die Diskette mit Anwender-Software muß in einem der vom C 128 lesbaren Formate erstellt sein, z.B. CP/M-C-128, CP/M-C-64, Kaypro II und Osborne (siehe Abschnitt 2.3). Zuerst erstellt man eine neu formatierte Arbeitsdiskette im CP/M-C-128-Format, die nur die CP/M-Systemspuren

enthält (siehe Abschnitt 3.4). In einem nächsten Schritt kopiert man das CP/M-Kopierprogramm PIP.COM mit sich selbst von der CP/M-Systemdiskette auf die neue CP/M-Arbeitsdiskette, z.B. mit

```
A>PIP A:=B:PIP.COM[V]<CR>
```

Danach müssen nun alle Dateien mit dem PIP-Befehl von der Diskette mit der Anwender-Software auf die CP/M-Arbeitsdiskette kopiert werden.

Mit zwei Laufwerken legt man die zu erstellende CP/M-Arbeitsdiskette in das Laufwerk A und die Diskette mit der Anwender-Software in das Laufwerk B und startet den Kopiervorgang mit:

```
A>PIP A:=B:*.*[V]<CR>
```

Mit einem Einzellaufwerk lautet der entsprechende PIP-Befehl:

```
A>PIP E:=A:*.*[V]<CR>.
```

7.1 NEVADA-EDIT

7.1.1 Einleitung

Ein preiswertes Textverarbeitungsprogramm (Texteditor) zum Erstellen von Textdateien und Quellprogrammen ist NEVADA-EDIT von Ellis Computing Inc., San Francisco, USA.

Die Installation von NEVADA-EDIT benötigt einen Rechner (Hardware) mit

- einem 8080/8085/Z80-Mikroprozessor,
- einer Mindestspeichergröße von 32 Kbyte RAM,
- einem Diskettenlaufwerk.

An zusätzlicher Software wird das CP/M-Betriebssystem Version 1.4, 2.2 oder 3.0 benötigt.

Im folgenden soll eine kurze Einführung in die Arbeitsweise und Anwendung dieses WordStar ähnlichen Texteditierprogramms gegeben werden. Zur ausführlichen Information sollte das mitgelieferte Handbuch zu Rate gezogen werden.

7.1.2 NEVADA-EDIT-Diskette

Die NEVADA-EDIT-Diskette enthält das Texteditierprogramm NVEDIT.COM in einer nicht auf dem Rechner angepaßten Version. Zusätzlich sind noch einige Dienstprogramme auf der Diskette enthalten, die entweder unbedingt zum Arbeiten notwendig sind oder bei eventuellen Anpassungen an einen Rechner verwendet werden können.

Die Diskette enthält folgende Dateien:

NVEDIT.COM	der unangepaßte Texteditor zum Erzeugen des lauffähigen Texteditors EDIT.COM.
NVEDIT.ERR	eine Fehlertextdatei, die unbedingt bei der Konfigurierung von EDIT.COM auf der Bezugsdiskette sein muß.

NVEDIT.PRN eine Quelldatei im Assembler-Code, die zum Erstellen besonderer Modifikationen von EDIT.COM dienen kann.

READ-ME.TXT eine evtl. vorhandene Textdatei, die ergänzende Informationen enthält, zu lesen mit dem CP/M-Befehl TYPE READ-ME.TXT<CR>.

EDTKEY.COM eine Datei zum Ändern der Tastaturbelegung.

7.1.3 Konfigurieren von NEVADA-EDIT

Nach dem Laden des CP/M-Betriebssystems muß zuerst das Textverarbeitungsprogramm an den Rechner angepaßt werden. Hierzu lädt man den nicht angepaßten Texteditor mit

```
A>NVEDIT<CR>
```

Die Fehlertextdatei NVEDIT.ERR muß sich auf derselben Diskette wie NVEDIT.COM befinden. Nach der Einschaltmeldung erscheint eine Liste von Konsolen, deren Anpassung direkt vorgenommen werden kann.

```
NVEDIT Version 3.0 configuring
Copyright (C) 1981,1982 Ellis Computing,Inc.

@ ANSI MODE TERMINAL
A ...
:
M LEAR SIEGLER ADM-3A
N LEAR SIEGLER ADM-31
:
W NONE OF THE ABOVE
Type a single letter to select terminal.
<Carriage Return> for more terminals
```

Es ist sowohl die Konsole »Lear-Siegler ADM-3A« als auch »Lear Siegler ADM-31« zur Installation von EDIT.COM auf dem C 128 mit dem CP/M 3.0 geeignet (siehe Kapitel 8). Es muß also entweder der Buchstabe M oder N gedrückt werden.

Anmerkung: Falls ein nicht in der Liste aufgeführter Rechner verwendet werden soll, kann nach Eingaben von »W« die Konfiguration durch Beantworten einer Reihe von Fragen nach den Eigenschaften der Konsole durchgeführt werden. Die Datei NVEDIT.PRN enthält weitere Hilfen zum Konfigurieren und zum Ändern der Tastenbelegung.

Als nächstes wird nach dem Dateityp der zu bearbeitenden Datei gefragt:

```
ENTER FILE TYPE TO BE USED
AS DEFAULT (3 CHARACTERS) XXX
```

Es müssen drei Zeichen oder Leerstellen eingegeben werden, z.B. BAS für BASIC-Programme. Bei der Editierung von Programmen ist es hilfreich, wenn Tabulator-Stops vorhanden sind; voreingestellt sind diese für die Bearbeitung von COBOL-Programmen.

```
DEFAULT TAB STOPS ARE SET FOR COBOL
DO YOU WISH TO CHANGE THEM? (Y/N)? X
```

Will man den Tabulator ändern, wird man nach Eingabe von »Y« durch ein Untermenü geführt. Nach direkter Eingabe von »N« ist die Konfigurierung von EDIT abgeschlossen. Der angepaßte Texteditor wird jetzt als Datei EDIT.COM auf der Bezugsdiskette gespeichert:

```
CONFIGURING OF EDIT.COM IS COMPLETE.
...NOW SAVING IT.
EDIT.COM SAVED ON THE DEFAULT DRIVE.
```

Nachdem man sich nun einen arbeitsfähigen Texteditor EDIT.COM erzeugt hat, werden die Dateien NVEDIT.COM, NVEDIT.ERR und NVEDIT.PRN nicht mehr benötigt und können mit dem ERASE-Befehl von der Arbeitsdiskette gelöscht werden.

Alle Editierfehlermeldungen stehen in der Fehlertextdatei NVEDIT.ERR. Diese englischen Fehlermeldungen können vom Anwender geändert, z.B. ins Deutsche übersetzt werden. Dies kann beispielsweise mit EDIT.COM erfolgen. Danach muß man sich aber eine neue Version des EDIT.COM erzeugen.

Nachdem man sich einen arbeitsfähigen Texteditor erstellt hat, kann man nun zusätzlich mit dem Programm EDTKEY.COM die Tastaturbelegung für spezielle Anwendungen ändern. Nach Aufruf des Programms mit

```
A>EDTKEY<CR>
```

wird man nach einer ausführlichen Einleitung durch Beantworten von Fragen zu den gewünschten Änderungen veranlaßt.

7.1.4 Starten von EDIT

Format: **EDIT quelldatei <CR>**
oder
EDIT quelldatei zieldatei<CR>

Es bedeuten:

quelldatei — eine zu bearbeitende Quelldatei im Format »d:dateiname.dateityp«. Diese Datei wird in den Arbeitsspeicher geladen.

zieldatei — die Datei im Format »d:dateiname.dateityp«, in die die editierte Datei geschrieben wird. Ohne Angabe einer Zieldatei wird die editierte Datei in die Quelldatei zurückgeschrieben.

Beschreibung

Die Quelldatei wird von der angegebenen Diskette in den Arbeitsspeicher geschrieben. Es wird im allgemeinen der gesamte freie Arbeitsspeicher TPA verwendet.

Wenn die bezeichnete Quelldatei nicht existiert, fragt der Editor, ob eine neue Datei mit dem angegebenen Namen erzeugt werden soll:

```
FILE IS NON EXISTENT. CREATE? Y
```

Nach der Bejahung mit »Y« wird eine leere Datei mit dem angegebenen oder voreingestellten Dateityp in den Speicher geladen. Bei Eingabe eines anderen Zeichens als »Y« erfolgt ein Rücksprung ins CP/M.

Nach dem erfolgreichen Laden eines Programms wird der freie Speicherplatz angezeigt:

```
BYTES FREE XXXXX DECIMAL
C/R TO CONTINUE
```

Nach Drücken der RETURN-Taste wird die erste Bildschirmseite der geladenen Datei oder ein leerer Bildschirm bei einer neu erzeugten Datei angezeigt. Der Cursor befindet sich in Zeile 1 und Spalte 1 (Home-Position). Die Datei kann nun bearbeitet bzw. editiert werden.

7.1.5 Arbeiten mit EDIT

a) Speicherbereich

Das Programm EDIT.COM wird von der Speicherstelle 100H in den Arbeitsspeicher TPA geladen. Die Quelldatei wird dann in den Speicherbereich geschrieben, der direkt nach dem Programm EDIT.COM beginnt und bis zum Ende des TPA reicht. Die ursprüngliche Quelldatei muß vollständig in den frei verfügbaren, nachfolgenden Speicher passen. Wenn ein EDIT-Kommando die Größe einer Datei so vergrößert, daß der Text nicht mehr in den Speicherbereich paßt, wird in der Zeile des Cursors die folgende Meldung ausgegeben:

```
FULL - TYPE CONTROL Q
```

Das vorher eingegebene Kommando wird nicht ausgeführt. Nach Eingabe von CTRL-Q sollte entweder die Datei gekürzt oder der Editor durch Eingabe von CTRL-K verlassen werden. Mit CTRL-K wird die Datei auf die Diskette zurückgeschrieben. Mit dem Editor kann nun eine Folgedatei erzeugt oder die zu große Datei in zwei kleinere Dateien geteilt werden, so daß in jeder weitere Zusätze durchgeführt werden. Zum Teilen einer Datei dienen die EDIT-Kommandos CTRL-B:EN (Dateiende setzen) und CTRL-B:ST (Dateianfang setzen).

b) Beenden des Editierens

Es gibt zwei Möglichkeiten, die Editierung einer Datei zu beenden.

- Rückschreiben der Textdatei mit

  `CTRL-K.`

- Bei Angabe einer Zieldatei im EDIT-Befehl wird der Inhalt des Textspeichers in diese Datei zurückgeschrieben. Ohne Angabe einer Zieldatei wird die Quelldatei mit dem editierten Text überschrieben, so daß die alte Quelldatei verlorengeht. In beiden Fällen kann die Datei, in die der Text geschrieben werden soll, noch gewählt werden. Danach wird ins CP/M zurückgesprungen.

- Ohne Rückschreiben der Textdatei

  `CTRL-B:AB<CR>`

- Hiermit wird die Editierung ohne Speichern des Textspeichers in einer Diskettendatei abgebrochen und direkt ins CP/M zurückgesprungen.

c) Editierkommandos

Wie schon erläutert, kann ein Texteditierprogramm (Editor) für zwei Zwecke benutzt werden. Zum einen kann eine neue Datei erzeugt werden und mit dem Editor ein Text, z.B. ein Programm, eingegeben werden; hierbei erlauben die Steuerkommandos die Korrektur von Fehlern oder den Austausch von Wörtern oder Textteilen, ohne daß der restliche Text nochmals eingegeben werden muß. Zum anderen können mit dem Editor bereits existierende Textdateien geändert werden, die früher einmal mit demselben Editor erzeugt wurden.

Tabelle 7.1 zeigt eine Zusammenstellung der Steuerzeichen des Editors EDIT. Diese EDIT-Kommandos werden durch gleichzeitiges Drücken der CONTROL-Taste CTRL und der bezeichneten Taste aufgerufen.

Tabelle 7.1: EDIT-Kommandos

Steuerzeichen	Bedeutung
CTRL-E	Verschieben des Cursors um eine Zeile nach oben.
CTRL-X	Verschieben des Cursors um eine Zeile nach unten.
CTRL-S	Verschieben des Cursors um eine Stelle nach links.
CTRL-D	Verschieben des Cursors um eine Stelle nach rechts.
CTRL-I o. TAB	Verschieben des Cursors zum nächsten Tabulator-Stop.
CTRL-Q	Cursor in die erste Zeile und erste Spalte (Home-Position).
CTRL-A	Cursor in die Bildschirmmitte, erste Spalte.
CTRL-Z	Scrollen des Textes um eine Zeile nach oben.
CTRL-W	Scrollen des Textes um eine Zeile nach unten.
CTRL-C	Scrollen des Textes um eine halbe Bildschirmseite nach oben.
CTRL-R	Scrollen des Textes um eine halbe Bildschirmseite nach unten.
CTRL-F	Suchen einer Zeichenkette (Suchmodus), die nach einem Doppelpunkt eingegeben und mit <CR> abgeschlossen wird.
CTRL-L	Suchen der nächsten Zeichenkette im Suchmodus (nach CTRL-F).

Steuerzeichen	Bedeutung
CTRL-V	Umschalten (Ein bzw. Aus) des Einfügemodus (insert).
CTRL-	n-maliges Wiederholen des darauffolgenden Kommandos, z.B. CTRL- CTRL-n<CR>.
CTRL-G	Löschen eines Zeichens unter dem Cursor.
CTRL-N	Einfügen einer Zeile über dem Cursor.
CTRL-Y	Löschen der Zeile, in der der Cursor steht.
CTRL-\	Teilen der Zeile an der Cursorposition.
CTRL-T	Löschen aller Zeichen von der Cursorposition bis zum Zeilenende.
CTRL-M o. <CR>	Eingabe und Abschluß der Zeile, in der der Cursor steht. Der Cursor wird auf die erste Stelle der nächsten Zeile verschoben.
CTRL-K	Beenden der Editierung und Speichern des Textes in einer Diskettendatei.
CTRL-B	Erweiterung für weitere Kommandos. Die Eingabe der weiteren Befehle erfolgt nach einem auf dem Bildschirm erscheinenden Doppelpunkt. Die Übergabe erfolgt mit <CR>.
:AB	Abbruch der Editierung ohne Speichern des editierten Textes und Rücksprung ins CP/M.
:TA	Setzen der Tabulatorfunktion (für CTRL-U). Die Stelle (Spalte) eines Tabulator-Stops wird mit einem ASCII-Zeichen als Ersatz für den ASCII-Wert festgelegt, wobei nur Zeichen mit Werten größer als 30H = 48 verwendet werden können. Die Spalte berechnet sich aus dem ASCII-Wert minus 48.

Steuerzeichen	Bedeutung
:BC	Setzen des Blockkopierens (für CTRL-U).
:BM	Setzen des Blockverschiebens (für CTRL-U).
:NU	CTRL-U hat keine Wirkung.
:FL	Verschieben des Cursors in die erste Zeile der Datei.
:LL	Verschieben des Cursors in die letzte Zeile der Datei.
:ST	Festlegen des Dateianfangs. Alle über dem Cursor stehenden Zeilen werden gelöscht.
:EN	Festlegen des Dateiendes. Alle hinter dem Cursor stehenden Zeilen werden gelöscht.
:PA R	Suchen und Ersetzen von vorgegebenen Mustern (pattern) bzw. Zeichenketten.
:PA RS	Suchen und selektives Ersetzen von vorgegebenen Mustern (pattern) bzw. Zeichenketten.
:PA D	Suchen und Löschen von vorgegebenen Mustern (pattern) bzw. Zeichenketten.
:PA DS	Suchen und selektives Löschen von vorgegebenen Mustern (pattern) bzw. Zeichenketten.
:IF datei	Einfügen einer externen Datei mit dem Format »d:dateiname.dateityp« in die gerade editierte Datei.
:TS	Setzen oder Löschen von Tabulator-Stops.
CTRL-U	Ausführen der mit CTRL-B gesetzten Kommandos (Tabulator, Blockkopieren und Blockverschieben).

7.2 MBASIC-Interpreter

7.2.1 Einleitung

Eine der komfortabelsten BASIC-Implementierungen für Mikrocomputer ist der Microsoft-BASIC-Interpreter (MBASIC oder BASIC-80), wobei die verwendete BASIC-Programmiersprache mit dem Microsoft-BASIC-Compiler (BASCOM) weitgehend kompatibel ist. Zum Erstellen eines BASIC-Programms verwendet man zweckmäßigerweise den BASIC-Interpreter, um anschließend mit dem BASIC-Compiler eine direkt ablauffähige COM-Datei zu erzeugen. (Anmerkung: Dem Verfasser stand leider der Microsoft-BASIC-Compiler nicht zur Verfügung.)

Die Installation von MBASIC benötigt

- einen Rechner mit einer Mindestspeichergröße von 32 Kbyte, davon 24 Kbyte für das BASIC-80, sowie zusätzlich Speicherplatz für das auszuführende Programm,
- ein Diskettenlaufwerk und
- das CP/M-Betriebssystem.

7.2.2 Laden des MBASIC-Interpreters

Die Diskette mit dem MBASIC-Interpreter enthält neben einigen Testprogrammen die Datei MBASIC.COM. Dies kann man nach dem Laden des CP/M-Betriebssystems durch Ausgabe des Disketten-Inhaltsverzeichnisses mit dem Befehl »DIR« kontrollieren:

MBASIC.COM	die aktuelle Version des MBASIC-Interpreters,
OBASIC.COM	BASIC-80-Version für früher erstellte Programme,
RUNTEST.BAS	ein BASIC-Testprogramm,
RFILE	eine von RUNTEST.BAS erzeugte Datendatei.

Da die MBASIC-Datei eine direkt ausführbare COM-Datei ist, kann sie von CP/M direkt in den Arbeitsspeicher geladen und ausgeführt werden. Alles, was man noch tun muß, ist, dem Rechner den Dateinamen einzugeben:

```
A>MBASIC<CR>
```

Einige Sekunden, nachdem man die RETURN-Taste <CR> gedrückt hat, sollte die Microsoft-Einschaltmeldung mit der Versionsnummer der MBASIC-Version auf dem Bildschirm zu sehen sein:

```
xxxx Bytes Free
Microsoft BASIC Version x.xx
(CP/M-Version)
Copyright 19xx (C) by Microsoft
Created: tt-mm-jj
Ok
```

Dieser Schriftzug endet mit dem Wort »Ok«, das anzeigt, daß CP/M die Kontrolle an den MBASIC-Interpreter übergeben hat und daß man nun im direkten BASIC-Modus ist. Der direkte Modus (auch Command- oder Befehls-Modus) erlaubt das Kommunizieren mit dem Rechner über MBASIC.

Man kann zusätzlich noch einen Satz von Optionen in der Befehlszeile zum Laden des MBASIC-Interpreters eingeben:

Format: **MBASIC datei/F:f/M:m/S:s**

Es bedeuten:

datei	eine CP/M-BASIC-Datei mit der Dateikennung »BAS«.
/F:f	eine Anzahl von »f« Datendateien, die irgendwann während der Ausführung eines BASIC-Programms auf einer Diskette eröffnet werden können. Ohne eine Angabe wird für »f« eine »3« vorgegeben.
/M:m	die höchste Speicherstelle, die vom BASIC-Programm benutzt wird. Ohne eine Angabe für »m« wird der maximal mögliche Speicherplatz vorgegeben.
/S:s	die maximale Datensatzgröße einer Datei mit wahlfreiem Zugriff (random access file). Die vorgegebene Datensatzgröße ist 128 Byte.

Beschreibung

Mit dem Befehl »MBASIC« wird der MBASIC-Interpreter geladen und gestartet. Nach dem Start befindet man sich im MBASIC-Befehlsmodus (direkter Modus). Ein Programm kann in der bei BASIC üblichen Weise eingegeben werden, indem man die BASIC-Anweisungen mit einer vorangestellten Zeilennummer eingibt und die Zeile mit »RETURN« abschließt. Dieses Programm wird mit »RUN« gestartet. Gibt man mit dem MBASIC-Befehl den Dateinamen eines vorhandenen BASIC-Programms ein, so wird dieses Programm geladen und ausgeführt.

Werden Unterprogramme in Assemblersprache vom BASIC-Hauptprogramm aufgerufen, sollte man die Speichergröße kleiner als die unter CP/M maximal mögliche wählen, so daß genügend Speicherplatz für die Assembler-Unterprogramme reserviert ist. Die höchste BASIC-Speicheradresse »m« sollte in jedem Fall unter dem CP/M-Betriebssystem liegen, die an der Adresse 6 und 7 abgelegt ist.

Die Zahlen für »f« (Anzahl der Dateien), »m« (höchste Speicheradresse) und »s« (maximale Datensatzgröße) können als Dezimalzahl (ohne Zusatz), als Oktalzahl (mit dem vorangestellten Zusatz »&O«) und als Hexadezimalzahl (mit dem vorangestellten Zusatz »&H«) eingegeben werden.

Beispiele

`MBASIC PGM.BAS`	Benutzen des maximal möglichen Speichers und Reservieren von 3 Dateien, Laden und Ausführen des BASIC-Programms PGM.BAS.
`MBASIC PGM/F:6`	Benutzen des maximal möglichen Speichers und Reservieren von 6 Dateien, Laden und Ausführen des BASIC-Programms PGM.BAS.
`MBASIC /M:32768`	Benutzen der ersten 32 Kbyte des Speichers und Reservieren von 3 Dateien.
`MBASIC /S:256`	Benutzen des maximal möglichen Speichers und Reservieren von 3 Dateien, die maximale Datensatzlänge ist 256 Byte.
`MBASIC PGM/F:2/M: &H9000`	Benutzen der ersten 36 Kbyte des Speichers und Reservieren von 2 Dateien, Laden und Ausführen von PGM.BAS.

7.2.3 Rücksprung ins CP/M

Um den MBASIC-Interpreter zu verlassen und ins CP/M-Betriebssystem zurückzuspringen, muß der Befehl

```
SYSTEM
```

eingegeben werden. Mit diesem Befehl werden alle Dateien abgeschlossen und ein CP/M-Warmstart erzeugt, ohne daß ein geladenes Programm oder geladene Daten gelöscht werden. Gibt man im BASIC-Modus den Befehl

```
CTRL-C
```

ein, so kehrt man in den Befehlsmodus zurück, jedoch nicht ins CP/M.

7.2.4 Bezeichnung einer Datei

Format: **d:dateiname.dateityp**

Es bedeuten:

d	ein Diskettenlaufwerk
dateiname	ein Dateiname und
dateityp	ein Dateityp

Beschreibung

Die Dateibezeichnungen im MBASIC entsprechen denen im CP/M. Vor einem Dateinamen kann A, B, C oder D zum Ansprechen des entsprechenden Laufwerks hinzugefügt werden. Ohne eine Laufwerkangabe wird das derzeitige Bezugslaufwerk angesprochen. Der Dateiname »dateiname« kann aus maximal 8 Zeichen bestehen. Der Dateityp BAS wird bei den Befehlen »LOAD«, »SAVE«, »MERGE« und »RUN« an den Dateinamen angefügt, wenn keine Dateikennung angegeben wurde.

Beispiele

`RUN"NEUFILE.EXE"`	Laden von der Diskette im Bezugslaufwerk und Ausführen des Programms NEUFILE.EXE.
`RUN"A:NEUFILE.EXE"`	Laden von der Diskette im Laufwerk A und Ausführen des Programms.
`SAVE"NEUFILE"`	Speichern des Programms NEUFILE.BAS auf der Diskette im Bezugslaufwerk.

7.2.5 Anzeigen des Disketten-Inhaltsverzeichnisses

Befindet man sich im direkten MBASIC-Modus, kann man das Disketten-Inhaltsverzeichnis (directory) nicht mehr mit dem CP/M-Befehl »DIR« auflisten; mit diesem Befehl erhält man die Fehlermeldung »Syntax error«. Der DIR-Befehl ist nicht mehr im MBASIC-Befehlssatz enthalten. Der entsprechende MBASIC-Befehl lautet:

Format: **FILES**
oder
FILES"d:dateiname.dateityp"

Beschreibung

Der MBASIC-Befehl »FILES« dient zum Auflisten des Disketten-Inhaltsverzeichnisses. Ohne einen Zusatz nach dem FILES-Befehl werden alle Dateien auf der Bezugsdiskette ausgegeben. Im Dateinamen »dateiname« und in der Dateikennung »dateityp« können die Ersatzzeichen »?« und »*« eingesetzt werden. Das Fragezeichen steht für ein einziges beliebiges Zeichen, der Stern für eine beliebige Kombination mehrerer Zeichen.

Beispiele

`FILES`	Auflisten aller Dateien auf der Bezugsdiskette.
`FILES"*.BAS"`	Auflisten aller Dateien mit dem Dateityp BAS auf der Bezugsdiskette.
`FILES"B:*.*"`	Auflisten aller Dateien auf der Diskette im Laufwerk B.
`FILES"TEST?.BAS"`	Auflisten aller Dateien mit einem 5stelligen Dateinamen, der mit TEST beginnt, und dem Dateityp BAS auf der Bezugsdiskette.

7.2.6 MBASIC-Instruktionen

Tabelle 7.2 zeigt eine Übersicht der BASIC-80-Befehle, Tabelle 7.3 die BASIC-80-Anweisungen und Tabelle 7.4 die BASIC-80-Funktionen. Weitere Informationen über die Anwendung und Syntax sollte man dem Microsoft-Reference-Manual oder der einschlägigen Literatur entnehmen (siehe Literaturverzeichnis, Anhang A).

Tabelle 7.2: MBASIC-Befehle (Befehlsmodus)

Befehl	Beschreibung
AUTO	Automatisches Durchnumerieren von Programmzeilen.
CLEAR	Rücksetzen aller Variablen auf Null und Schließen aller offenen Dateien.
CLOAD	Laden einer Datei von Kassette.
CONT	Fortsetzen eines Programms nach einer Unterbrechung mit »CTRL-C« oder einer STOP- oder END-Anweisung.
CSAVE	Speichern einer Datei auf Kassette.
DELETE	Löschen von Programmzeilen.
EDIT	Festlegen der zu editierenden Programmzeile.
KILL	Löschen einer Diskettendatei.
LIST	Anzeigen der bezeichneten Zeile.
LLIST	Ausdrucken der bezeichneten Zeile auf einem Drucker.
LOAD	Laden einer Datei von einer Diskette in den Speicher.
MERGE	Anfügen eines Programms an ein im Speicher befindliches Programm.
NAME	Umbenennen einer Datei.
NEW	Löschen des Programms und aller Variablen im Speicher.
NULL	Setzen einer Anzahl von Nullen, die am Ende jeder Zeile gedruckt werden sollen.
RENUM	Umnumerieren von Programmzeilen.
RUN	Starten eines Programms.

Befehl	Beschreibung
SAVE	Speichern eines Programms in einer Diskettendatei.
TRON	Einschalten der TRACE-Funktion.
TROFF	Ausschalten der TRACE-Funktion.
WIDTH	Ändern der Zeilenlänge.
CTRL-C	Umschalten vom Programm-Modus in den direkten Modus (Befehlsmodus).

Tabelle 7.3: MBASIC-Anweisungen (statement) für BASIC-Programme

Anweisung	Beschreibung
CALL	Aufruf eines Assembler-Unterprogramms.
CHAIN	Aufruf eines Programms mit Übergabe von Variablen.
COMMON	Kennzeichnen von Variablen für einen CHAIN-Aufruf.
DATA	Speichern von Konstanten.
DEF FN	Definieren einer benutzereigenen Funktion.
DEFINT	Definieren einer Variablen als ganze Zahl.
DEFSNG	Definieren als Zahl mit einfacher Genauigkeit.
DEFDBL	Definieren als Zahl mit doppelter Genauigkeit.
DEFSTR	Definieren als Zeichenkette (string).
DEF USR	Definieren einer Startadresse eines Assembler-Unterprogramms.
DIM	Angabe der maximalen Größe einer Zahlenfeld-Laufzahl.
END	Beenden eines Programms, Schließen aller Dateien und Rücksprung in den Befehlsmodus.
ERASE	Löschen aller Zahlenfelder (array) in einem Programm.
ERR	Fehlercode-Variable.
ERL	Variable mit der Nummer der fehlerhaften Zeile.
ERROR	Fortsetzen des Programms mit der ERROR-Anweisung bei einem Fehler im BASIC-Programmlauf.
FIELD	Zuweisen der Speichergröße für eine Datei mit wahlfreiem Zugriff.

Anweisung	Beschreibung
FOR-NEXT-STEP	Erzeugen einer Schleife.
GET	Lesen von Daten aus einer Diskettendatei.
GOSUB-RETURN	Unterprogramm (subroutine).
GOTO	Unbedingte Sprunganweisung.
IF-THEN-ELSE	Bedingte Sprunganweisung.
IF-GOTO	Bedingte Sprunganweisung.
INPUT	Eingabe über Tastatur.
INPUT#	Eingabe aus einer sequentiellen Diskettendatei.
LET	Einer Variablen einen Wert zuweisen.
LINE INPUT	Eingeben einer vollen Zeile in eine Zeichenkette-Variable.
LINE INPUT#	Einlesen einer vollen Zeile in eine Zeichenkette-Variable von einer sequentiellen Diskettendatei.
LPRINT (USING)	Ausdrucken auf einem Drucker.
LSET	Linksbündiges Justieren eines Textes.
ON ERROR GOTO	Verzweigen bei fehlerhaftem Programmlauf.
ON-GOSUB	Verzweigen zu einem Unterprogramm, abhängig von einer Berechnung.
ON-GOTO	Verzweigen zu einer Zeile, abhängig von einer Berechnung.

Anweisung	Beschreibung
OPEN	Eröffnen einer Diskettendatei.
OPTION BASE	Niedrigste Zahlenfeld-Laufzahl 0 oder 1.
OUT	Senden eines Bytes zu einem Ausgabekanal.
POKE	Schreiben eines Bytes in eine Speicherstelle.
PRINT	Ausgabe auf dem Bildschirm.
PRINT USING	Ausgabe einer Zeichenkette in einem vorgegebenen Format.
PRINT#	Schreiben von Daten in eine sequentielle Datei.
PRINT USING#	Schreiben von formatierten Daten in eine sequentielle Datei.
PUT	Speichern von Daten in eine Diskettendatei mit wahlfreiem Zugriff.
RANDO-MIZE	Zufallsgenerator
READ	Lesen von Daten aus DATA-Anweisungen.
REM	Kommentarzeile im Programm.
RESTORE	Zurücksetzen des Zeigers zum Lesen von DATAs.
RESUME	Wiederstarten des Programms nach einem Fehler.
RETURN	Rücksprunganweisung in einem Unterprogramm.
RSET	Rechtsbündiges Justieren eines Textes.
STOP	Unterbrechen eines Programmlaufs.
SWAP	Austauschen der Werte zweier Variablen.

Anweisung	Beschreibung
WAIT	Unterbrechen des Programmlaufs zum Abfragen eines Eingabekanal-Status.
WHILE-WEND	Ausführen von Anweisungen, bis eine gegebene Bedingung erfüllt ist.
WRITE	Ausgeben von Daten auf die Konsole.
WRITE#	Ausgeben von Daten in eine sequentielle Datei.

Tabelle 7.4: MBASIC-Funktionen

Funktion	Beschreibung
ABS	Absolutwert.
ASC	ASCII-Wert vom ersten Zeichen der Zeichenkette.
ATN	Arcus-Tangens.
CDBL	Gleitkommazahl mit doppelter Genauigkeit.
CHR$	Zeichen aus ASCII-Code.
CINT	Umwandeln in eine Ganzzahl.
COS	Cosinus.
CSNG	Gleitkommazahl mit einfacher Genauigkeit.
CVD	Umwandeln einer 8-Byte-Zeichenkette in eine Gleitkommazahl mit doppelter Genauigkeit.
CVI	Umwandeln einer 2-Byte-Zeichenkette in eine Ganzzahl.
CVS	Umwandeln einer 4-Byte-Zeichenkette in eine Gleitkommazahl mit einfacher Genauigkeit.
EOF	Dateiende-Kennung.
EXP	Exponentialfunktion.
FIX	Umwandeln in eine Ganzzahl.
FRE	Freier Speicherplatz in Bytes.
HEX$	Umwandeln einer Hexadezimalzahl in eine Zeichenkette.
INKEY$	Einlesen eines Zeichens als Zeichenkette.
INP	Lesen eines Bytes von einem Kanal.

Funktion	Beschreibung
INPUT$	Einlesen von Zeichen in eine Zeichenkette.
INSTR	Suchen eines Teiles in einer Zeichenkette.
INT	Umwandeln in eine Ganzzahl.
LEFT$	Linksbündige Zeichen einer Zeichenkette.
LEN	Anzahl der Zeichen einer Zeichenkette.
LOC	Logische Dateinummer in einem OPEN-Befehl.
LOG	Natürlicher Logarithmus.
LPOS	Position des Druckerkopfes.
MID$	Mittlere Zeichen einer Zeichenkette.
MKD$	Umwandeln einer Gleitkommazahl mit doppelter Genauigkeit in eine 8-Byte-Zeichenkette.
MKI$	Umwandeln einer Ganzzahl in eine 2-Byte-Zeichenkette.
MKS$	Umwandeln einer Gleitkommazahl mit einfacher Genauigkeit in eine 4-Byte-Zeichenkette.
OCT$	Umwandeln einer Dezimalzahl in eine Oktalzahl.
PEEK	Inhalt einer Speicherstelle.
POS	Position des Cursors.
RIGHT$	Rechtsbündige Zeichen einer Zeichenkette.
RND	Zufallszahl zwischen 0 und 1.
SGN	Vorzeichen.
SIN	Sinus.

Funktion	Beschreibung
SPACE$	Leerzeichen in einer Zeichenkette.
SPC	Leerzeichen beim Drucken.
SQR	Quadratwurzel.
STR$	Umwandeln einer Zahl in eine Zeichenkette.
STRING$	Mehrmaliges gleiches Zeichen in einer Zeichenkette.
TAB	Leerstellen bis zur angegebenen Position.
TAN	Tangens.
USR	Aufruf eines Assembler-Unterprogramms.
VAL	Numerischer Wert einer Zeichenkette.
VARPTR	Speicheradresse.

7.3 NEVADA-FORTRAN

7.3.1 Einleitung

Eine preiswerte FORTRAN-VI-Implementierung für Mikrocomputer ist der NEVADA-FORTRAN-Compiler von Ellis Computing Inc., San Francisco, USA.

Die Installation von NEVADA-FORTRAN benötigt einen Rechner (Hardware) mit

- einem 8080/8085/Z80-Prozessor,
- einer Mindestspeichergröße von 48 Kbyte und
- einem Diskettenlaufwerk.

An zusätzlicher Software wird neben dem CP/M-Betriebssystem ein Texteditierprogramm (Texteditor) zum Erstellen eigener FORTRAN-Quellprogramme benötigt. Hierzu eignet sich z.B. das CP/M-Editier-Dienstprogramm »ED« oder NEVADA-EDIT.

Im folgenden soll eine kurze Einführung in die Arbeitsweise und Anwendung dieses FORTRAN-Compilers wiedergegeben werden. Für ausführliche Informationen sollte das mitgelieferte Handbuch zu Rate gezogen werden.

7.3.2 NEVADA-FORTRAN-Diskette

Die NEVADA-FORTRAN-Diskette enthält neben dem eigentlichen FORTRAN-Compiler und den notwendigen Dienstprogrammen noch einige Testprogramme. Nach dem Laden des CP/M-Betriebssystems kann man mit »DIR« das Disketten-Inhaltsverzeichnis auflisten. Die Diskette muß unbedingt folgende Dateien enthalten:

FORT.COM	der FORTRAN-Compiler (z.Z. aktuelle Version 3.2-Mod0) zum Erstellen einer Objektdatei (OBJ-Datei) aus einem FORTRAN-Quellprogramm (FOR-Datei).
FRUN.COM	Programm zum Laden und direkten Ausführen des kompilierten Programms (OBJ-Datei) oder zum Erzeugen einer direkt ausführbaren COM-Datei (runtime package).
CONFIG.COM	Programm zum Erzeugen einer Fehlerdatei und zum

	Eingeben vorbestimmter FORT- und FRUN-Parameter.
ERRORS	eine von »CONFIG« benutzte Fehlertextdatei.
ASSM.COM	ein Assemblerprogramm (ähnlich dem CP/M-Dienstprogramm ASM.COM) zum Assemblieren eines Assembler-Quellprogramms (ASM-Datei) in eine Objektdatei (OBJ-Datei). Dieser Assembler muß unbedingt mit FORT.COM auf derselben Diskette sein.
RUNA.COM	ein Programm zum Laden und direkten Ausführen einer Objektdatei oder zum Erzeugen einer direkt ausführbaren COM-Datei.
READ.ME	eine evtl. vorhandene Textdatei, die ergänzende Informationen enthalten kann.

Im allgemeinen wird die NEVADA-FORTRAN-Diskette ohne das CP/M-Betriebssystem geliefert. Von der Original-NEVADA-FORTRAN-Diskette sollte man sich Arbeitsdisketten mit den CP/M-Systemspuren anfertigen.

7.3.3 Konfigurieren des NEVADA-FORTRAN-Systems

Nach dem Laden des CP/M-Betriebssystems sollte man sich die gewünschte Konfiguration des FORTRAN-Systems einstellen. Hierzu lädt man das FORTRAN-Konfigurationsprogramm mit

CONFIG<CR>

Auf dem Bildschirm erscheint die Einschaltmeldung:

```
NEVADA FORTRAN CONFIGURATION PROGRAM (ttmmjj)
```

a) Erzeugen der Fehlerdatei FORT.ERR

Das Programm »CONFIG« liest die Textdatei »ERRORS«, die die Compiler-Fehlermeldungen enthält. Diese Fehlermeldungen kann man mit einem Texteditierprogramm in der Datei »ERRORS« ändern, dabei darf eine Fehlermeldung nur eine Zeile von 80 Zeichen Länge umfassen. Die ersten beiden Zeichen sind die Fehlernummer, nach einer Leerstelle folgt der Fehlertext. Die Fehlerdatei kann auch mit deutschen Kommentaren versehen werden.

Nach der Einschaltmeldung beantwortet man die Frage, ob man eine Fehlerdatei FORT.ERR erzeugen will, mit »Y«. Danach wird gefragt, welches Laufwerk die Datei »ERRORS« enthält und auf welches Laufwerk (auf welche Diskette) die Datei FORT.ERR geschrieben werden soll. Bei einem Einzellaufwerk arbeitet man zweckmäßigerweise nur mit der Bezugsdiskette »A«. Nun wird die Datei FORT.ERR erzeugt und auf die Bezugsdiskette geschrieben. Diese Datei muß man vor einem Kompiliervorgang erzeugen, da sie nicht auf der NEVADA-FORTRAN-Diskette ist.

b) Konfigurieren des NEVADA-FORTRAN-Systems

Das Programm »CONFIG« erlaubt nun, verschiedene Parameterwerte für den Compiler FORT.COM und FRUN.COM festzulegen. Einen neuen Parameterwert gibt man als Dezimalzahl mit anschließendem Drücken der RETURN-Taste ein. Die vorgegebenen Parameterwerte sind in rechteckigen Klammern bei jedem Parameter angegeben. Um die vorgegebenen Parameterwerte zu belassen, drückt man die RETURN-Taste ohne eine Zahleneingabe.

Zuerst muß das Laufwerk, das den Compiler FORT.COM enthält, angegeben werden. Danach wird nach folgenden Parameterwerten gefragt:

- die Größe der Symboltabelle,
- die Größe der Label- bzw. Sprungmarkentabelle,
- die Anzahl der Zahlenfelder (array),
- die maximale Verschachtelungstiefe von DO-Schleifen,
- die maximale Verschachtelungstiefe von IF-THEN-ELSE-Anweisungen,
- die maximale Variablenzahl in einem Ausdruck,
- die maximale Anzahl der Operanden-Code-Stack-Variablen,
- die Anzahl der Operanden-Stack-Variablen und
- das Zeichen, das vor und hinter einer Hexadezimalkonstanten in einer Zeichenkette verwendet wird.

Zusätzlich kann man die Frage, ob die Konsole auch Kleinbuchstaben ausgeben kann, mit »Y« bejahen.

Schließlich können im Programm FRUN.COM auch einige Parameter eingestellt werden. Zuerst muß wieder das Laufwerk, in dem sich die Diskette mit FRUN.COM befindet, angegeben werden, hier wieder die Bezugsdiskette »A«. Dann kann die Methode, mit der FRUN.COM die CP/M-Konsolenein-/-ausgaben durchführt, spezifiziert werden.

Es sind drei Wege möglich:

- direkte BIOS-Aufrufe,
- CP/M-BDOS-Funktion 1 und 2 oder
- CP/M-BDOS-Funktion 6.

Jeder Weg hat seinen Vorteil. Die BDOS-Funktion 1 und 2 erlaubt mit »CTRL-P« eine direkte FORTRAN-Ausgabe auf einen angeschlossenen Drucker. Als nächstes können Sie die Frage, ob die Konsole auch Kleinbuchstaben verarbeiten kann, wieder mit »Y« bejahen. Eine North-Star-Floating-Point-Karte ist nicht im C 128 eingebaut, deshalb wird hier ein »N« eingegeben.

Nun wird die Fehlerdatei FORT.ERR erzeugt und auf die Bezugsdiskette geschrieben. Die Dateien CONFIG.COM und ERRORS werden jetzt nicht mehr benötigt und können deshalb auf der Arbeitsdiskette gelöscht werden.

7.3.4 Kompilieren mit dem NEVADA-FORTRAN-Compiler

Nachdem man sich ein FORTRAN-Quellprogramm mit der Dateikennung »FOR« erstellt hat, kann man dieses mit dem FORTRAN-Compiler FORT.COM kompilieren.

Format: **FORT d:dateiname.PPP $o**

Es bedeuten:

d	das Diskettenlaufwerk, das die Diskette mit der Datei »dateiname.FOR« (FORTRAN-Quellprogramm) enthält.
dateiname	das zu kompilierende FORTRAN-Quellprogramm »dateiname.FOR«.
PPP	Parameter (bis zu 3 Zeichen) zum Kennzeichnen der Laufwerke für die Zieldateien.
$o	Compiler-Optionen.

Beschreibung

Der Befehl »FORT« lädt den FORTRAN-Compiler FORT.COM und kompiliert das FORTRAN-Programm »dateiname.FOR« wie folgt:

- Das Programm »dateiname.FOR« wird geladen und die darin enthaltenen Anweisungen in Assembler-Befehle umgewandelt.

- Daraus wird eine Objektdatei erzeugt und diese unter dem Dateinamen »dateiname.OBJ« auf der Diskette gespeichert.

- Zusätzlich kann eine Print-Datei »dateiname.LST« erstellt werden.

Beim Kompilieren müssen die Dateien FORT.ERR und ASSM.COM auf der Bezugsdiskette sein.

a) Parameter

Mit dem 3stelligen Parameter PPP wird dem FORTRAN-Compiler mitgeteilt, auf welche Laufwerke die auszugebenden Dateien geschrieben werden sollen. Jede Stelle der Zeichenkette PPP hat eine besondere Bedeutung:

1. Stelle: Ziellaufwerk für Print-Datei »dateiname.LST«

2. Stelle: Ziellaufwerk für Assembler-Zwischendatei »dateiname.ASM«

3. Stelle: Ziellaufwerk für Objektdatei »dateiname.OBJ«.

Beim C 128 kann für jeden der 3 Parameter P das Laufwerk A bis D gewählt werden. Gibt man keinen Laufwerkparameter P an, wird das Bezugslaufwerk eingesetzt.

Statt die Print-Datei zu speichern, kann diese mit »X« an der 1. Stelle auf dem Bildschirm aufgelistet und mit »Y« auf einem CP/M-LIST-Gerät ausgegeben werden.

Schreibt man an eine Stelle ein »Z«, wird keine LST-, ASM- oder OBJ-Datei erzeugt. Wenn man ein »Z« an der 2. oder 3. Stelle einsetzt, wird keine Objektdatei erzeugt. Ist an der 3. Stelle kein »Z« eingesetzt, wird die Assembler-Zwischendatei zwar erzeugt, aber automatisch auf der Diskette wieder gelöscht. Wenn also an der 3. Stelle ein »Z« steht, wird keine Objektdatei erzeugt; die Assembler-Datei wird jedoch nicht gelöscht.

1. Stelle	2. Stelle	3. Stelle	Bedeutung
X	-	-	keine Print-Datei
Y	-	-	Druckerausgabe
Z	-	-	keine Print-Datei
-	Z	-	keine ASM-und OBJ-Datei
-	-	Z	keine OBJ-Datei, aber ASM-Datei

Nach der Kompilierung werden eventuelle Fehler beim Kompilieren und Assemblieren angezeigt.

b) Compiler-Optionen

In der Befehlszeile kann hinter dem Dollar-Zeichen »$« zusätzlich die Kompilierung des FORTRAN-Programms mit folgenden Optionen beeinflußt werden:

N — Es wird keine Assembler- und damit keine Objektdatei erzeugt.

P — Seitenumbruch, d.h., die Print-Datei wird auf 66 Zeilen pro Seite formatiert. Jede neue FORTRAN-Routine beginnt mit einer neuen Seite.

1 bzw. 2 — Die Quellprogramm-Anweisungen werden in der Print-Datei so formatiert, daß die Zeilen auf 64 bzw. 72 Zeichen mit Leerstellen ergänzt werden.

H — Die Kopfzeile einer Print-Datei durch die Option »P« wird unterdrückt.

C=xxxx — Maximale Anzahl der COMMON-Blöcke im Quellprogramm. Die Vorgabe beträgt 15.

B=xxxx — Größe des Eingabepuffers für FORTRAN-Anweisungen. Die Vorgabe beträgt 530 Zeichen für eine komplette FORTRAN-Anweisung.

M=xxxx — Hexadezimale Speicheradresse, die von den COMMON-Blöcken nicht überschritten werden darf.

Beispiele

`A>FORT PROG $C=20`	Kompilieren des Programms PROG.FOR von der Bezugsdiskette, Erzeugen von PROG.ASM, PROG.LST und PROG.OBJ auf der Bezugsdiskette A, wobei bis zu 20 COMMON-Blöcke möglich sind.
`A>FORT B:PROG.XAB $P1`	Kompilieren von PROG.FOR von Laufwerk B, Erzeugen von PROG.ASM auf Laufwerk A und von PROG.OBJ auf Laufwerk B, Ausgabe der formatierten Print-Datei auf dem Bildschirm.
`A>FORT PROG.BZZ $P`	Kompilieren von PROG.FOR von der Bezugsdiskette A, kein Erzeugen von PROG.ASM und PROG.OBJ, Ausgabe der formatierten Print-Datei auf das Laufwerk B.
`A>FORT PROG.XBZ $PH`	Kompilieren von PROG.FOR von der Bezugsdiskette A, Erzeugen von PROG.ASM auf Laufwerk B, Bildschirmausgabe der formatierten Print-Datei (ohne Kopfzeile), keine Datei PROG.OBJ.

7.3.5 Ausführen eines Programms

Nachdem man eine Objektdatei erzeugt hat, kann das Programm durch folgende Eingabe ausgeführt werden:

Format: **FRUN d:dateiname**

Es bedeuten:

d	Diskettenlaufwerk, das die Diskette mit der Objektdatei »dateiname.OBJ« enthält.
dateiname	Name der Datei »dateiname.OBJ«.

Beschreibung

Das Programm FRUN.COM wird in den Speicherbereich von 100H bis 3FFFH geladen, danach das Programm »dateiname.OBJ« von der Speicherstelle 4000H an. Anschließend wird es in ein ausführbares Programm überführt (Linker) und dann die Ausführung gestartet (RUN-Funktion). Nach dem Ende der Ausführung wird wieder ins CP/M-Betriebssystem zurückgesprungen.

Beispiel

`FORT PROG`	Print- und Objektdatei auf die Bezugsdiskette,
`FRUN PROG`	Ausführen des Programms.

7.3.6 Erzeugen einer COM-Datei

Mit FRUN.COM kann auch eine direkt ausführbare COM-Datei erzeugt werden.

Format: **FRUN d:dateiname.C**

Beschreibung

Die erzeugte COM-Datei »dateiname.COM« enthält eine Kopie von FRUN.COM und des auszuführenden Programms »dateiname.OBJ«. Das hat den Vorteil, daß man nur den Programmnamen »dateiname« zum Starten der Programmausführung eingeben muß. Jede so erzeugte COM-Datei ist mindestens 16 Kbyte lang, dies ist die Größe von FRUN.COM.

Beispiel

```
FRUN PROG.C     Erzeugen der Programmdatei PROG.COM,
PROG            Ausführen des Programms PROG.COM.
```

7.3.7 NEVADA-FORTRAN-Instruktionen

Jede FORTRAN-Anweisung hat folgendes Format:

- Die ersten 5 Zeichenstellen können eine Anweisungsmarke (LABEL) enthalten, z.B. wenn auf diese Anweisung ein Sprung durchgeführt wird. Enthält die erste Stelle ein »C«, so wird die nachfolgende Information nicht kompiliert; sie kann als Kommentarzeile verwendet werden.

- Die 6. Stelle muß im allgemeinen leer bleiben oder eine Null »0« enthalten. Paßt eine Anweisung nicht in eine Zeile, muß die mit der Fortsetzung der Anweisung folgende(n) Zeile(n) in Spalte 6 ein von Null oder dem Leerzeichen verschiedenes Zeichen enthalten.

- Die 7. bis 72. Stelle einer Zeile kann für eine FORTRAN-Anweisung genutzt werden.

Man unterscheidet bei FORTRAN zwei Arten von Zahlen:

- Gleitkommazahlen (REAL) und
- Festkomma- bzw. ganze Zahlen (INTEGER)

und 5 Arten von Konstanten:

- Gleitkommakonstanten,
- Festkommakonstanten,
- doppeltgenaue Konstanten (hier nicht implementiert),
- komplexe Konstanten (hier nicht implementiert) und
- logische Konstanten.

Tabelle 7.5 zeigt eine Übersicht der NEVADA-FORTRAN-Anweisungen, Tabelle 7.6 die FORTRAN-Funktionen und Tabelle 7.7 die FORTRAN-Subroutinen.

Tabelle 7.5: NEVADA-FORTRAN-Anweisungen (statement)

Anweisung	Beschreibung
v = a	Zuweisen des Wertes eines Ausdrucks zu einer Variablen.
ACCEPT	Einlesen von Werten von der Tastatur und Zuweisen zu einer Variablen.
ASSIGN TO	Zuweisen eines Anweisungs-Labels zu einer Variablen in einer zuweisenden GO-TO-Anweisung.
BACKSPACE	Stellt das bezeichnete Gerät auf den Record-Anfang.
BLOCK DATA	Anfang im BLOCK-DATA-Unterprogramm zur Initialisierung der COMMON-Variablen.
CALL	Aufruf eines Unterprogramms (subroutine).
COMMON	Deklaration von Variablen und Zahlenfeldern (array) für verschiedene Programmteile (Routinen).
CONTINUE	Fortsetzen des Programms ohne eine ausführbare Anweisung, normalerweise als Sprungmarke einer GOTO-Anweisung oder einer DO-Schleife verwendet.
COPY	Einfügen des bezeichneten Programms an der Stelle der COPY-Anweisung.
CTRL DISABLE	Unterdrücken eines Programmabbruchs mit »CTRL-C« (Warmstart) von der Tastatur aus.
CTRL ENABLE	Ein Programmabbruch mit »CTRL-C« (Warmstart) von der Konsole aus möglich.
DATA	Zuweisen von Konstanten zu Variablen.
DIMENSION	Festlegen der maximalen Werte der Laufindizes von Zahlenfeldern (array).

Anweisung	Beschreibung
DO n i=n1,n2,n3	Ausführen der Anweisungen von der DO-Anweisung bis zur Anweisung n, wobei i als Zählindex dient und von n1 nach n2 mit der Schrittweite n3 läuft.
DOUBLE PRECISION	Deklaration von Variablen mit doppelter Genauigkeit, hier mit einfacher Genauigkeit behandelt.
DUMP	Meldung bei einem Fehler während des Programmlaufs.
END	Letzte Anweisung jeder Routine.
ENDFILE	Schreiben einer Dateiende-Marke (1AH) an die derzeitige Position der Ausgabeeinheit.
ERRCLR	Löschen von ERRSET.
ERRSET n,v	Bei einem Ablauffehler wird zu der mit n bezeichneten Anweisung gesprungen, wobei die Variable v den Fehlercode enthält.
FORMAT	Ein- und Ausgabeformat.
FUNCTION	Definition eines vom Programmierer definierten FUNCTION-Unterprogramms.
GO TO n	Unbedingter Sprung zur mit n markierten Anweisung, auch als berechnetes und zugewiesenes »GO TO«.
IF(a)s	Ist der Wert des Ausdrucks a wahr, wird die Anweisung s ausgeführt.
IF(a)n1, n2,n3	Bedingter Sprung nach n1 bei a<0, n2 bei a=0 oder n3 bei a>0 (arithmetische IF-Anweisung).
IF(a) THEN s1 ELSE s2 ENDIF	Bei a wahr wird die Anweisung s1 ausgeführt, bei a falsch die Anweisung s2.
IMPLICIT	Ändern des bei Programmbeginn festgelegten Variablentyps.

Anweisung	Beschreibung
INTEGER	Festlegen als Festkommavariable.
LOGICAL	Festlegen als logische Variable.
PAUSE	Programmunterbrechung bis zum Drücken einer Taste.
READ	Variableneingabe von der Tastatur, aus Dateien oder DATA-Zeilen.
REAL	Festlegen als Gleitkommavariable.
RETURN	Rücksprunganweisung in einem Unterprogramm.
REWIND	Schließen und Wiedereröffnen einer logischen Datei.
STOP	Beenden der Ausführung eines Programms.
SUBROUTINE	Beginn eines SUBROUTINE-Unterprogramms.
TRACE OFF	Löschen von »Trace on«.
TRACE ON	Anzeigen der Zeilennummer der gerade ausgeführten Anweisung bei einem Programmablauf.
TYPE	Anzeigen der Variablenwerte auf dem Bildschirm.
WRITE	Ausgabe von Variablenwerten im gewählten Format auf ein bezeichnetes Ausgabegerät (Bildschirm, Diskette, Drucker).

Tabelle 7.6: NEVADA-FORTRAN-Funktionen

Funktion	Beschreibung
ABS, IABS	Absolutbetrag.
AMAX0, AMAX1, MAX0, MAX1	Maximalwert.
AMIN0,AMIN1, MIN0, MIN1	Minimalwert.
AMOD, MOD	Modulo-Funktion.
DIM, IDIM	Positive Differenz.
FLOAT	Umwandlung einer Festkommazahl in eine Gleitkommazahl.
IFIX	Umwandlung einer Gleitkommazahl in eine Festkommazahl.
SIGN, ISIGN	Vorzeichenverschiebung.
EXP	Exponentialfunktion.
ALOG	Natürlicher Logarithmus.
ALOG10	Zehnerlogarithmus.
SIN	Sinus.
COS	Cosinus.
TAN	Tangens.
ATAN,ATAN2	Arcustangens.
SQRT	Quadratwurzel.
RAND	Zufallsgenerator (0<x<1).

Funktion	Beschreibung
INP	Eingabe über einen Kanal.
CHAR	Wert eines Zeichens.
COMP	Vergleich von Zeichenketten.
PEEK	Inhalt einer Speicherzelle.

Tabelle 7.7: Installierte NEVADA-FORTRAN-System-Subroutinen (Aufruf mit CALL)

Subroutine	Beschreibung
BIT	Bit-Verarbeitung.
CBTOF	Umwandeln einer Binärzahl in ihre Gleitkommazahl.
CHAIN	Laden und Ausführen eines anderen Programms.
CIN	Lesen eines einzigen Zeichens von der Tastatur.
CLOSE	Schließen einer Datei auf einer Ein-/Ausgabeeinheit
CTEST	Abbruch bei Eingabe eines Zeichens über die Tastatur.
DELAY	Verzögern der Ausführung um eine bestimmte Zeitdauer.
DELETE	Löschen einer Datei von einer Diskette.
EXIT	Abbruch der Programmausführung.
MOVE	Verschieben von Bits.
OPEN	Eröffnen einer Datei auf einer Ein-/Ausgabeeinheit.
LOAD	Laden einer HEX- oder OBJ-Datei an die bezeichnete Stelle.
LOPEN	Öffnen einer Datei auf einem Drucker.
POKE	Ändern einer Speicherzelle.
OUT	Ändern eines Wertes in eine 8-Bit-Zahl und Ausgabe auf einem Kanal.
PUT	Ausgabe eines Zeichens auf die Konsole, z.B. Bildschirmlöschen.
RENAME	Umbenennen einer Datei.

Subroutine	Beschreibung
RESET	Reset-Routine zum Wechseln und anschließenden Beschreiben einer Diskette während der Ausführung eines Programms.
SEEK	Einstellen einer Datei auf eine vorgegebene Stelle.
SETIO	Ändern der Konsolenein-/-ausgabe.

7.4 Microsoft-FORTRAN-80

7.4.1 Einleitung

Ein weiteres auf dem C 128 installierbares FORTRAN-System ist FORTRAN-80 von Microsoft. Dieses FORTRAN-80 ist vergleichbar mit FORTRAN-Compilern für Großrechner und Minicomputer. Es enthält den Befehlssatz von ANSI-Standard-FORTRAN X 3.9-1966, außer dem Datentyp COMPLEX.

Für die Erstellung eines FORTRAN-Quellprogramms gilt das im vorigen Abschnitt Gesagte bzw. die Hinweise der einschlägigen FORTRAN-80-Handbücher.

Die Installation von FORTRAN-80 benötigt

- einen Rechner mit einer Mindestspeichergröße von 32 Kbyte,
- ein Diskettenlaufwerk und
- ein CP/M-Betriebssystem.

7.4.2 FORTRAN-80-Diskette

Die FORTRAN-80-Diskette muß folgende Dateien enthalten:

F80.COM	FORTRAN-80-Compiler (z.B. Version 3.4) zum Erstellen einer verschiebbaren Objektdatei (REL-Datei).
L80.COM	Linker-Programm (LINK-80, z.B. Version 3.41) zum Erstellen eines ablauffähigen Maschinenprogramms (COM-Datei) aus REL-Dateien.
FORLIB.REL	FORTRAN-Bibliotheksdatei mit den FORTRAN-Systemroutinen, die beim Erstellen einer COM-Datei mit dem Linker L80.COM auf der Diskette sein muß.

Die Diskette mit den obigen Programmen muß bei einer Verwendung auf dem C 128 mit den CP/M-Systemspuren formatiert sein. Vor einem ersten Arbeiten sollte man sich eine Arbeitsdiskette anfertigen.

7.4.3 Kompilieren mit dem FORTRAN-80-Compiler

Nachdem man sich ein FORTRAN-Quellprogramm mit der Dateikennung »FOR« erstellt hat, kann man dieses mit dem FORTRAN-80-Compiler in eine relative Objektdatei (REL-Datei) kompilieren:

Format:

```
F80
*objektdatei,listdatei=quelldatei/o
oder
F80    objektdatei,listdatei=quelldatei/o
```

Es bedeuten:

objektdatei	Zieldatei mit dem Format »d:dateiname.dateityp«, wohin die verschiebbare Objektdatei mit dem kompilierten Programm kopiert werden soll.
listdatei	Zieldatei mit dem Format »d:dateiname.dateityp«, die die Print-Datei enthält.
quelldatei	die zu kompilierende FORTRAN-Quelldatei mit dem Format »d:dateiname.dateityp«
o	Compiler-Option.

Beschreibung

Man kann den FORTRAN-80-Compiler (Datei F80.COM) auf zwei Arten aufrufen:

1. Dienstprogramm

   ```
   F80<CR>
   ```

   Der FORTRAN-80-Compiler F80-COM wird geladen und meldet sich mit dem Bereitschaftszeichen »*« zur Eingabe weiterer Befehle. Jetzt kann man nacheinander mehrere FORTRAN-Quellprogramme kompilieren, indem man die entsprechenden Dateinamen eingibt:

   ```
   *objektdatei,listdatei=quelldatei<CR>
   ```

   Mit einem Warmstart durch »CTRL-C« nach dem Bereitschaftszeichen kann man ins CP/M-Betriebssystem zurückspringen.

2. Befehlszeile

```
F80 objektdatei,listdatei=quelldatei<CR>
```

Der FORTRAN-80-Compiler F80.COM, dann das zu kompilierende FORTRAN-Quellprogramm »quelldatei« werden geladen und letzteres kompiliert. Danach erfolgt ein CP/M-Warmstart.

Das Kompilieren erzeugt aus den FORTRAN-Anweisungen einen verschiebbaren, relativen Objektcode, der in der mit »objektdatei« bezeichneten Objektdatei gespeichert wird. Gleichzeitig kann eine Print- oder List-Datei in die mit »listdatei« bezeichnete Datei geschrieben werden.

Bei der Kompilierung auftretende Fehler werden angezeigt. Es gibt zwei Arten einer Fehlermeldung: Warnungen und »fatale« Fehler.

Bei einer Warnung wird die Kompilierung mit dem nächsten Befehl in der Programmzeile fortgesetzt. Wenn ein »fataler« Fehler auftritt, wird der Rest einer logischen Zeile, einschließlich aller Fortsetzungszeilen, ignoriert. Warnungen werden mit einem Prozentzeichen »%« und »fatale« Fehler mit einem Fragezeichen »?« gekennzeichnet. Danach werden, falls vorhanden, die editierte oder die physikalische Zeilennummer und der Fehlercode oder die Fehlermeldung ausgedruckt.

a) Quell- und Zieldateien

Die Bezeichnung eines Laufwerkes d (A bis D) kann entfallen, dann wird die Bezugsdiskette angenommen.

Die Dateikennungen »dateityp« können ebenfalls weggelassen werden. Für das FORTRAN-Quellprogramm wird dann die Dateikennung »FOR« verlangt, für die Objektdatei die Dateikennung »REL«, und für die Print-Datei die Dateikennung »PRN« eingesetzt.

Wenn weder eine Print-Datei noch eine Objektdatei erzeugt werden soll, muß man nur ein Komma auf der rechten Seite des Gleichheitszeichens angeben.

Läßt man die Namen der Objektdatei bzw. der Print-Datei weg, wird der Name der Quelldatei eingesetzt.

Beispiele

`A>F80 =TEST`	Kompilieren des Programms TEST.FOR und Erzeugen der Objektdatei TEST.REL auf der Bezugsdiskette A.
`A>F80 ,=TEST.FOR`	Kompilieren von TEST.FOR, eine Objekt- und Print-Datei wird nicht erzeugt. Hiermit kann das Programm auf Fehler geprüft werden.
`A>F80 ,TTY:=TEST`	Kompilieren von TEST.FOR und Auflisten der Print-Datei auf dem Bildschirm. Eine Objektdatei wird nicht erzeugt.
`A>F80<CR>` `*TESTOBJ=TEST.FOR<CR>`	Kompilieren von TEST.FOR, Schreiben der Objektdatei in TESTOBJ.REL.
`A>F80<CR>` `*TEST,TEST=TEST<CR>`	Kompilieren von TEST.FOR, Schreiben der Objektdatei in TEST.REL und der Print-Datei in TEST.PRN.

b) Compiler-Optionen

In der Befehlszeile kann hinter einem Schrägstrich »/« zusätzlich die Kompilierung eines FORTRAN-Programms mit folgenden Optionen beeinflußt werden:

/O	Drucken aller Print-Datei-Adressen usw. in Oktalzahlen.
/N	Die Print-Datei enthält nicht den erzeugten Code.
/R	Eine Objektdatei wird auch bei Fehlern erzeugt.
/L	Eine Print-Datei wird auch bei Fehlern erzeugt.
/P	Jedes »P« reserviert zusätzlich einen 100 Byte großen Stapelspeicher (Stack) während des Kompilierens. Man benutzt diese Option, wenn während der Kompilierung ein Überlauf auftritt.
/M	Der kompilierte Code soll in ein ROM geladen werden.

Beispiele

`A>F80<CR>` `*,TTY:=PROG/N<CR>`	Kompilieren von PROG.FOR und Auflisten des Programms auf dem Bildschirm ohne den erzeugten Assembler-Code.
`A>F80 =TEST/L<CR>`	Kompilieren von TEST.FOR und Erzeugen von PROG.REL und TEST.LST.
`A>F80<CR>` `*=PROG/P/P<CR>`	Kompilieren von PROG.FOR und Erzeugen von PROG.REL. Beim Kompilieren wird ein zusätzlicher Stapelspeicher von 200 Byte angelegt.

7.4.4 Erzeugen eines ablauffähigen Maschinenprogramms

Das Erzeugen eines ablauffähigen Maschinenprogramms wird mit Linken (link=verbinden) bezeichnet. Zum Laden und Linken eines kompilierten Programms (REL-Datei) benutzt man ein Link-Programm (Linker Link-80, z.B. Version 3.41).

Format: **L80**
***objektdatei,zieldatei/o**
oder
L80 objektdatei,zieldatei/o

Es bedeuten:

objektdatei	die zu linkende Datei mit dem Format »d.dateiname.REL«
zieldatei	die mit dem Linker erzeugte Zieldatei mit dem Format »d:dateiname.COM«
o	Linker-Optionen

Beschreibung

Man kann den Microsoft-Linker (Datei L80.COM) auf zwei Arten aufrufen. Die FORTRAN-Bibliotheksdatei FORLIB.REL muß unbedingt mit dem Linker auf derselben Diskette sein.

1. Dienstprogramm

```
L80<CR>
```

   Der Linker L80.COM wird geladen und meldet sich mit der Bereitschaftsmeldung »*« zur Eingabe weiterer Befehle. Jetzt kann man nacheinander Dateien im Format »d:dateiname.REL« und zusätzliche Optionen eingeben. Die Dateien werden entsprechend der Reihenfolge der Eingabe gelinkt, d.h. aneinandergefügt.

2. Befehlszeile

```
L80 objektdatei,zieldatei/o<CR>
```

   Statt nacheinander die zu linkenden Dateien und Optionen einzugeben, können alle Befehle, durch Kommas getrennt, in einer Befehlszeile eingegeben werden.

a) Dateinamen

Der Linker lädt die Dateien in der angegebenen Reihenfolge nacheinander von der Startadresse 103H in den Speicher. Die Dateien müssen nicht in der Reihenfolge des späteren Ablaufs eingegeben werden, da in der Speicheradresse 100H - 102H ein Sprungbefehl zur Startadresse der ersten auszuführenden Anweisung geschrieben wird. Die relativen Adressen der verschiedenen Programme werden entsprechend der eingegebenen Reihenfolge und dem Assembler-Code in absolute Adressen umadressiert.

Dies ist der wesentliche Schritt bei der Konvertierung einer verschiebbaren REL-Datei in eine ablauffähige COM-Datei. Linken bedeutet also, daß jede geladene Datei, die den Programmablauf in einer anderen Datei steuert, selbst so mit der anderen Datei verbunden (gelinkt) wird, daß sie selbst den entsprechenden Code mit den entsprechenden Adressen enthält. Es können nicht nur kompilierte FORTRAN-Programme, sondern auch kompilierte BASIC-, Pascal-, COBOL- und Assemblerprogramme zu einer ablauffähigen COM-Datei gelinkt werden.

Der Linker kann danach das assemblierte und gelinkte Programm als ablauffähiges Maschinenprogramm auf der Diskette in einer COM-Datei abspeichern.

Jede vollständige Link-Anweisung enthält normalerweise mindestens 2 Dateinamen:

- Der erste Dateiname übermittelt dem Linker, welche REL-Datei geladen und gelinkt werden soll.

- Der zweite Dateiname befiehlt dem Linker, den ablauffähigen Code in dieser Datei zu speichern.

- Die eingegebenen Dateinamen werden durch Kommas getrennt.

Gibt man nur einen Dateinamen ein, wird entweder die COM-Datei nicht gespeichert, oder es wird überhaupt kein Programm geladen. Alle Dateien werden normalerweise vom Bezugslaufwerk gelesen und auf dieses geschrieben.

b) Link-Optionen (Switches)

Die Link-Optionen beeinflussen das Laden und Linken. Die Link-Optionen werden hinter einem Schrägstrich »/« eingegeben. Man kann beliebig viele Optionen verwenden. Die wichtigsten Link-Optionen sind folgende:

/G	Linken der in der Befehlszeile eingegebenen Dateien, dann Ausführen der erzeugten COM-Datei, danach CP/M-Warmstart.
/G:dateiname	Wie »/G«, zusätzlich wird die Anfangsadresse der erzeugten COM-Datei auf die der Datei »dateiname« gesetzt.
/E	Linken der in der Befehlszeile angegebenen Dateien, danach CP/M-Warmstart.
/E:dateiname	Wie »/E«, zusätzlich wird die Anfangsadresse der erzeugten COM-Datei auf die der Datei »dateiname« gesetzt.
/N	Speichern des ausführbaren Maschinenprogramms unter dem Namen, der unmittelbar vor »/N« steht, auf der bezeichneten Diskette (im allgemeinen die Bezugsdiskette). Ohne eine Angabe einer Dateikennung wird die Kennung »COM« eingesetzt. Die COM-Datei wird erst dann gespeichert, wenn ein »/E« oder »/G« nach dem »/N« folgt.
/N:P	Wie »/N«, es wird jedoch nur der Programmspeicher abgespeichert.
/P:xxxx	Setzen der hexadezimalen Anfangsadresse für Programme und Daten. Zusammen mit »/D« setzt »/P« nur den Programmstart.
/D:xxxx	Setzen der hexadezimalen Anfangsadresse nur für den Datenbereich.
/R	Rücksetzen des Linkers auf seinen Anfangszustand.

/U	Auflisten undefinierter globaler Variabler (globals).
/M	Auflisten der gesamten Referenzliste der globalen Variablen.
/O	Verwenden von Oktalwerten im Programm.
/H	Verwenden von Hexadezimalwerten im Programm; nur notwendig, wenn vorher »/O« verwendet wurde.
/X	Speichern der COM-Datei im ASCII-Hex-Format.
/Y	Speichern der COM-Datei und einer SYM-Datei in einem Sonderformat, um diese mit dem symbolischen Debugger SID oder ZSID von Digital Research zu bearbeiten.

Bei Fehlern während des Linkens werden Fehlermeldungen ausgegeben, entweder eine Warnung mit einem vorangestellten Prozentzeichen »%« oder »fatale« Fehler mit einem vorangestellten Fragezeichen »?«, dahinter dann eine Fehlermeldung.

Beispiele

`A>L80 PROG,PROG/N/E`	Laden und Linken von PROG.REL, Speichern von PROG.COM und CP/M-Warmstart.
`A>L80 PROG,PGM/N/G`	Laden und Linken von PROG.REL, Ausführen und Speichern von PGM.COM, danach ein CP/M-Warmstart.

`A>L80<CR>`
`*PROG<CR>`
`*PGM/N/G<CR>`
	Wie vorher.

7.4.5 Ausführen eines ausführbaren Programms

Nachdem man ein ausführbares Maschinenprogramm mit dem Dateityp »COM« von einem Programm erzeugt hat, kann die Ausführung dieses Programms durch Eingabe des Dateinamens (ohne die Dateikennung »COM«) gestartet werden.

Format: **d:dateiname**

Bei einem Fehler während des Programmablaufs wird ebenfalls eine Fehlermeldung ausgegeben. Bei einem »fatalen« Fehler wird der Ablauf abgebrochen und mit einem Warmstart ins CP/M zurückgesprungen. Bei einer Warnung wird das Programm nach einer Meldung fortgesetzt.

7.4.6 FORTRAN-80-Instruktionen

Tabelle 7.8 zeigt eine Übersicht der FORTRAN-80-Anweisungen, Tabelle 7.9 die FORTRAN-Funktionen und Tabelle 7.10 die FORTRAN-Subroutinen. Besonders vorteilhaft bei FORTRAN-80 sind die zahlreichen Funktionen zur Verarbeitung von Zahlen mit doppelter Genauigkeit (in BCD-Arithmetik).

Tabelle 7.8: FORTRAN-80-Anweisungen (statement)

Anweisung	Beschreibung
v = a	Zuweisen des Wertes eines Ausdrucks zu einer Variablen.
ASSIGN TO	Zuweisung eines Anweisungs-Labels zu einer Variablen in einer zuweisenden GO-TO-Anweisung.
BACKSPACE	Stellt das bezeichnete Gerät auf den vorherigen Record-Anfang.
BLOCK DATA	Anfang im BLOCK-DATA-Unterprogramm zur Initialisierung der COMMON-Variablen.
CALL	Aufruf eines Unterprogramms (SUBROUTINE).
COMMON	Deklaration von Variablen und Zahlenfeldern (array) für verschiedene Programmteile (Routinen).
CONTINUE	Fortsetzen des Programms ohne eine ausführbare Anweisung, normalerweise als Sprungmarke einer GO-TO-Anweisung oder einer DO-Schleife verwendet.
DATA	Zuweisen von Konstanten zu Variablen.
DECODE	Übertragen von Daten im Speicher mit gleichzeitigem Wandeln vom ASCII-Format in das spezifizierte.
DIMENSION	Festlegen der maximalen Werte der Laufindizes von Zahlenfeldern (array).
DO n i=n1,n2,n3	Ausführen der Anweisungen von der DO-Anweisung bis zur Anweisung n, wobei i als Zählindex dient und von n1 nach n2 mit der Schrittweite n3 läuft.
DOUBLE PRECISION	Deklaration von Variablen mit doppelter Genauigkeit, hier mit einfacher Genauigkeit behandelt.
ENCODE	Übertragen von Daten im Speicher mit gleichzeitigem Wandeln vom spezifizierten Format ins ASCII-Format.

Anweisung	Beschreibung
END	Letzte Anweisung jeder Routine.
ENDFILE	Schreiben einer Dateiende-Marke (1AH) an die derzeitige Position der Ausgabeeinheit.
EQUI-VALANCE	Der gleiche Speicherplatz wird von mehreren Variablen belegt.
EXTERNAL	Verwenden von Namen einer SUBROUTINE, einer FUNCTION oder eines BLOCK DATA als Argument in einer Anweisung.
FORMAT	Ein- und Ausgabeformat.
FUNCTION	Definition eines vom Programmierer definierten FUNCTION-Unterprogramms.
GO TO n	Unbedingter Sprung zur mit n markierten Anweisung, auch als berechnetes und zugewiesenes »GO TO«.
HEXA-DECIMAL	Festlegen als Hexadezimalwert mit Z'xxxx' oder X'xxxx'.
IF (a) s	Ist der Wert des Ausdrucks a wahr, wird die Anweisung s ausgeführt.
IF (a) n1,n2,n3	Bedingter Sprung nach n1 bei a<0, n2 bei a=0 oder n3 bei a>0 (arithmetische IF-Anweisung).
IF (a) THEN s1 ELSE s2 ENDIF	Bei a wahr wird die Anweisung s1 ausgeführt, bei a falsch die Anweisung s2.
INTEGER	Festlegen als Festkommavariable.
LITERAL	Festlegen als Zeichenkette (Hollerith), in einfachen Anführungsstrichen eingeschlossen.
LOGICAL	Festlegen als logische Variable.

Anweisung	Beschreibung
PAUSE	Programmunterbrechung bis zum Drücken irgendeiner Taste.
PROGRAM	Definition und erste Anweisung eines Programms.
READ	Eingabe einer Variablen.
REAL	Festlegen als Gleitkomma-Variable.
RETURN	Rücksprunganweisung in einem Unterprogramm.
REWIND	Schließen und Wiedereröffnen einer logischen Datei.
STOP	Beenden der Ausführung eines Programms.
SUB-ROUTINE	Beginn eines SUBROUTINE-Unterprogramms.
WRITE	Ausgabe von Variablenwerten im gewählten Format auf ein bezeichnetes Ausgabegerät.

Tabelle 7.9: FORTRAN-80-Funktionen

Funktion	Beschreibung
ABS, IABS, DABS, AINT, INT, IDINT	Absolutbetrag
AMAX0, AMAX1, MAX0, MAX1	Maximalwert.
AMIN0, AMIN1, MIN0, MIN1, DMIN1	Minimalwert.
AMOD, MOD, DMOD	Modulo-Funktion.
DIM, IDIM	Positive Differenz.
FLOAT	Umwandlung einer Festkommazahl in eine Gleitkommazahl.
IFIX	Umwandlung einer Gleitkommazahl in eine Festkommazahl.
SIGN, ISIGN, DSIGN	Vorzeichenverschiebung.
SNGL	Umwandeln einer doppelt genauen Zahl in eine einfach genaue Zahl.
DBLE	Umwandeln einer einfach genauen Zahl in eine doppelt genaue Zahl.
EXP, DEXP	Exponentialfunktion.
ALOG, DLOG	Natürlicher Logarithmus.
ALOG10, DLOG10	Zehnerlogarithmus.
SIN, DSIN	Sinus.
COS, DCOS	Cosinus.

Funktion	Beschreibung
TANH	Tangenshyperbolikus.
ATAN, DATAN ATAN2, DATAN2	Arcustangens.
SQRT, DSQRT	Quadratwurzel.
INP	Eingabe über einen Kanal.
PEEK	Inhalt einer Speicherzelle.

Tabelle 7.10: FORTRAN-80-System-Subroutinen (Aufruf mit CALL)

Subroutine	Beschreibung
FCHAIN	Laden und Ausführen eines anderen Programms.
OPEN	Öffnen einer Datei auf einem Gerät.
OUT	Ausgabe auf einen Kanal.
POKE	Ändern einer Speicherzelle.

7.5 Turbo-Pascal

7.5.1 Einleitung

Eine preiswerte und in letzter Zeit stark beachtete Pascal-Implementierung für Mikrocomputer ist Turbo-Pascal von Borland International Inc., Scotts Valley, USA.

Die Installation von Turbo-Pascal benötigt einen Rechner mit

- einem 8080/8085/Z80-Mikroprozessor,
- einer Mindestspeichergröße von etwa 31 Kbyte und
- einem Diskettenlaufwerk.

An zusätzlicher Software wird das CP/M-Betriebssystem benötigt. Im folgenden soll eine kurze Einführung in die Arbeitsweise und Anwendung dieses Pascal-Compiler-Systems gegeben werden. Zur ausführlichen Information sollten das mitgelieferte Handbuch und die einschlägigen Pascal-Lehrbücher zu Rate gezogen werden.

Der große Vorteil von Turbo-Pascal ist, daß Compiler und eingebauter Editor (ähnlich WordStar) eine Einheit bilden. Die Programme werden mit dem Bildschirmeditor eingegeben und dann kompiliert. Findet der Compiler einen Fehler, springt er genau an die Fehlerstelle im Quellprogramm. Der Fehler kann sofort behoben werden.

7.5.2 Turbo-Pascal-Diskette

Die Turbo-Pascal-Diskette enthält neben dem eigentlichen Pascal-Compiler noch Installationsprogramme zur Anpassung des eingebauten Texteditierprogramms an den verwendeten Rechner (Terminal, Konsole).

Die Diskette enthält folgende Dateien:

TURBO.COM	Das Turbo-Pascal-Programm (31 Kbyte für die Version 1.0) mit dem Pascal-Compiler; die aktuelle Fassung ist z.Z. die Version 3.0.
TURBO.OVR	Overlay-Datei für TURBO.COM (1 Kbyte); nur auf der Diskette beim Ausführen von COM-Dateien erforderlich.

TURBOMSG.OVR	Textdatei mit den Fehlermeldungen (1,5 Kbyte); dann nicht auf der Diskette beim Ausführen von Programmen erforderlich, wenn das System keine Kompilier-Fehlermeldungen ausgeben soll. Ein Fehler wird dann nur als Fehlernummer ausgegeben, wenn sie im Handbuch näher erläutert ist. In jedem Fall wird ein Fehler automatisch angezeigt. Diese Textdatei kann geändert werden, z.B. können die Texte ins Deutsche übersetzt werden.
TLIST.COM	Programm zum Listen eines Quelltextes; nicht auf der Diskette beim Ausführen eines Programms notwendig.
TINST.COM	Programm zum Installieren des Editors in das Turbo-Pascal-Programm; nicht auf der Diskette beim Ausführen eines Programms notwendig.
TINST.DTA	Datendatei für die Terminal-Installation, nicht auf der Diskette beim Ausführen eines Programms notwendig.
TINSTMSG.OVR	Texte des Installationsprogramms; nicht auf der Diskette beim Ausführen eines Programms erforderlich. Die Texte können z.B. in eine andere Sprache übersetzt werden.

Dateien mit Pascal-Quellprogrammen haben die Dateikennung »PAS«.

7.5.3 Laden von Turbo-Pascal

Nachdem man sich eine Arbeitskopie der Turbo-Pascal-Diskette erstellt hat, kann das System vom CP/M-Betriebssystem aus geladen werden.

```
TURBO<CR>
```

Es erscheint die Einschaltmeldung:

```
TURBO Pascal version 2.00 (CP/M-80, Z80) - Serial # xxxxxx
Copyright (C) 1983 by BORLAND International INC.

No Terminal selected

Include error messages (Y/N)?
```

Will man Kompilier-Fehlermeldungen ausgeben, muß man die Frage mit »Y« bejahen. Es wird dann »TURBOMSG.OVR« geladen. Danach erscheint das Hauptmenü (siehe Abschnitt 7.5.5). Turbo-Pascal kann ohne eine Terminal-Installation verwendet werden. Zur Installation verläßt man das Turbo-Pascal durch Eingabe von »Q«.

7.5.4 Terminal-Installation

Falls man eine Pascal-Version ohne installiertes Terminal hat und den eingebauten Editor verwenden will, lädt man das Installationsprogramm von CP/M aus mit:

```
TINST<CR>
```

Es müssen alle TINST-Dateien und TURBO.COM auf der Diskette sein. Auf dem Bildschirm erscheint das Installationsmenü:

```
TURBO Pascal installation menu.
Choose installation item from the following:
[S]creen installation - [C]ommand installation - e[X]it
Enter S, C or X:
```

Nach Eingabe von »S« wird eine Reihe von Namen direkt installierbarer Terminals ausgelistet. Ist der Commodore 128 nicht aufgeführt, kann man eine der erlaubten Konsolen (Terminals) des C 128 verwenden. Die Installation des »Lear-Siegler ADM-31« zeigte befriedigende Ergebnisse. Die nachfolgende Frage nach einer Modifizierung kann mit »N« verneint werden. Die Frage nach einem Ändern der CPU-Betriebsfrequenz von 4 MHz kann mit RETURN übersprungen werden.

Als nächstes kann man eventuell durch Eingabe von »C« die Editierbefehle neu zuordnen. Der eingebaute Editor hat nahezu denselben Satz an Editierbefehlen wie das Textverarbeitungsprogramm »WordStar«. Durch einfaches Drücken der gewünschten Taste oder durch Eingabe des Tastencodes kann die Zuordnung geändert werden, z.B. können die Funktionstasten des C 128 belegt werden.

Durch Eingaben von »X« werden die neuen Terminaldaten in die Datei TURBO.COM geschrieben.

7.5.5 Bearbeiten einer Programmdatei

Nachdem man das Turbo-Pascal nochmals geladen hat, erscheint nach der Einschaltmeldung das Hauptmenü auf dem Bildschirm:

```
Logged drive: A

Work file:
Main file:

Edit    Compile Run  Save
eXecute Dir     Quit compiler Options

Text:      0 bytes (7714-7714)
Free: 10 203 bytes (7715-EA06)

>
```

Dieses Menü zeigt die jetzt verfügbaren Funktionsbefehle. Ein Befehl wird durch Drücken des in diesem Befehl enthaltenen Großbuchstabens eingegeben und direkt ausgeführt. Die RETURN-Taste darf nicht gedrückt werden.

a) L (Bezugslaufwerk)

Mit »L« kann das Bezugslaufwerk geändert werden. Nach Eingabe von »L« wird nach dem neuen Bezugslaufwerk gefragt:

```
New drive:
```

Es kann das Laufwerk A bis D gewählt werden. Die Eingabe muß mit »RETURN« abgeschlossen werden. Soll nicht das Laufwerk, sondern die Diskette gewechselt werden, drückt man nach Einlegen der Diskette sofort die RETURN-Taste.

b) W (Arbeitsdatei)

Mit »W« wird die Arbeitsdatei gewählt, d.h. die Datei, die editiert, kompiliert, ausgeführt und gespeichert werden soll. Nach Eingabe von »W« wird nach dem Namen der Arbeitsdatei gefragt:

```
Work file name:
```

Es kann ein gültiger Dateiname im CP/M-Format »dateiname.typ« eingegeben werden. Bei Eingabe eines Dateinamens ohne einen Dateityp und ohne einen Punkt wird automatisch die Kennung »PAS« angenommen. Die Dateitypen »BAK«, »CHN« und »COM« sind bei Turbo-Pascal für besondere Zwecke reserviert.

c) M (Hauptdatei)

Mit »M« kann eine Hauptdatei (main file) definiert werden, die dann angegeben werden muß, wenn nicht die Arbeitsdatei kompiliert werden soll.

d) E (Editieren)

Mit »E« wird der eingebaute Editor aufgerufen, und die vorher definierte Arbeitsdatei kann editiert (bearbeitet) werden.

e) C (Kompilieren)

Mit »C« wird der Compiler aktiviert. Ohne Angabe einer Hauptdatei wird die Arbeitsdatei kompiliert, sonst die Hauptdatei. Mit einer Compiler-Option (Befehl »O«) kann vorher eingestellt werden, ob die kompilierte Datei im Arbeitsspeicher bleibt oder in einer COM- oder einer CHN-Datei gespeichert wird. Beim Kompilieren wird Maschinencode erzeugt.

f) O (Compiler-Optionen)

Mit »O« können vor dem Kompilieren einer Datei verschiedene Compiler-Optionen gewählt werden:

```
compile ->      Memory
                Com-file
                cHn-file
Find run-time error
Quit
```

Die wichtigsten Compiler-Optionen sind folgende:

M Die kompilierte Datei bleibt im Arbeitsspeicher. Im Hauptmenü kann diese Datei mit »R« gestartet werden.

C Die kompilierte Datei mit dem Namen der Arbeitsdatei (evtl. der Hauptdatei) wird als COM-Datei auf der Bezugsdiskette gespeichert. Diese Datei enthält den Programmcode und das Pascal-Runtime-Bibliotheksprogramm. Sie kann deshalb wie üblich durch Eingabe des Dateinamens »dateiname« (ohne den Dateityp) direkt gestartet werden. Zusätzlich können mit »S« die Anfangsadresse und mit »E« die Endadresse vorgewählt werden.

H Die kompilierte Datei wird mit dem Namen der Arbeitsdatei (oder evtl. der Hauptdatei) als CHN-Datei auf der Bezugsdiskette gespeichert. Diese Datei enthält nur den Programmcode. Sie kann nur von einem anderen Turbo-Pascal-Programm mit der Pascal-Prozedur »CHAIN« aufgerufen werden.

F Bei einem Ablauffehler eines im Arbeitsspeicher befindlichen, kompilierten Programms kann der Fehler im Quelltext gefunden werden.

Q Rücksprung ins Hauptmenü.

g) S (Speichern)

Mit »S« kann die derzeitige Arbeitsdatei auf der Bezugsdiskette gespeichert werden. Die alte Version dieser Datei wird, falls vorhanden, mit dem Dateityp »BAK« umbenannt.

h) X (Ausführen anderer Programme)

Mit »X« können andere Programme genauso wie unter dem CP/M-Betriebssystem ausgeführt werden, z.B. Kopieren von Programmen mit »PIP«. Nach Eingabe von »X« erscheint auf dem Bildschirm:

```
Command:
```

Danach kann jeder gültige CP/M-Befehl eingegeben werden.

i) D (Disketten-Inhaltsverzeichnis)

Mit »D« werden das Disketten-Inhaltsverzeichnis (directory) und der verbleibende Diskettenspeicher aufgelistet. Nach Eingabe von »D« erscheint auf dem Bildschirm:

```
Dir mask:
```

Jetzt kann das gewünschte Diskettenlaufwerk (A bis D) oder ein gültiger Dateiname im CP/M-Format »d:dateiname.typ« (auch mit den Ersatzzeichen »*« und »?«) eingetippt werden. Drückt man nur die RETURN-Taste, wird das gesamte Disketten-Inhaltsverzeichnis der Bezugsdiskette aufgelistet.

j) Q (Rücksprung ins CP/M)

Mit »Q« verläßt man das Turbo-Pascal-System und kehrt mit einem Warmstart ins CP/M-Betriebssystem zurück. Falls man die Arbeitsdatei nach dem Laden editiert hatte, muß man die Frage, ob man diese noch auf der Diskette speichern möchte, mit »Y« bejahen.

7.5.6 Turbo-Pascal-Befehle

Der Turbo-Pascal-Compiler ist ein vollständiges Pascal-Entwicklungssystem, das sich fast genau an die Sprachdefinition von Jensen und Wirth hält und sogar über den neu vorgeschlagenen, leistungsfähigen ISO-Standard hinausgeht. Zwei Abweichungen sind das Fehlen von »GET« und »PUT« sowie die Beschränkung der Reichweite der »GO-TO«-Anweisung auf den jeweiligen Programmblock. Die Standardprozeduren »GET« und »PUT« sind durch »READ« und »WRITE« ersetzt, die erweitert wurden.

Gegenüber dem Pascal-Standard von Jensen und Wirth ist Turbo-Pascal erweitert worden:

- Die Reihenfolge der Variablen-, Konstanten- und Typendefinitionen kann beliebig sein.

- Es sind zusätzliche Prozeduren vorhanden.

Die Rechengenauigkeit der Gleitkomma-Arithmetik beträgt 11 Stellen und umfaßt den Bereich von E-38 bis E+38.

Tabelle 7.11 zeigt eine Übersicht der Turbo-Pascal-Anweisungen, Tabelle 7.12 die Pascal-Funktionen und Tabelle 7.13 die Pascal-Prozeduren. Als Arbeitsreferenz sollte man das mitgelieferte Handbuch »TURBO-PASCAL - Reference Manual« verwenden sowie zusätzlich auf die einschlägigen Lehrbücher zurückgreifen.

Tabelle 7.11: Übersicht über die Turbo-Pascal-Anweisungen (statement)

Anweisung	Beschreibung
v:=e	Zuweisungsoperator.
absolute	absolute Variable, Zuweisen der Speicheradresse.
and	arithmetischer und logischer UND-Operator.
array..of..	Typdefinition von Feldern; vordefinierte Felder: Mem (CPU-Speicher), Port (CPU-Daten-Ports).
begin..end	abgeschlossene Anweisung (Programmblock).
case..of..end	Ausführen der durch einen Ausdruck
case..of..	gekennzeichneten Anweisung.
else..end	
const	Definition von Konstanten.
div	Divisionsoperator für Ganzzahlen.
external	Zuweisen der Anfangsadresse bei der Deklaration von externen Maschinencode-Prozeduren und -Funktionen.
file	Definition von nichttypisierten Dateien.
file of	Definition von Dateien.
for..to..do..	Programmschleife mit Laufparameter.
for..downto..do	
forward	Vordeklaration einer Funktion oder Prozedur.
function	Deklaration einer Funktion.
goto	unbedingte Sprunganweisung.
if..then..	bedingte Anweisung: Anweisung ausführen,
if..then..	falls Ausdruck wahr.
else..	
in	Operator zum Prüfen auf Untermenge.
inline	Einfügen eines Maschinencode-Programms.
label	Deklaration von Sprungmarken (Labels).
mod	Modulus-Operator für Ganzzahlen.
nil	leere dynamische Variable (pointer).
not	Negation (Operator).
or	arithmetischer und logischer ODER-Operator.
packed	nicht implementiert (leere Anweisung).
procedure	Deklaration einer Prozedur.
program	Eröffnen eines Programms.
record..end	Typdefinition eines Record-Feldes.
repeat..until..	Wiederholen einer Anweisung bis Ausdruck wahr.
set of	Typdefinition eines Satzes von Elementen.

Anweisung	Beschreibung
shl	bitweise Schiebeoperation nach links.
shr	bitweise Schiebeoperation nach rechts.
string	Typdefinition von Zeichenketten.
type	Typdefinition, vordefinierte Typen: Boolean, Byte, Char, Integer, Real.
var	Deklaration von Variablen, vordefinierte Variablen: Boolean, Byte, Char, Integer, Real, Text, ↑ (dynamische Variable, pointer).
while..do..	Ausführen bzw. Wiederholen einer Anweisung solange Ausdruck wahr.
with..do..	Zuweisen von Variablen in Record-Feldern.
xor	arithmetischer und logischer Exklusiv-ODER-Operator.

Tabelle 7.12: Übersicht über die Turbo-Pascal-Funktionen (function)

Funktion	Beschreibung
Abs	Absolutwert.
Addr	Erste Speicheradresse.
ArcTan	Arcustangens.
AuxIn	Geräteeingabe, BIOS-Funktion READER.
BdosHL	Aufruf der bez. BDOS-Funktion, Ergebnis in HL.
BiosHL	Aufruf der bez. BIOS-Funktion, Ergebnis in HL.
Chr	ASCII-Zeichen aus Ganzzahl.
Concat	Zusammenfassen mehrerer Zeichenketten.
ConIn	Konsoleneingabe (Tastatur), BIOS-Funktion CONIN.
ConSt	Konsolenstatus, BIOS-Funktion CONST.
Copy	Vervielfachen einer Zeichenkette.
Cos	Cosinus.
Eof	Abfrage auf Dateiende.
Eoln	Abfrage auf Zeilenende in einer Textdatei.
Exp	Exponentialfunktion.
FilePos	Position des Diskettendatei-Zeigers.
FileSize	Größe einer Diskettendatei.
Frac	Nachkommateil einer Zahl.
Hi	Umwandeln eines High-Bytes in ein Low-Byte, High-Byte dann Null.
IOresult	Abfrage nach Ein-/Ausgabefehler während eines Programmablaufs.
Int	Vorkommateil einer Zahl.
KeyPressed	Anzeige bei einer gedrückten Konsolentaste.
Length	Zeichenanzahl einer Zeichenkette.
Ln	Natürlicher Logarithmus.
Lo	Löschen des High-Bytes.
MemAvail	Abfrage des verfügbaren Arbeitsspeichers für dynamische Variablen.
Odd	Abfrage, ob ungerade Zahl.
Ord	Ordnung der Typdeklaration.
Pos	Erstes Auftreten einer Zeichenkette.
Pred	Vorstehende Größe.
Ptr	Umwandeln einer Ganzzahl in eine dynamische Variable (pointer).
Random	Zufallszahl zwischen 0 und 1.
Round	Runden einer Zahl auf die nächste Ganzzahl.
Sin	Sinus.

Funktion	Beschreibung
SizeOf	Von einer Variablen belegter Speicherbereich.
Sqr	Quadrat.
Sqrt	Quadratwurzel.
Succ	Nachfolgende Größe.
Swap	Vertauschen von High- und Low-Byte.
Trunc	Die zu Null nächstgelegene Ganzzahl.
UsrIn	Geräteeingabe, BIOS-Funktion CONIN.

Tabelle 7.13: Übersicht über die Turbo-Pascal-Prozeduren (procedure)

Prozedur	Beschreibung
Assign	Zuweisen eines Diskettendateinamens.
AuxOut	Geräteausgabe, BIOS-Funktion PUNCH.
Bdos	Aufruf der bezeichneten BDOS-Funktion.
Bios	Aufruf der bezeichneten BIOS-Funktion.
BlockRead	Blocklesen aus Diskettendatei.
BlockWrite	Blockschreiben in Diskettendatei.
Chain	Laden und Starten einer CHN-Datei.
Close	Schließen einer Diskettendatei.
ClrEol	Löschen aller Zeichen von der Cursorposition bis zum Zeilenende.
ClrScr	Löschen des Bildschirms, Zurücksetzen des Cursors.
ConOut	Konsolenausgabe (Bildschirm), BIOS-Fkt. CONOUT.
DelLine	Löschen einer Zeile.
Delay	Warteschleife.
Delete	Teilzeichenkette in einer Zeichenkette löschen.
Erase	Löschen einer Diskettendatei.
Execute	Laden und Starten einer COM-Datei.
Exit	Senden der Terminal-Reset-Zeichenkette.
FillChar	Füllen eines Speicherbereichs mit einem vorgegebenen Wert.
Flush	Löschen des internen Sektorpuffers einer Diskettendatei.
GetMem	Zuweisen eines Speicherbereiches für dynamische Variablen.
GotoXY	Cursor auf eine vorgegebene Position setzen.
Halt	Unbedingtes Anhalten des Programmablaufs.
HighVideo	Helligkeit des Bildschirms hochsetzen.
Init	Terminal-Initialisierungs-Zeichenkette senden.
InsLine	Einfügen einer Leerzeile.
Insert	Teilzeichenkette in eine Zeichenkette einfügen.
LowVideo	Helligkeit des Bildschirms zurücksetzen.
LstOut	Druckerausgabe, BIOS-Funktion LIST.
Mark	Anfangsadresse des Speicherbereichs für dynamische Variablen.
Move	Kopieren eines Speicherbereichs.
New	Speicherbereich einer dynamischen Variablen (pointer) zuweisen.
Randomize	Initialisieren des Zufallsgenerators.
Read	Lesen einer Datei.
Readln	Springen zum nächsten Zeilenanfang in einer Textdatei.

Prozedur	Beschreibung
Release	Zurücksetzen eines Zeigers für dynamische Variablen auf die Anfangsadresse des verfügbaren Speicherbereichs.
Rename	Umbenennen einer Datei.
Reset	Vorbereiten einer Datei.
Rewrite	Eröffnen einer neuen Datei.
Seek	Suchen in einer Diskettendatei.
Str	Umwandeln einer Zahl in eine Zeichenkette.
UsrOut	Geräteausgabe, BIOS-Funktion CONOUT.
Val	Umwandeln einer Zeichenkette in eine Zahl.
Write	Schreiben einer Datei.
Writeln	Einfügen einer Zeilenendmarke (CR/LF).

8 Struktur von CP/M 3.0 auf dem Commodore 128

8.1 Einleitung

Im folgenden sollen einige technische Informationen über die Implementierung von CP/M 3.0 (CP/M Plus) auf dem Commodore 128 gegeben werden.

Der C 128 ist ein 2-Prozessor-System mit dem Hauptprozessor 8502 (eine Weiterentwicklung des 6502 und 6510) und dem Zusatzprozessor Z80. Der 8502 hat denselben Befehlssatz wie der 6502. Im normalen C-128-Modus mit BASIC arbeitet der Commodore 128 nur mit dem 8502. Die Hauptaufgabe des Z80 ist das Arbeiten mit dem CP/M-Betriebssystem (CP/M-Modus). Unter CP/M sind die normalen Funktionen des C 128 (C-128-Modus, C-64-Modus) nicht verfügbar; d.h. CP/M und BASIC können nicht gleichzeitig arbeiten. CP/M unterstützt nicht alle Anzeige-Moden des VIC-Chips; es ist möglich, ein Treiberprogramm für die Anwendung des VIC-Chips mit seinen besonderen Graphikeigenschaften unter CP/M zu schreiben, aber es müssen dabei alle Einzelheiten, wie z.B. die Speicherbereiche, beachtet werden.

8.2 Speicherorganisation

Der Speicherbereich ist auf 64 Kbyte begrenzt. Jedoch kann eine RAM-Bank ausgewählt werden; dabei können verschiedene ROM-Bereiche den RAM-Bereich überlagern. Der aktuelle Speicherbereich wird von der MMU (memory management unit = Speicherverwaltungseinheit) gesteuert. Die MMU kann auf den Ein-/Ausgabebereich infolge des Lade-Konfigurations-Registers (load configuration register) an der Speicheradresse FF00H bis FF04H zugreifen. Beim Lesen des Lade-Konfigurations-Registers werden nur deren gerade aktuelle Werte gelesen. Ein Schreiben nach FF00H (Konfigurations-Register) ändert die Konfiguration nach dem Beenden der laufenden Anweisung. Ein Schreiben in die Prekonfigurations-Register von FF01H bis FF04H aktualisiert die derzeit laufende Konfiguration. Die MMU-Page-Zeiger bestehen aus einem Low- und einem High-Zeiger. Der High-Zeiger wird zuerst geschrieben und in der MMU zwischengespeichert; er wird aktualisiert, wenn das Low-Byte geschrieben wird.

Die MMU-Steuerregister zeigt Tabelle 8.1. In Tabelle 8.2 ist die allgemeine, in Tabelle 8.3 die Z80- und in Tabelle 8.4 die 8502-Speicherbelegung des C 128 dargestellt.

Tabelle 8.1: MMU-Steuerregister

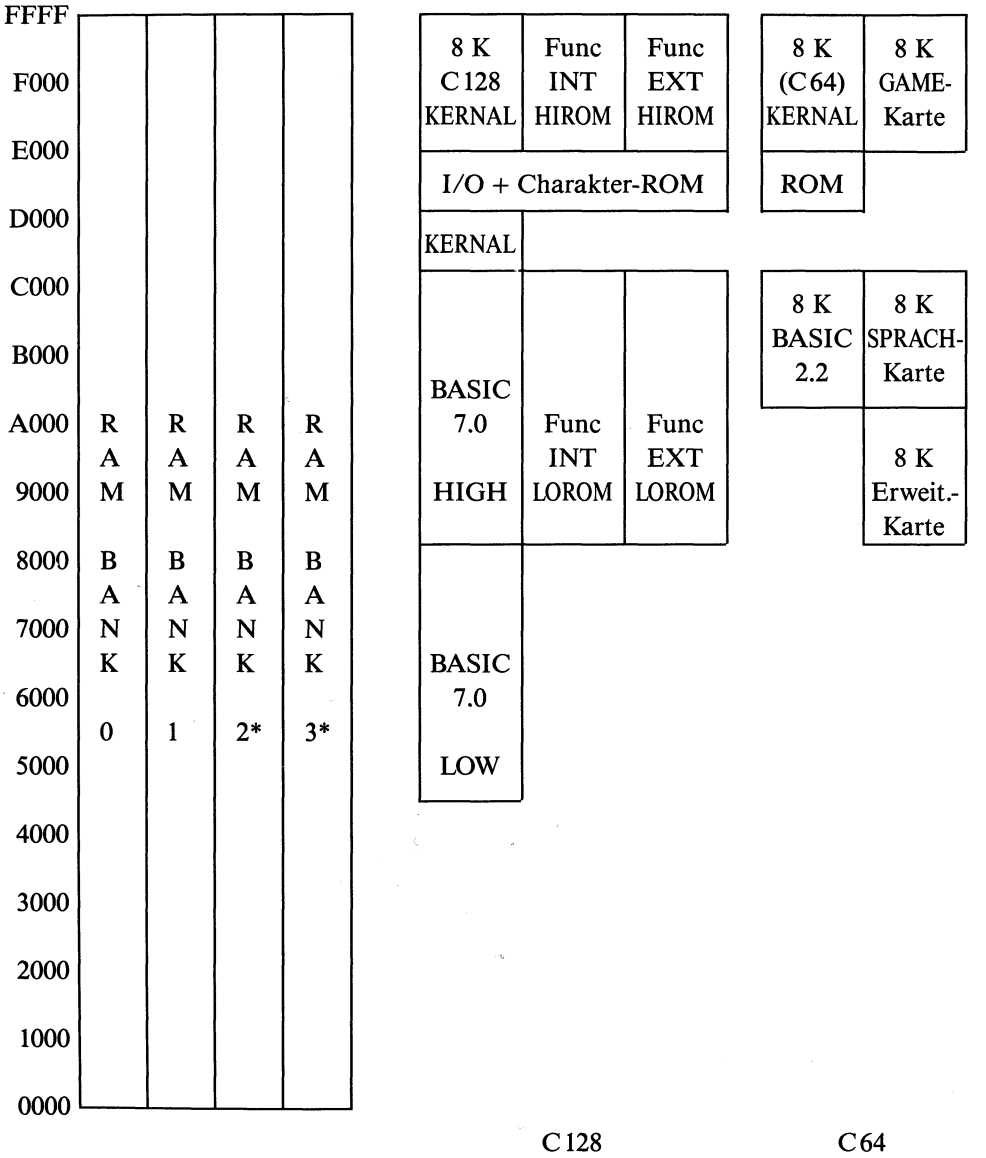
Speicher	Register	Bit	Bedeutung
D500 (FF00) D501 (FF01) D502 (FF02) D503 (FF03) D504 (FF04)	Konfig.-Reg. Prekonfig.-Reg. A Prekonfig.-Reg. B Prekonfig.-Reg. C Prekonfig.-Reg. D	7,6	RAM-Bank-Auswahl 00 = Bank 0 01 = Bank 1 10 = Bank 2 11 = Bank 3
		5,4	Speicherbereich $C000 - $FFFF 00 = System ROM 01 = Internal Funktions-ROM (high) 10 = External Funktions-ROM (high) 11 = RAM
		3,2	Speicherbereich $8000 - $BFFF 00 = BASIC-ROM (high) 01 = Internal Funktions-ROM (low) 10 = External Funktions-ROM (low) 11 = RAM
		1	Speicherbereich $4000 - $7FFF 0 = BASIC-ROM (low) 1 = RAM
		0	Speicherbereich $D000 - $DFFF
D505	Modus-Konfig.-Reg.	7	40/80-Zeichenbildschirm 0 = Schalter geschlossen 1 = Schalter offen

Speicher	Register	Bit	Bedeutung
		6	Betriebsart 0 = C 128 1 = C 64
		5	Bidirektionaler Port 0 = EXROM 1 = Farb-RAM mit VIC aktiv
		4	Bidirektionaler Port 0 = GAME 1 = Farb-RAM mit CPU aktiv
		3	Schneller serieller Port 0 = Dateneingang 1 = Datenausgang
		2,1	nicht benutzt
		0	CPU 0 = 8502 1 = Z80
D506	RAM-Konfig.-Reg.	7,6	VIC-RAM-Bank 0 = RAM-Bank 0 1 = RAM-Bank 1
		5,4	RAM-Bank 00 = 0-256 K 01 = 256-512 K* 10 = 512-768 K* 11 = 768-1 M*
		3,2	Gemeinsamer Speicherbereich 00 = 1 Kbyte 01 = 4 Kbyte 10 = 8 Kbyte 11 = 16 Kbyte

Speicher	Register	Bit	Bedeutung
D507	Zeiger f. Zero-Page	High-Byte	
D508	Zeiger, Page 0 (high)	7-4	ohne Bedeutung
D509	Zeiger, Page 1 (low)	3-0	gewählte Seite
D50A	Zeiger, Page 1 (high)	Low-Byte 7-0	Speicherseite in der 64-K-Bank

(*) Im C 128 nicht implementiert; für zukünftige Anwendungen reserviert.

Tabelle 8.2: Allgemeine Speicherbelegung des C 128

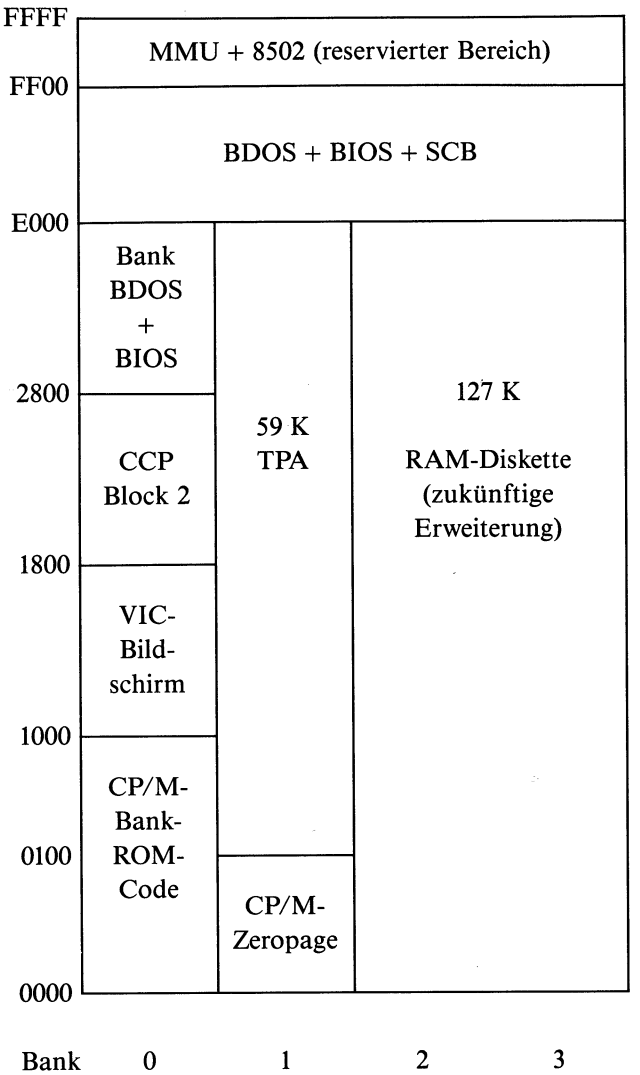


(*) Für zukünftige Erweiterungen

Bildschirmanzeige: 24 Zeilen (1 - 24)
80 Spalten (1 - 80)

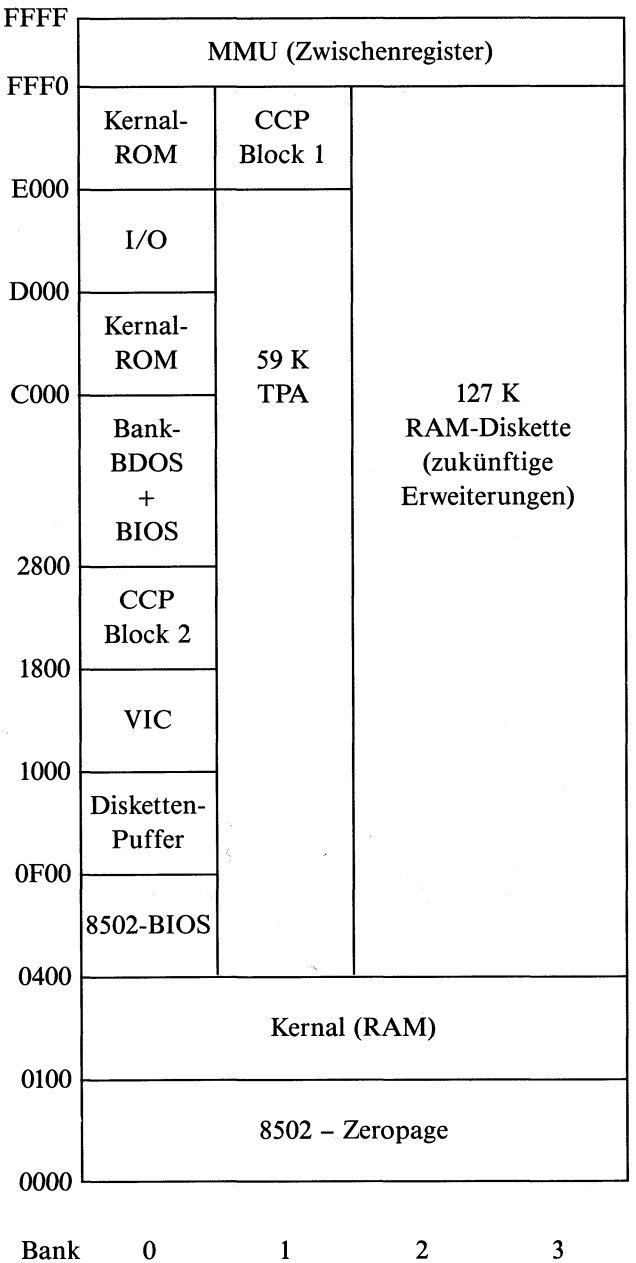
Home-Position des Cursors: Zeile 1, Spalte 1

Tabelle 8.3: Z80-Speicherbelegung



Es bedeutet: SCB = System-Kontroll-Block

Tabelle 8.4: 8502-Speicherbelegung



8.3 Diskettenorganisation

CP/M Plus auf dem C 128 kann verschiedene Diskettenformate lesen.

8.3.1 C-64-CP/M-2.2-Format

Als erstes wird das Diskettenformat für das CP/M 2.2 auf dem C 64 unterstützt (Tabelle 8.5). Bei diesem Format ist der Datei-Kontroll-Block FCB (file control block) auf 32 Spuren mit je 17 Sektoren und einem Spur-Offset von 2 gesetzt. Das BIOS überspringt die Spur 18, die Spur des C-64-Disketten-Inhaltsverzeichnisses, indem bei den Spuren, die größer als 18 sind, ein Offset von 1 hinzuaddiert wird.

8.3.2 C-128-CP/M-3.0-Format

Das C-128-CP/M-Format ist neu und nutzt die gesamte Diskettenkapazität aus (Tabelle 8.6). Der FCB ist auf 638 Spuren pro Seite mit je einem Sektor und einem Offset von eins (d.h. einem Sektor) gesetzt. Hierbei setzt CP/M die Spur auf die Blocknummer, die relativ zum Diskettenanfang (Sektor 0) gezählt wird. Der folgende Algorithmus dient zum Umrechnen der gewünschten Blocknummer T in eine reale Spur- und Sektornummer:

Effektive Spur (1 - 35) = INT (F)
Effektiver Sektor (0 - 20) = MOD (F) mit

Blocknummer T	Funktion F	Diskettenbereich N
000 >= T > 357	((T-000-0)/21) - 01	1
357 >= T > 490	((T-357-1)/19) - 18	2
490 >= T > 598	((T-490-1)/18) - 25	3
598 >= T > 683	((T-598-1)/17) + 31	4

Der effektive Sektor wird dann in eine Tabelle übertragen, um einen Schlüssel zur Beschleunigung der Vorgänge bereitzustellen. Diese Schlüsseltabelle wird nur beim neuen Format verwendet. Eine eigene Schlüsseltabelle wird für jeden Diskettenbereich verwendet.

Diese Art der Behandlung einer zweiseitigen Diskette ist nicht normal, aber mit dieser Belegung einer Diskette kann man mit einem Laufwerk 1541 noch die Daten auf der ersten Seite einer zweiseitigen Diskette lesen.

8.3.3 Weitere Diskettenformate

Die übrigen Formate benötigen das Diskettenlaufwerk 1571. Dieses Laufwerk unterstützt einseitige und doppelseitige Disketten. Die folgenden Diskettenformate sind eingebaut und über die Tastatur anwählbar: OSBORNE, KAYPRO, EPSON und IBM-CP/M86. Weitere Formate sind mit einem Konfigurationsprogramm verfügbar. Die eingebauten Formate können über die Tastatur mit einer speziellen Tastensequenz ausgewählt werden. Obwohl CP/M86-Programme auf dem C 128 nicht arbeiten, können Daten im IBM-CP/M86-Format zwischen Rechnern übertragen werden.

OSBORNE	nur einseitig		
KAYPRO	einseitig		
	doppelseitig		
EPSON			
IBM-CP/M86	einseitig,	8 Sektoren/Spur	(SS-8)
	einseitig,	9 Sektoren/Spur	(SS-9)
	zweiseitig,	8 Sektoren/Spur	(DS-8)
	zweiseitig,	9 Sektoren/Spur	(DS-9)

SS-8

40 Spuren/Seite (1 Seite)
8 Sektoren, 512-Byte-Sektoren/Spur
1 Spur als Offset 1K Belegungsgröße (allocation size)
64 Directory-Einträge (2 Belegungseinheiten)

SS-9

40 Spuren/Seite (1 Seite)
9 Sektoren, 512-Byte-Sektoren/Spur
1 Spur als Offset
1K Belegungsgröße (allocation size)
64 Directory-Einträge (2 Belegungseinheiten)

DS-8	40 Spuren/Seite (2 Seiten) 8 Sektoren, 512-Byte-Sektoren/Spur 1 Spur als Offset 2 K Belegungsgröße (allocation size) 64 Directory-Einträge (2 Belegungseinheiten)
DS-9	40 Spuren/Seite (2 Seiten) 9 Sektoren, 512-Byte-Sektoren/Spur 1 Spur als Offset 2 K Belegungsgröße (allocation size) 64 Directory-Einträge (2 Belegungseinheiten)

Tabelle 8.5: C-64-CP/M-Diskettenformat

Spur	Sektor																			
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	X	X	X	X
1	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	X	X	X	X
2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X	X	X	X
3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X	X	X	X
4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X	X	X	X
5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X	X	X	X
6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X	X	X	X
7	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X	X	X	X
8	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X	X	X	X
9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X	X	X	X
10	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X	X	X	X
11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X	X	X	X
12	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X	X	X	X
13	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X	X	X	X
14	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X	X	X	X
15	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X	X	X	X
16	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X	X	X	X
17	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D		
18	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X	X		
19	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X	X		
20	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X	X		
21	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X	X		
22	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X	X		
23	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X	X		
24	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X			
25	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X			
26	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X			
27	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X			
28	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X			
29	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X			
30	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-				
31	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-				
32	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-				
33	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-				
34	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-				

. von CP/M verwendet
B Ladesektor (Betriebssystem)
D Sektor des C-64-Disketten-Inhaltsverzeichnisses
X nicht von CP/M verwendet

Tabelle 8.6: C-128-CP/M-Diskettenformat

Spur	Sektor																					N
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
0	B	B	B	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
7	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1
8	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
10	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
12	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
13	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
14	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
15	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
16	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
17	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-			
18	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-			
19	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-			
20	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-			
21	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-			2
22	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-			
23	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-			
24	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-				
25	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-				
26	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-				3
27	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-				
28	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-				
29	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-				
30	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-					
31	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-					
32	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-					4
33	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-					
34	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-					

Bereich (N)	Schlüsseltabelle																				
N	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	00	17	13	09	05	01	18	14	10	06	02	19	15	11	07	03	20	16	12	08	04
2	00	04	08	12	16	01	05	09	13	17	02	06	10	14	18	03	07	11	15		
3	00	11	04	15	08	01	12	05	16	09	02	13	06	17	10	03	14	07			
4	00	07	14	04	11	01	08	15	05	12	02	09	16	06	13	03	10				

. von CP/M verwendet
B Ladesektor

8.4 Überblick über die Organisation des CP/M-Betriebssystems

Der 8502 und Z80 bearbeiten einige einfache Funktionen unter CP/M 3.0 gemeinsam, um die Geschwindigkeit und den System-Code zu optimieren. Die folgenden Funktionen werden in den folgenden Abschnitten näher erläutert:

- Laden (boot) von CP/M 3.0,
- Abfragen der Tastatur,
- Einstellen des 40/80-Spalten-Bildschirms,
- Drucker-Interface am seriellen Bus,
- Zeichen vom RS232C-Interface holen (mit XON/XOFF),
- Zeichen an das RS232C-Interface senden,
- Lesen eines Sektors von der Diskette (schnell und langsam),
- Diskette formatieren,
- CCP vom TPA (100H) in den Hintergrund-RAM kopieren,
- CCP vom Hintergrund-RAM in den TPA (100H) kopieren,
- Zeit im Betriebssystem setzen und
- Zeit im Betriebssystem aktualisieren,
- Speicherbereich in einen anderen Speicherbereich übertragen.

8.4.1 Blockübertragung

Es gibt drei verschiedene durchzuführende Blockoperationen:

Unterstützung der Disketten, der RAM-Diskette und Übertragen des CCP von und in einen Puffer.

Der Z80 hat eine Routine, mit der man Daten von einer Adresse und Bank an eine andere Adresse und Bank übertragen kann. Diese Routine beruht auf der Tatsache, daß die MMU von Seite 0 und Seite 1 auf jede Seitengrenze innerhalb des Betriebssystems zeigen kann. Sie verwendet die LDIR-Anweisung, wobei die Übertragung in Blöcke von 256 Byte oder weniger unterteilt ist. Diese Routine ist ein Hauptteil des CP/M-3.0-Betriebssystems. Alle anderen Routinen verwenden diese Routine für Blockbewegungen zwischen den Banken.

Es gibt zwei verschiedene Diskettenlaufwerke für den C 128, das langsame Laufwerk 1541 und das neue schnelle Laufwerk 1571 bzw. 1570. Das 8502-BIOS (BIOS unter dem 8502) legt fest, welches Laufwerk angeschlossen ist und ruft danach die passenden KERNAL-Routinen auf. Alle Datenüber-

tragungen vom 8502-BIOS werden über einen 256-Byte-Puffer durchgeführt. Der Z80 ist dann für die Datenübertragung in den eigenen Speicherbereich (Bank und Adresse) zuständig.

8.4.1.1 Laden von CP/M 3.0

Beim C 128 wird das CP/M 3.0 selbsttätig geladen, indem die erste Spur auf der Diskette gelesen und geprüft wird, ob sich ein Ladesektor (boot sector) darauf befindet, wobei die ersten drei Byte »CBM« lauten müssen. Ist ein Ladesektor vorhanden, wird er in den Speicher ab Adresse $1000 (CPU 8502) geladen und ausgeführt. Der CP/M-Ladesektor erfüllt diese Anforderung und beginnt mit dem Ladevorgang des 8502-BIOS. Die Kontrolle wird danach an das 8502-BIOS übergeben. Es legt die Z80-Speicherbelegung an und übergibt die Kontrolle an den Z80 (an die Speicheradresse 0000H, Bank 0). Das CP/M-ROM übernimmt die Kontrolle und lädt das CP/M-Betriebssystem von der Diskette, wobei das 8502-BIOS für die Diskettenfunktionen verwendet wird. Nach dem Laden von CP/M erfolgt ein CP/M-Kaltstart. Die Kaltstart-Routine liest die Datei CCP.COM und speichert diese an der Adresse 100H in der Bank 1. Eine Kopie der Datei CCP.COM wird dann im Speicher abgelegt, so daß bei einem Warmstart CCP.COM nicht mehr von Diskette geladen werden muß, sondern nur vom Speicher. Wenn CCP im TPA geladen ist, wird die Kontrolle an die Adresse 100H in der Bank 1 übergeben.

Der Ladesektor (Tabelle 8.7) wird wie folgt verarbeitet:

1. Lesen des Diskettenblocks von Spur 1, Sektor 0 in den RAM an die Adresse $0B00 (»TBUFFR«).

2. Prüfen auf »Selbstladen«, Rücksprung falls nicht.

3. Falls #BLK>0, erfolgt blockweises, sequentielles Lesen der Sektoren in den RAM an die Adressen ADRL, ADRH, BANK.

4. Ist ein Dateiname vorhanden, wird diese Datei ins RAM der Bank 0 geladen (normaler Ladevorgang).

5. Danach erfolgt ein Sprung zum (nach dem Dateinamen stehenden) Code der geladenen Datei.

Tabelle 8.7: CP/M-Ladesektor

Sektor	Inhalt
$00	C
$01	B
$02	M
$03	(ADRL)
$04	(ADRH)
$05	(BANK)
$06	(#BLK)
.	
.	(Diskettenname)
.	
$0A	0
.	
.	(Dateiname)
.	
$0B	0
.	
.	(6502-Code)
.	
$FF	

8.4.1.2 Lesen eines Sektors von einer Diskette

Vor dem Aufruf dieser Routine legt CP/M Spur, Sektor, DMA-Adresse (direct memory address = absolute Speicheradresse), Bank und Sektornummer fest. Spur und Sektor werden aus den Daten im Disketten-Parameter-Block DPB (disk parameter block) berechnet, d.h., für das neue Diskettenformat ist der Sektor immer Null und die Spur ist die Blocknummer relativ zum Diskettenanfang. Das Z80-BIOS überträgt diese Werte in die eigentliche Spur- und Sektornummer für das Diskettenlaufwerk. Beim alten C-64-CP/M-Format müssen Spur- und Sektornummern von CP/M nicht übertragen werden.

Bei den neuen, schnellen Laufwerken 1571 und 1570 bestimmt das Z80-BIOS die Anzahl der nachfolgenden Sektoren auf derselben Spur. Das 8502-BIOS legt die mehrfache Sektorübertragung fest. Die Kontrolle wird dann zurück ans Z80-BIOS gegeben, und der Z80 liest dann die Diskettendaten

direkt aus dem Ein-/Ausgabepuffer und schreibt sie in den Speicher an die DMA-Adresse im Speicher. Das 8502-BIOS ist verantwortlich für die Anzeige von Fehlern und für die Rückverbindung der Diskette mit dem Z80.

Beim alten Laufwerk 1541 erfolgt das Lesen entsprechend dem Schreiben eines Sektors (siehe Abschnitt 8.4.1.3).

8.4.1.3 Schreiben eines Sektors auf eine Diskette

Vor dem Aufruf dieser Routine legt CP/M Spur, Sektor, DMA-Adresse (direct memory address = absolute Speicheradresse), Bank und Sektornummer fest. Spur und Sektor werden aus den Daten im Disketten-Parameter-Block DPB (disk parameter block) berechnet, d.h., für das neue Diskettenformat ist der Sektor immer Null, und die Spur ist die Blocknummer relativ zum Diskettenanfang. Das Z80-BIOS überträgt diese Werte in die eigentliche Spur- und Sektornummer für das Diskettenlaufwerk.

Bei den neuen, schnellen Laufwerken 1571 und 1570 zählt das Z80-BIOS die Anzahl der nachfolgenden Sektoren auf derselben Spur und übergibt diese Information an das 8502-BIOS. Das 8502-BIOS legt dann die Übertragung dieser Sektoren fest und übergibt an das Z80-BIOS. Das Z80-BIOS liest dann die Daten aus der aktuellen DMA-Adresse und schreibt diese Daten direkt an den Ein-/Ausgabeport.

Beim alten Laufwerk 1541 schreibt das Z80-BIOS nur einen Sektor auf einmal, indem der Inhalt des DMA-Puffers in den 256-Byte-Disketten-Ein-/Ausgabepuffer übertragen wird. Danach wird das 8502-BIOS aufgerufen und der Puffer auf die Diskette geschrieben. Das 8502-BIOS prüft weiterhin, ob eine Diskette gewechselt wurde und gibt dann die zugehörigen Fehlermeldungen aus.

8.4.1.4 Kopieren des CCP von 100H ins Hintergrund-RAM

Diese Funktion wird nur beim ersten Laden des CP/M-Betriebssystems durchgeführt. Die Datei CCP.COM wird von der Diskette in den TPA geladen. Im allgemeinen wird die Datei CCP.COM jedesmal bei einem Warmstart, d.h. bei einer Rückmeldung mit der CP/M-Einschaltmeldung »A>«, wieder von der Diskette gelesen. Das Neuladen des CCP wird oft vorgenommen, wodurch das Betriebssystem infolge zahlreicher Diskettenzugriffe verlangsamt wird, so daß es nützlich ist, eine Kopie von CCP im Bankspeicher bereitzuhalten. Diese Routine veranlaßt das Speichern einer Kopie der Datei CCP.COM bei ihrem ersten Laden. Diese Kopie des CCP

wird dann an die Speicheradresse 100H in der Bank 1, dem TPA-Speicher, bei einem Warmstart geladen. Diese Funktion wird vollständig vom Z80 durchgeführt.

8.4.1.5 Kopieren des CCP vom Hintergrund-RAM

Diese Funktion wird bei einem Warmstart des CP/M-Betriebssystems aufgerufen. Im Abschnitt 8.4.1.4 sind weitere Einzelheiten dieser Funktion zu finden. CCP wird aus seinem RAM-Puffer nach 100H in der Bank 1 kopiert. Diese Funktion wird vollständig vom Z80 durchgeführt.

8.4.1.6 Formatieren einer Diskette

Mit dieser Funktion kann eine Diskette formatiert werden. Das Formatierprogramm kann zwei verschiedene Diskettenformate erzeugen, das alte C-64-CP/M-Format und das neue C-128-CP/M-Format.

Für das alte Format kann das Formatierprogramm wahlweise entweder die C-64-CP/M-Systemspuren von einer anderen Diskette kopieren oder einfach die Systemspuren leer belassen; die Spur 18 für das C-64-Disketten-Inhaltsverzeichnis wird ebenfalls angelegt.

Für das neue C-128-CP/M-Format werden die Ladesektoren auf der neuen Diskette erzeugt und der Bereich des Disketten-Inhaltsverzeichnisses formatiert. Zusätzlich können die Dateien des CP/M3-Betriebssystems CPM.SYS und/oder CCP.COM auf die neuformatierte Diskette kopiert werden.

8.4.2 Zeichenübertragung

Die Übertragung von Zeichen wird vorzugsweise vom Mikroprozessor 8502 durchgeführt. Die wesentliche Ausnahme ist die 40/80-Zeichen-Bildschirmanzeige durch den Mikroprozessor Z80.

8.4.2.1 Tastatur

Die Tastatur-Abfrage-Routine tastet die Tastatur zyklisch ab und gibt den Zeichen-Code der gedrückten Taste zurück. Ist keine Taste gedrückt, ist der Code »FFH«. Der Tastatur-Abfrage-Code ist außerdem verantwortlich für die Behandlung der programmierbaren Tasten und der programmierbaren Funktionstasten, für das Festlegen der Zeichen und Hintergrundbild-

schirmfarben, für die Auswahl der Diskettenformate und für die Auswahl des aktuellen Bildschirmtyps.

Die Änderung der Tastaturbelegung ist ausführlich in Abschnitt 4.6 beschrieben.

8.4.2.2 Einstellen des 40/80-Zeichen-Bildschirms

Es sind zwei völlig verschiedene Bildschirm-Anzeigesysteme im C 128 eingebaut.

Einmal erzeugt der Videochip VIC 8566 eine Anzeige von 25 Zeilen mit 40 Spalten mit vielen Graphikmöglichkeiten, bei der ein normales Farbfernsehgerät oder ein Farbmonitor verwendet werden kann. Unter CP/M wird hierbei der Normalmodus verwendet, wobei jedes Zeichen und der Bildschirmhintergrund in 16 Farben dargestellt werden können.

Das zweite Bildschirm-Anzeigesystem kann nur mit einem Monitor (Schwarzweiß oder Farbe) arbeiten und verwendet den Videocontroller VDC 8563. Das Anzeigeformat dieses Controllers ist 24 Zeilen mit je 80 Zeichen und der Möglichkeit farbiger Zeichen.

Das VIC-Chip verwendet einen Bildschirmspeicher, während der VDC von Ein-/Ausgabesequenzen gesteuert wird. Die beiden Bildschirm-Anzeigesysteme werden als zwei vollständig getrennte Anzeigen behandelt. CP/M 3.0 kann beide oder nur eine als Konsolenausgabeeinheit festlegen. Beide Anzeigen werden von einem gewöhnlichen Terminal-Emulationsprogramm gesteuert, normalerweise ein Treiberprogramm für ADM-3A.

Das Anzeigesystem wird ausschließlich vom Z80-BIOS gesteuert. Das Emulationsprogramm ist so geschrieben, daß der Benutzer die Emulation ADM-3A durch verschiedene andere Terminal-Emulierungen ersetzen kann.

Das Terminal-Treiberprogramm ist zweiteilig: die Terminal-Emulation und die Terminal-Funktionen.

Zwei verschiedene Emulationen stehen direkt zur Verfügung: ADM-3A und ADM-31. Die Terminal-Emulation ist Teil des Z80-BIOS, die Terminal-Funktionen befinden sich im wesentlichen im Z80-ROM.

In Tabelle 8.8 ist das Terminal-Emulationsprotokoll für den Lear-Siegler ADM-3A und in Tabelle 8.9 das für den Lear-Siegler ADM-31 beschrieben.

Tabelle 8.8: Terminal Lear-Siegler ADM-3A

Funktion	Zeichenfolge	Hex-Code
Cursorposition	<ESC> = (Zeile+32) (Spalte +32)	1B 3D 20+ 20+
Cursor nach links	^H	08
Cursor nach rechts	^L	0C
Cursor nach unten	^J	0A
Cursor nach oben	^K	0B
Cursor in Home-Position, Bildschirm löschen	^Z	1A
RETURN <CR>	^M	0D
Escape	^[	1B
Glocke	^G	07

Bildschirmanzeige: 24 Zeilen (1-24)
80 Spalten (1-80)

Home-Position des Cursors: Zeile 1, Spalte 1

Tabelle 8.9: Terminal Lear-Siegler ADM-31

Funktion	Zeichenfolge	Hex-Code
Löschen bis Zeilenende	<ESC>T oder	1B 54
	<ESC>t	1B 74
Löschen bis Bildschirmende	<ESC>Y oder	1B 59
	<ESC>y	1B 79
Cursor in Home-Position	<ESC>: oder	1B 3A
und Bildschirm löschen	<ESC>*	1B 2A
Halbe Intensität ein	<ESC>)	1B 29
Halbe Intensität aus	<ESC>(	1B 28
Revers-Darstellung ein	<ESC>G4	1B 47 34
Blinkender Cursor ein	<ESC>G2	1B 47 32
Unterstreichen ein (*)	<ESC>G3	1B 47 33
ALT-Zeichensatz auswählen (*)	<ESC>G1	1B 47 31
Revers-Darstellung und blinkender Cursor aus	<ESC>G0	1B 47 30
Zeile einfügen	<ESC>E	1B 45
Zeichen einfügen	<ESC>Q	1B 51
Zeile löschen	<ESC>R	1B 52
Zeichen löschen	<ESC>W	1B 57
Bildschirmfarben setzen	<ESC ESC ESC>#	1B 1B 1B xx

Es bedeuten:

* keine normale ADM-31-Funktion

\# 20H bis 2FH - Zeichenfarbe,
30H bis 3FH - Hintergrundfarbe
40H bis 4FH - Rahmenfarbe (nur beim 40-Zeichen-Bildschirm)

Bildschirmanzeige:

24 Zeilen (1-24)
80 Spalten (1-80)

Home-Position des Cursors:

Zeile 1, Spalte 1

Tabelle 8.10: Terminal VT 52

Funktion	Zeichenfolge	Hex-Code
Cursor nach oben	<ESC>A	1B 41
Cursor nach unten	<ESC>B	1B 42
Cursor nach rechts	<ESC>C	1B 43
Cursor nach links	<ESC>D	1B 44
Graphikmodus ein	<ESC>F	1B 46
Graphikmodus aus	<ESC>G	1B 47
Cursor in Home-Position	<ESC>H	1B 48
Zeilenvorschub (revers)	<ESC>I	1B 49
Löschen bis Bildschirmende	<ESC>J	1B 4A
Löschen bis Zeilenende	<ESC>K	1B 4B
Cursorposition (Adressierung)	<ESC>Y (Zeile+1FH) (Spalte +1FH)	1B 59 1F+ 1F+
Identifizierung	<ESC>Z	1B 5A
ALT-Tastatur-Modus ein	<ESC>=	1B 3D
ALT-Tastatur-Modus aus	<ESC>>	1B 3E
VT-100-Modus ein	<ESC><	1B 3C

Bildschirmanzeige:	24 Zeilen (1-24), 80 Spalten (1-80)
Home-Position des Cursors:	Zeile 1, Spalte 1

Tabelle 8.11: Terminal VT 100
(Untermenge von ANSI X3.64-1979 und X3.41-1974)

Funktion	Zeichenfolge	Hex-Code
Cursor-Darstellung (Video-Attribute)	⟨ESC⟩ x;...;xm	1B 5B xx 3B..3B xx 6D
normal	mit x = 0	xx = 30
fett	1	31
unterstrichen	2	32
blinken	3	33
revers	4	34
Zeichensätze	G0-Satz G1-Satz	
UK	⟨ESC⟩(A ⟨ESC⟩)A	1B 28 41 1B 29 41
ASCII	⟨ESC⟩(B ⟨ESC⟩)B	1B 28 42 1B 29 42
Sonder-Graphik	⟨ESC⟩(0 ⟨ESC⟩)0	1B 28 30 1B 29 30
ALT-ROM	⟨ESC⟩(1 ⟨ESC⟩)1	1B 28 31 1B 29 31
ALT-ROM-Sondergraphik	⟨ESC⟩(2 ⟨ESC⟩)2	1B 28 32 1B 29 32
Zeichengröße		
Doppelte Höhe, obere Hälfte	⟨ESC⟩#3	1B 23 33
Doppelte Höhe, untere Hälfte	⟨ESC⟩#4	1B 23 34
Einfache Breite, einfache Höhe	⟨ESC⟩#5	1B 23 35
Doppelte Breite, einfache Höhe	⟨ESC⟩#6	1B 23 36
Cursor-Bewegungen		
Position des Cursors	⟨ESC⟩ Zeile;SpalteH	1B 5B 30+ 3B 30+ 48
	⟨ESC⟩ Zeile;Spaltef	1B 5B 30+ 3B 30+ 66
Cursor nach links	⟨ESC⟩ SpalteD	1B 5B 30+ 44
Cursor nach rechts	⟨ESC⟩ SpalteC	1B 5B 30+ 43
Cursor nach unten	⟨ESC⟩ ZeileB	1B 5B 30+ 42
Cursor nach oben	⟨ESC⟩ ZeileA	1B 5B 30+ 41
Index	⟨ESC⟩D	1B 44
Neue Zeile	⟨ESC⟩E	1B 45
Reverser Index	⟨ESC⟩M	1B 4D
Cursor und Attribute speichern	⟨ESC⟩7	1B 37
Cursor und Attribute zurückholen	⟨ESC⟩8	1B 38
Löschen		
– bis Zeilenende	⟨ESC⟩ K	1B 5B 4B
– bis Bildschirmende	⟨ESC⟩ J	1B 5B 4A
– vom Zeilenanfang	⟨ESC⟩ 1K	1B 5B 31 4B
– vom Bildschirmanfang	⟨ESC⟩ 1J	1B 5B 31 4A
– der Cursor-Zeile	⟨ESC⟩ 2K	1B 5B 32 4B
– des ganzen Bildschirms	⟨ESC⟩ 2J	1B 5B 32 4A
LED	⟨ESC⟩ x;...;xq	1B 5B xx 3B..3B xx 71
alle löschen	x = 0	xx = 30
L1 ein	1	31
L2 ein	2	32
L3 ein	3	33
L4 ein	4	34

Funktion	Zeichenfolge	Hex-Code
Modus		
Einschalten	⟨ESC⟩ xh	1B 5B xx 68
– neue Zeile	mit x = 0	xx = 30
– Anwendung	1	31
– n/a	2	32
– 132 Spalten	3	33
– Sanftes Rollen	4	34
– reverse Bildschirm-Anzeige	5	35
– Original-Bildschirm, relativ	6	36
– Zeile, falten, ein	7	37
– Automatisches Wiederholen, ein	8	38
– Verketten, ein	9	39
Modus		
Ausschalten	⟨ESC⟩ xl	1B 5B xx 6C
– Zeilenvorschub	mit x = 0	xx = 30
– Cursor	1	31
– VT 52	2	32
– 80 Spalten	3	33
– Sprung-Rollen	4	34
– normale Bildschirm-Anzeige	5	35
– Original-Bildschirm, absolut	6	36
– Zeilen falten, aus	7	37
– Automatisches Wiederholen, aus	8	38
– Verketten, aus	9	39
– Tastatur		
– Anwendungsmodus	⟨ESC⟩ =	1B 3D
– numerischer Modus	⟨ESC⟩)	1B 3E
Rückmeldungen		
Cursor-Position		
– Aufruf	⟨ESC⟩ 6n	1B 5B 36 6E
– Antwort	⟨ESC⟩ Zeile;SpalteR	1B 5B xx 3B xx 52
Status		
– Aufruf	⟨ESC⟩ 5	1B 5B 35
– Antwort:		
Terminal OK	⟨ESC⟩ On	1B 5B 30 6E
Terminal nicht OK	⟨ESC⟩ 3n	1B 5B 33 6E
Terminal-Parameter		
– Aufruf	⟨ESC⟩	1B
– Antwort	⟨ESC⟩	1B
Terminal VT-100		
– Aufruf	⟨ESC⟩ c oder	1B 5B 63 oder
	⟨ESC⟩Z	1B 5A
– Antwort, wenn VT 100-Terminal	⟨ESC⟩ ? ;Oc	1B 5B 3F 5D 3B 30 63
Reset		
(Initialisieren d. Status)	⟨ESC⟩c	1B 63

Funktion	Zeichenfolge	Hex-Code
Rollen eines Bereichs	⟨ESC⟩ t;br mit t = oberste Zeile b = unterste Zeile	1B 5B 30+ 3B 30+ 72
Tabulator		
– Setzen	⟨ESC⟩H	1B 48
– Löschen	⟨ESC⟩g	1B 67
– alles löschen	⟨ESC⟩3g	1B 33 67
Test (Füllen des Bildschirms mit E)	⟨ESC⟩#8	1B 23 38
Null		00
Glocke	^G	07
HT	^I	09
LF	^J	0A
VT (wie LF)	^K	0B
FF (wie LF)	^L	0C
RETURN ⟨CR⟩	^M	0D
SO (G1-Zeichen-Satz)	^N	0E
SI (G0-Zeichen-Satz)	^O	0F
CAN (Escape-Sequenz löschen)	^X	18
SUB (wie CAN)	^Z	1A

Bildschirmanzeige: 24 Zeilen (1–24), 80 Spalten (1–80).
Home-Position: Zeile 1, Spalte 1.

8.4.2.3 Serieller Commodore-Bus

Der serielle Commodore-Bus ist nicht sehr schnell, deshalb wurden zwei Drucker-Treiberprogramme im CP/M-System installiert. Die RS232C-Schnittstelle kann an jeden Drucker mit einer solchen Schnittstelle angeschlossen werden, wobei diese dann aber nicht mehr für andere Operationen verwendet werden kann. Das Treiberprogramm wird im Abschnitt 8.4.2.4 beschrieben. Hier sollen die Drucker mit dem seriellen Commodore-Bus beschrieben werden. Alle diese Drucker arbeiten mit zwei speziellen C=-Zeichensätzen, dem Großschrift-/Graphikmodus und dem Klein-/Großschriftmodus, einige auch mit einem normalen ASCII-Zeichensatz. Deshalb müssen die eigentlichen CP/M-ASCII-Zeichen in die speziellen C=-Zeichen umgewandelt werden, ehe sie zum Drucker gesendet werden. Einige Drucker können nur Großschrift-/Graphikzeichen ausgeben, so daß dieser Modus bevorzugt ist. Das Z80-BIOS führt alle diese Umwandlungen durch und überträgt alle Zeichen an den 8502, um sie dann an den Drucker zu schicken. Der 8502 macht eine Pause, wenn der Drucker fehlt oder nicht eingeschaltet ist, so daß das Betriebssystem sich nicht aufhängt.

Folgende Drucker werden unterstützt:

MPS-801	Nadel-Matrix-Drucker (50 Zeichen/Sekunde)
MPS-802	Nadel-Matrix-Drucker (80 Zeichen/Sekunde)
MPS-803	Nadel-Matrix-Drucker (60 Zeichen/Sekunde)
DPS-1101	Typenraddrucker (normaler ASCII-Zeichensatz)

8.4.2.4 Holen eines Zeichens von der RS232C-Schnittstelle (mit XON/XOFF)

Für den RS232C-Eingang erfolgt die Umwandlung der seriellen Daten in parallele Daten durch die Mikroprozessoren. Das KERNAL des Mikroprozessors 8502 enthält Routinen zum Empfang der Bits und zum Erzeugen der Zeichen, so daß hierzu der Mikroprozessor 8502 eingeschaltet werden muß. Zum Holen eines Zeichens wird die XON/XOFF-Logik verwendet. Wenn der Z80 ein Zeichen benötigt, fordert er vom 8502 ein Zeichen an. Der 8502 sendet ein XON-Zeichen. Nachdem das erste Zeichen empfangen wurde, wird ein XOFF gesendet. Der 8502 wartet eine Periode für ein komplettes Zeichen nach dem Senden eines XOFF oder bis zum Empfangen des nächsten Zeichens, ehe der Z80 wieder eingeschaltet wird. Jedes Zeichen nach dem ersten Zeichen wird in einen Puffer geschrieben. Fordert der Z80 das nächste Mal ein Zeichen, wird es, falls eins im Puffer ist, an den Z80 übergeben, ohne daß der Kanal mit XON wieder eingeschaltet wird.

8.4.2.5 Senden eines Zeichens an die RS232C-Schnittstelle

Der RS232C-Ausgang ist einfacher als der -Eingang (siehe Abschnitt 8.4.2.4). Der Z80 sendet das Zeichen an den 8502, und der 8502 behält die Kontrolle so lange, bis das Zeichen vollständig - einschließlich der Stop-Bits - über die Schnittstelle ausgegeben wurde. Dann wird die Kontrolle zurück an den Z80 gegeben. Ein Spezialfall ist der, daß ein XOFF gesendet wird, ehe das erste Zeichen ausgegeben wird, so daß das Gerät, mit dem kommuniziert werden soll, nicht senden kann. Außerdem kann das Gerät, mit dem kommuniziert werden soll, ein XON oder XOFF senden. Um einen Zeichenverlust zu vermeiden, muß in der Zeichensenderoutine (Send Char) so lange geblieben werden, bis ein Steuerzeichen empfangen wurde. Einige Geräte senden überhaupt kein XON/XOFF, so daß eine zusätzliche Verzögerungszeit nicht notwendig ist. Deshalb kann die zusätzliche Verzögerungszeit im Menü ausgewählt werden. Die Wirkung der zusätzlichen XON/XOFF-Verzögerung entspricht etwa einer Halbierung der effektiven Baudrate, d.h., eine Baudrate von 1200 hat dann etwa eine effektive Baudrate von 600.

8.4.2.6 Einstellen der RS232C-Parameter

Die Baudrate kann zwischen 110 bis 1200 Baud eingestellt werden. Diese Funktion ändert die Baudrate auf den eingestellten Wert und sendet ein XOFF-Zeichen an das angeschlossene Gerät.

8.4.3 Operationen des Rechners

8.4.3.1 Einstellen der Rechneruhr

Mit dieser Funktion wird die Tageszeit eingestellt. Die Tageszeit wird im gepackten BCD-Format in den System-Kontroll-Block SCB (system control block) an drei Adressen (Stunden, Minuten, Sekunden) gespeichert. Diese Routine liest die SCB-Zeit und addiert diese Zeile zur Zeit der Uhr des 6526-Chips. Diese Zeit wird dauernd in diesem Chip aktualisiert und von CP/M verwendet. Der Z80 kann direkt den 6526-Chip einstellen und lesen.

8.4.3.2 Rechneruhr

Die SCB-Zeit wird von der Rechneruhr im 6526-Chip gesteuert. Diese Funktion wird direkt vom Z80 durchgeführt.

8.4.3.3 Kopieren von Speicherbereichen

Diese Routine ermöglicht die Benutzung der zusätzlichen RAM-Bänke. Diese Routine übergibt die Quelladresse und -banknummer, die Zieladresse und -banknummer und die Anzahl der zu übertragenden Bytes. Die Übertragung wird mit Hilfe der zwei MMU-Seitenzeiger (page-pointer) und der Z80-Block-Übertragung (blockmove) durchgeführt.

Es gibt drei grundsätzliche Fälle:

Der erste ist die seitenorientierte 256-Byte-Übertragung, der zweite eine seitenorientierte Übertragung von weniger als 256 Byte. Der dritte Fall liegt dann vor, wenn die acht LSBs (niederwertigsten Bits) nicht gleich sind, so daß die effektiven Adressen zweimal für jede 256-Byte-Block-Übertragung berechnet werden müssen. Der dritte Fall sollte im allgemeinen nicht auftreten, aber auf ihm sollte geprüft und beim Auftreten entsprechend gehandelt werden.

Die häufigsten Speicherübertragungen sind vom ersten Fall, weshalb diese Routine in der Geschwindigkeit optimiert ist. Der zweite und dritte Fall

tritt weniger häufig auf, so daß diese nicht sehr schnell sein müssen. Der Code dieser Routine muß im COMMON-Speicherbereich liegen, da Bank 0 nicht mit ihm arbeiten kann, wenn er aufgerufen wird. Der Grund dafür ist, daß das Z80-CP/M-ROM die Seiten (page) 0 und 1 überlagert, wenn die Bank 0 gewählt wird.

8.5 Struktur des 8502-BIOS

Der 8502 ist verantwortlich für die meisten einfachen Ein-/Ausgabefunktionen. Der Aufruf dieser Funktionen erfolgt mit Hilfe eines Satzes von Kommunikationsregistern. Wenn die Kommunikationsregister angesprochen werden, schaltet sich der Z80 aus, und der 8502 schaltet sich ein (8502-BIOS). Der 8502 analysiert den Befehl im Kommunikationsregister, führt die angeforderte Aufgabe durch, stellt den Befehlsstatus ein und schaltet um. Der Z80 ist wieder eingeschaltet, analysiert den Befehlsstatus und übernimmt die entsprechenden Aufgaben.

8.6 Struktur des Z80-CP/M-BIOS

Die Größe des BIOS ist im Gegensatz zum BDOS und CCP nicht vorgegeben, da die einzelnen Treiberroutinen und Puffer je nach Rechner (Hardware) unterschiedlich lang sein können. Das BIOS muß allerdings bei einer bestimmten Adresse beginnen. Das BDOS erwartet am Anfang des BIOS eine Reihe von Sprungbefehlen zu den einzelnen BIOS-Routinen, die sogenannte BIOS-Sprungliste (s. Tabelle 8.12). Damit ist für eine eindeutige Schnittstelle zwischen BDOS und BIOS gesorgt, ohne daß Reihenfolge und Länge der einzelnen BIOS-Routinen festgelegt sind.

Die insgesamt 32 BIOS-Einsprünge lassen sich in vier Gruppen einteilen:

- Initialisierung,
- Ein-/Ausgabe einzelner Zeichen,
- Diskettenzugriffe und
- Speicherbereichsverschiebungen.

Tabelle 8.12: Z80-BIOS-Funktionen

Funktion	Name	Beschreibung	
0	BOOT	Kaltstart	
		Bank:	0
		Eingabe:	-
		Ausgabe:	-
		Funktion:	Rechner initialisieren, Zero-Page einrichten, CP/M-Einschaltmeldung ausgeben, CCP laden und dann Kontrolle an CCP übertragen.
1	WBOOT	Warmstart	
		Bank:	0 oder 1
		Eingabe:	-
		Ausgabe:	-
		Funktion:	Zero-Page einrichten, CCP neu laden und CCP starten, d.h. den Stapelspeicher in den COMMON-Speicher zurückschreiben, um BDOS-Aufrufe durchzuführen.
2	CONST	Konsolenstatus	
		Bank:	0 oder 1
		Eingabe:	-
		Ausgabe:	A
		Funktion:	Den Konsoleneingabestatus der aktuellen Konsoleneinheit prüfen. A = FFH, wenn ein Zeichen ansteht; A = 00H, wenn kein Zeichen vorhanden.

Funktion	Name	Beschreibung
3	CONIN	Konsoleneingabe Bank: 0 oder 1 Eingabe: - Ausgabe: A (ASCII-Zeichen) Funktion: Ein Zeichen von der eingeschalteten Konsoleneingabeeinheit lesen. Jede eingeschaltete Einheit wird abgetastet, bis ein Eingabezeichen gefunden wird.
4	CONOUT	Konsolenausgabe Bank: 0 oder 1 Eingabe: C (ASCII-Zeichen zum Ausgeben) Ausgabe: - Funktion: Das Zeichen in Register C wird an alle Einheiten gesendet, die gerade von der Konsole festgesetzt werden. Auf alle langsamen Einheiten wird gewartet.
5	LIST	Druckerausgabe Bank: 0 oder 1 Eingabe: C (ASCII-Zeichen zum Drucken) Ausgabe: - Funktion: Das Zeichen in C wird an alle gerade eingeschalteten LIST-Einheiten gesendet. Auf die langsameren Einheiten wird gewartet.

Funktion	Name	Beschreibung	
6	AUXOUT	Hilfsausgabe	
		Bank:	0 oder 1
		Eingabe:	C (ASCII-Zeichen an die Hilfseinheit senden)
		Ausgabe:	-
		Funktion:	Das Zeichen in C wird an alle eingeschalteten AUXOUT-Einheiten gesendet. Auf alle langsameren Einheiten wird gewartet.
7	AUXIN	Hilfseingabe	
		Bank:	0 oder 1
		Eingabe:	-
		Ausgabe:	A (ASCII-Zeichen von der Hilfseinheit holen)
		Funktion:	Ein Zeichen wird von der eingeschalteten AUXIN-Einheit gelesen. Jede eingeschaltete Einheit wird so lange abgefragt, bis ein Zeichen gefunden wird.
8	HOME	Laufwerk in die Grundstellung	
		Bank:	0
		Eingabe:	-
		Ausgabe:	-
		Funktion:	Der Schreib-/Lesekopf des Bezugslaufwerks wird in die Grundstellung gestellt, d.h. auf die Spur 0.
9	SELDSK	Laufwerk auswählen	
		Bank:	0
		Eingabe:	C (Diskettenlaufwerk 0-15, z.B. A = 0)
			E (Anfangs-Auswahl-Flag)
		Ausgabe:	HL (Adresse des Disketten-Parameter-Kopfes DPH)

Funktion	Name	Beschreibung	
		Funktion:	Es wird das Diskettenlaufwerk ausgewählt, dessen Adresse in C als Bezugslaufwerk für alle weiteren Operationen angegeben ist. Wenn LSB-Bit von Reg. E Null ist, dann wird diese Diskette zum ersten Mal angesprochen. Es wird auf das Diskettenformat (C 64- oder C-128-CP/M) geprüft und der Disketten-Parameter-Block DPB eingestellt. HL = 00H, wenn das Laufwerk nicht vorhanden ist.
10	SETTRK	Spur auswählen	
		Bank:	0
		Eingabe:	BC (Spurnummer 0-34)
		Ausgabe:	-
		Funktion:	Das Registerpaar BC enthält die Spurnummer für den nächsten Diskettenzugriff. Dieser Wert wird gespeichert.
11	SETSEC	Sektor auswählen	
		Bank:	0
		Eingabe:	BC (Sektornummer)
		Ausgabe:	-
		Funktion:	Das Registerpaar BC enthält die Sektornummer für den nächsten Diskettenzugriff. Dieser Wert wird gespeichert. Der Wert in BC ist der von der Sektor-Berechnungsroutine ausgegebene Wert.

Funktion	Name	Beschreibung	
12	SETDMA	Speicheradresse festlegen	
		Bank:	0
		Eingabe:	BC (Direkte Speicher-Zugriffs-adresse DMA)
		Ausgabe:	-
		Funktion:	Der Wert in BC wird als aktuelle DMA-Adresse gespeichert, d.h. die Adresse, in der jedes Disketten-Lesen und -Schreiben auftritt. Die eingestellte DMA Adresse wird so lange verwendet, bis sie mit einem späteren Aufruf dieser Funktion geändert wird.
13	READ	Sektor lesen	
		Bank:	0
		Eingabe:	-
		Ausgabe:	A
		Funktion:	Der Sektor, der von Bezugslaufwerk, Spur und Sektor adressiert wird, wird an die aktuelle DMA-Adresse gelesen. A = 00H, wenn kein Lesefehler auftritt. A = 01H, wenn ein Lesefehler trotz mehrmaligen Lesens auftritt. A = FFH, wenn die Diskette gewechselt wurde. Diese Prüfung wird bei jedem Aufruf dieser Routine durchgeführt.
14	WRITE	Sektor schreiben	
		Bank:	0
		Eingabe:	C (Entriegelungscode, nicht verwendet)
		Ausgabe:	A

Funktion	Name	Beschreibung	
		Funktion:	Der Sektor, der von Bezugslaufwerk, Spur und Sektor adressiert wird, wird an die aktuelle DMA-Adresse geschrieben. A = 00H, wenn kein Fehler auftritt. A = 01H, wenn ein Fehler trotz mehrmaliger Versuche auftritt. A = 02H, wenn eine schreibgeschützte Diskette beschrieben werden soll. A = FFH, wenn die Diskette gewechselt wurde. Diese Prüfung wird bei jedem Aufruf dieser Routine durchgeführt.
15	LISTST	Druckerstatus holen	
		Bank:	0 oder 1
		Eingabe:	-
		Ausgabe:	A
		Funktion:	Diese Routine prüft die derzeit aktuelle LIST-Einheit, ob ein Zeichen aufgenommen werden kann. A = 00H, wenn die LIST-Einheit nicht bereit ist. A = FFH, wenn die LIST-Einheit bereit ist.
16	SECTRN	Sektor übersetzen	
		Bank:	0
		Eingabe:	BC (logische Sektornummer 0-n) DE (Adresse der Übersetzungstabelle des Disketten-Parameter-Blocks DPB)
		Ausgabe:	HL (physikalische Sektornummer)

Funktion	Name	Beschreibung	
		Funktion:	Diese Routine übersetzt die logische Sektornummer in eine physikalische Sektornummer. Falls keine Übersetzung notwendig, wird BC in HL geschrieben.
17	CONOST	Konsolenausgabestatus	
		Bank:	0 oder 1
		Eingabe:	-
		Ausgabe:	A
		Funktion:	Diese Routine fragt die aktuelle Konsoleneinheit ab, ob die Bereitschaft zur Übernahme eines Zeichens besteht. A = FFH, wenn die Konsole bereit ist. A = 00H, wenn die Konsole nicht bereit ist.
18	AUXIST	Status der Eingabe-Hilfseinheit	
		Bank:	0 oder 1
		Eingabe:	-
		Ausgabe:	A
		Funktion:	Es wird der Status der aktuellen Eingabe-Hilfseinheit AUXIN geprüft, ob das Gerät ein Konsolenzeichen ausgibt. A = FFH, wenn ein Zeichen vorhanden ist. A = 00H, wenn kein Zeichen vorhanden ist.
19	AUXOST	Status der Ausgabe-Hilfseinheit	
		Bank:	0 oder 1
		Eingabe:	-
		Ausgabe:	A

Funktion	Name	Beschreibung	
		Funktion:	Es wird der Status der aktuellen Ausgabe-Hilfseinheit AUXOUT geprüft, ob das Gerät zur Aufnahme eines Zeichens bereitsteht. A = FFH, wenn AUXOUT bereit ist. A = 00H, wenn AUXOUT nicht bereit ist.
20	DEVTBL	Adresse der Definitionstabelle der Ein-/Ausgabeeinheiten	
		Bank:	0 oder 1
		Eingabe:	-
		Ausgabe:	HL (Adresse der Ein-/Ausgabe-Geräte-Tabelle)
		Funktion:	Es wird die Adresse der Definitionstabelle für die Ein-/Ausgabeeinheiten ausgegeben, um die Treiberprogramme zu benennen, die Baudrate und die XON/XOFF-Logik für alle Treiber einzustellen und zu prüfen. Die Treiberprogramme werden anstelle des I/O-Byte bei CP/M 2.2 verwendet.
21	DEVINI	Baudrate eines Gerätes ändern	
		Bank:	0 oder 1
		Eingabe:	C (Nummer der Geräteeinheit)
		Ausgabe:	-
		Funktion:	Es wird aus der DEVTBL-Tabelle die Baudrate der physikalischen Einheit in Register C initialisiert.

Funktion	Name	Beschreibung
22	DRVTBL	Adresse der Laufwerktabelle Bank: 0 Eingabe: - Ausgabe: HL (Adresse der Laufwerktabelle) Funktion: Es wird die Adresse der Laufwerktabelle ausgegeben; die erste Anweisung muß lauten LXI H,DRVTBL. Die Laufwerkstabelle ist eine Liste aus 16 Zeigern, die auf den zugehörigen Disketten-Parameter-Kopf DPH zeigen. Ist das Laufwerk nicht vorhanden, wird der Zeiger für dieses Laufwerk auf Null gesetzt.
23	MULTIO	Disketten-Lesen/Schreiben mit Mehrfach-Sektorzähler Bank: 0 Eingabe: C (Multisektor-Zähler) Ausgabe: - Funktion: Der Multisektor-Zähler wird eingestellt, ehe Spur, Sektor und DMA-Adresse adressiert und das Sektor-Lesen/Schreiben stattfindet. Maximal 16 K können von jedem Multisektor-Zähler übertragen werden. 1541 hat 256-Byte-Sektoren, so daß die Maximalzahl 64 ist. Bei der Entschlüsselung der Sektoren wird eine Liste mit der physikalischen Spur, Sektor und DMA-Adresse (für jeden Schreib-/Lesesektor) erzeugt. Die Liste ist nach Spur und Sektor sortiert. Danach wird der Lese-/Schreibvorgang durchgeführt. Diese Multisektorübertragung lohnt sich

Funktion	Name	Beschreibung	
			nur bei den neuen schnellen Diskettenlaufwerken 1571 und 1570.
24	FLUSH	Bank:	0
		Eingabe:	-
		Ausgabe:	A
		Funktion:	Diese Routine wird zum Ver-/Entriegeln (blocking/deblocking) im BIOS beim Diskettenbearbeiten verwendet. Da diese Routine hier nicht verwendet wird, ist immer A = 00H. A = 00H, kein Fehler. A = 01H, wenn ein Fehler trotz mehrmaliger Versuche auftritt. A = 02H, wenn die Diskette schreibgeschützt ist. A = FFH, wenn die Diskette gewechselt wurde.
25	MOVE	Datenblock im Speicher übertragen	
		Bank:	0
		Eingabe:	HL (Zieladresse) DE (Quelladresse) BC (Zähler)
		Ausgabe:	HL = HL (Eingabe) + BC (Eingabe) DE = DE(Eingabe) + BC(Eingabe)
		Funktion:	Es wird ein Datenblock im Speicher übertragen. Zu übertragende Daten werden in die (aus der) aktuelle(n) Speicherbank verschoben. Zusammen mit der Routine XMOVE (29) wird zwischen den Bänken übertragen.

Funktion	Name	Beschreibung	
26	TIME	Rechneruhr	
		Bank:	0 oder 1
		Eingabe:	C
		Ausgabe:	-
		Funktion:	Mit C = 00H wird die Zeit des Rechnersystems abgefragt; vorher muß mit dem System-Kontroll-block SCB die Uhr eingestellt worden sein. Mit C = FFH wird die Zeit mit dem SCB eingestellt. Die Register HL und DE sind hierbei reserviert.
27	SELMEM	Speicherbank wählen	
		Bank:	0 oder 1
		Eingabe:	A (Speicherbank)
		Ausgabe:	-
		Funktion:	Es wird die aktuelle Speicherbank gewählt. Dieser Code muß im COMMOM-Speicherbereich stehen.
28	SETBNK	Bank für Diskettenoperationen wählen	
		Bank:	0
		Eingabe:	A (DMA-Speicherbank)
		Ausgabe:	-
		Funktion:	Es wird die Bank für den nächsten Disketten-Lese-/ Schreibvorgang eingestellt.

Funktion	Name	Beschreibung
29	XMOVE	Quell- und Zielbank für eine Speicherübertragung Bank: 0 Eingabe: B (Zielbank) C (Quellbank) Ausgabe: - Funktion: Das System wird auf eine Speicherübertragung vorbereitet. Diese Funktion muß nicht verwendet werden; wenn sie fehlt, muß die erste Anweisung »RET« lauten.
30	USERF	für zukünftige Anwendungen reserviert.
31	RESERV1	für zukünftige Anwendungen reserviert.
32	RESERV2	für zukünftige Anwendungen reserviert.

Es bedeuten:

	Z80-CPU-Register
A	Akkumulator
BC	Doppelregister
DE	Doppelregister
HL	Doppelregister

8.7 Struktur einiger System-Parameter

8.7.1 System-Kontroll-Block SCB

Der System-Kontroll-Block SCB (system control block) hat 100 Byte. Er wird zur allgemeinen Kommunikation zwischen den verschiedenen Programmodulen des CP/M-3.0-Betriebssystems verwendet. Er besteht aus System-Parametern und Variablen.

8.7.2 Laufwerkstabelle DRVTBL

Die Laufwerkstabelle besteht aus einer Liste von 16 Zeigern (DPH0 bis DPH15) im reversen Byte-Format. Der erste Zeiger DPH0 steht für das Laufwerk A und der letzte DPH15 für das Laufwerk P. Die Zeiger zeigen auf den XDPH des zugehörigen Diskettenlaufwerks. Für jedes nicht vorhandene Laufwerk ist der Zeiger auf Null gesetzt.

8.7.3 Erweiterter Disketten-Parameter-Kopf DPH

Der erweiterte Disketten-Parameter-Kopf besteht aus dem normalen Disketten-Parameter-Kopf DPH (disk parameter header) mit einem zusätzlichen Vorspann (header). Der Aufbau ist in Tabelle 8.13 wiedergegeben, wobei die Reihenfolge der Parameter eingehalten wurde.

Tabelle 8.13: Erweiterter Disketten-Parameter-Kopf DPH für ein Laufwerk

Parameter	Bit	Inhalt
WRT	16	Adresse der Sektor-Schreib-Routine.
READ	16	Adresse der Sektor-Lese-Routine.
LOGIN	16	Adresse der LOGIN-Routine, um mit diesem Laufwerk den Bezug herzustellen.
INIT	16	Adresse der Routine zur ersten Initialisierung dieses Laufwerks.
TYPE	8	Byte für die Spurdichte und den Laufwerkstyp (in BIOS verwendet).
UNIT	8	Laufwerknummer relativ zum Diskettencontroller.
XLT	16	Adresse der Sektor-Übertragungstabelle; Null wenn keine Tabelle vorhanden.
-0-	72	BDOS-Arbeitsbereich.
MF	8	Medium-Flag wird auf 00H gesetzt, wenn der Bezug zur Diskette hergestellt wird und vom BIOS auf FFH, wenn die Diskette gewechselt wurde.
DPB	16	Zeiger zum aktuellen Disketten-Parameter-Block DPB, der den aktuellen Diskettentyp beschreibt.
CSV	16	Zeiger auf den Prüfsummenbereich des Disketten-Inhaltsverzeichnisses (für jedes Diskettenlaufwerk).
ALV	16	Zeiger auf den Zuweisungssektor-Bereich (einer pro Diskettenlaufwerk).

Parameter	Bit	Inhalt
DIRBCB	16	Zeiger auf den Puffer-Kontroll-Block BCB des Disketten-Inhaltsverzeichnisses.
DTABCB	16	Zeiger auf den Puffer-Kontroll-Block BCB einer Dateneinheit.
HASH	16	Zeiger auf eine Hash-Tabelle für ein optimales Disketten-Inhaltsverzeichnis. (FFFFH, wenn die Tabelle nicht verwendet wird.)
HBANK	8	Banknummer der obigen Hash-Tabelle.

8.7.4 Disketten-Parameter-Block DPB

Der Aufbau des Disketten-Parameter-Blocks DPB (disk parameter block) ist in Tabelle 8.14 wiedergegeben.

Tabelle 8.14: Disketten-Parameter-Block DPB

Parameter	Bit	Inhalt
SPT	16	Anzahl der 128 Records pro Spur.
BSH	8	Schiebefaktor des Daten-Zuweisungsvektors.
BLM	8	Block-Maske.
EXM	8	Extent-Maske
DSM	16	Anzahl der Zuweisungsblöcke auf der Diskette minus eins.
DRM	16	Anzahl der Einträge des Disketten-Inhaltsverzeichnisses minus eins.
AL0	8	Erstes Byte und
AL1	8	Zweites Byte des Vektors (MSB bis LSB) für den Zuweisungsblock des Disketten-Inhaltsverzeichnisses. Es können bis zu 16 Zuweisungsblöcke für das Disketten-Inhaltsverzeichnis verwendet werden.
CKS	16	Größe des Prüfvektors für das Disketten-Inhaltsverzeichnis; CKS = (DRM + 1)/4.
OFF	16	Anzahl der reservierten Spuren vom Diskettenanfang.
PSH	8	Schiebevektor für das physikalische Record.
PHM	8	Maske für das physikalische Record.

8.7.5 Puffer-Kontroll-Block BCB

Der Puffer-Kontroll-Block BCB (buffer control block) ist in Tabelle 8.15 dargestellt.

Tabelle 8.15: Puffer-Kontroll-Block BCB

Parameter	Bit	Inhalt
DRV	8	Laufwerk, das zu diesem Record gehört. Auf FFH gesetzt, wenn es nicht verwendet wird.
REC#	24	Absolute Sektornummer des Puffers.
WFLG	8	Byte wird auf FFH gesetzt, wenn der Puffer Daten enthält, die auf eine Diskette geschrieben werden müssen.
00	8	BDOS-Arbeitsbyte.
TRACK	16	Physikalische Spuradresse des Puffers.
SECTOR	16	Physikalische Sektoradresse des Puffers.
BUFFAD	16	Adresse des zu diesem BCB gehörenden Puffers.
BANK	8	Banknummer des zu diesem BCB gehörenden Puffers.
LINK	16	Adresse des nächsten BCB in dieser verketteten Liste. Die letzte ist Null.

Anhang A: Literaturverzeichnis

CP/M Plus (CP/M Version 3), Operating System User's Guide, 1982, Digital Research, Pacific Grove, CA 93950, USA.

CP/M Plus (CP/M Version 3), Betriebssystem, Benutzerhandbuch, 1. Ausgabe, Mai 1984, Digital Research, Pacific Grove, CA 93950, USA.

Commodore 128, Bedienungshandbuch, Kapitel 7: CP/M-Modus, 1985, Artikel-Nr. 580128, Commodore Büromaschinen GmbH, Lyoner Str. 38, D-6000 Frankfurt 71.

Commodore 128 Personal Computer, Technical Notes, 1985, Commodore Business Machines Inc, West-Cheste, PE 19380, USA.

Hückstädt, Jürgen: CP/M 3.0 Anwenderhandbuch C 128, Juni 1986, Verlag Markt & Technik, D-8013 Haar b. München.

Schieb, Weiler: Commodore 128, Das CP/M-Buch, Data Becker, Düsseldorf, 1985.

Schneider, Wolfgang: Programmieren von Mikrocomputern, Band 9, Einführung in die Anwendung des Betriebssystems CP/M, Verlag Friedrich Vieweg & Sohn, Braunschweig, 1983, ISBN 3-528-04252-4.

Rodnay, Zaks: CP/M-Handbuch, Sybex-Verlag, Düsseldorf, 1984, 5. Auflage, ISBN 3-88745-053-1

NEVADA EDIT, User's Reference Manual, 1983, Ellis Computing Inc, San Francisco, CA 94122, USA.

Microsoft BASIC Interpreter for 8080/8085 and Z80 Microprocessors and the CP/M-80, 1983, Operating System, Microsoft Corporation, Bellevue, Washington 98004.

Jack Jay Purdum: BASIC-80 and CP/M, Wie man Personal Computer programmiert - ein Lehrbuch, das Sie von den Anfängen bis zur sicheren Beherrschung von BASIC begleitet, Verlag Markt & Technik, D-Haar b. München, 1983, 296 Seiten, ISBN 3-922120-40-7.

Microsoft FORTRAN Compiler for CP/M-80, 1983, Microsoft Corporation, Bellevue, Washington 98004.

NEVADA FORTRAN, Version 3.0, Programmer's Reference Manual, 1983, Ellis Computing Inc, San Francisco, CA 94122.

TURBO-Pascal, Reference Manual, 1983, 254 S., Borland International Inc, Scotts Valley, CA 95066, USA.

Anhang B: Bezugsquellen-Nachweis

CP/M 3.0 für den Commodore 128: Commodore Büromaschinen GmbH, Lyoner Str. 38, D-6000 Frankfurt 71.

MBASIC (BASIC-Interpreter) und FORTRAN-80 (Compiler) von Microsoft: Markt & Technik Software Verlag, Hans-Pinsel-Str. 2, D-8013 Haar b. München.

NEVADA-EDIT und NEVADA-FORTRAN von Ellis Computing: TESCO GmbH, Postfach 10, D-8714 Wiesentheid.

TURBO-Pascal von Borland International: Heimsoeth Software, Postfach 14 02 80, D-8000 München 5.

Anhang C: Index

Spitzen-Software für Commodore 128/128 D

WordStar 3.0 mit MailMerge

Der Bestseller unter den Textverarbeitungsprogrammen für PCs bietet Ihnen bildschirmorientierte Formatierung, deutschen Zeichensatz und DIN-Tastatur sowie integrierte Hilfstexte. Mit MailMerge können Sie Serienbriefe mit persönlicher Anrede an eine beliebige Anzahl von Adressen schreiben und auch die Adreßaufkleber drucken.

WordStar/MailMerge für den Commodore 128 PC
Bestell-Nr. MS 103 (5¼"-Diskette)

Hardware-Anforderungen: Commodore 128 PC, Diskettenlaufwerk, 80-Zeichen-Monitor, beliebiger Commodore-Drucker oder ein Drucker mit Centronics-Schnittstelle

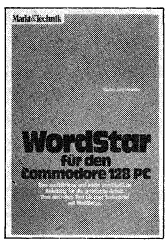
Für nur DM 199,–* (sFr. 178,–/öS 1890,–*)
***inkl. MwSt. Unverbindliche Preisempfehlung**

Markt&Technik
128er-Software

MicroPro
WordStar 3.0
mit MailMerge für den Commodore 128 PC

5¼"-Diskette
im Floppy 1541-Format

Und dazu die weiterführende Literatur:



Mit diesem Buch haben Sie eine wertvolle Ergänzung zum WordStar-Handbuch: Anhand vieler Beispiele steigen Sie mühelos in die Praxis der Textverarbeitung mit **WordStar** ein. Angefangen beim einfachen Brief bis hin zur umfangreichen Manuskripterstellung zeigt Ihnen dieses Buch auch, wie Sie mit Hilfe von MailMerge Serienbriefe an eine beliebige Anzahl von Adressen mit persönlicher Anrede senden können.

Best.-Nr. MT 780
ISBN 3-89090-181-6
DM 49,– (sFr. 45,10/öS 382,20)

Erhältlich bei Ihrem Buchhändler.

Sie erhalten jedes **WordStar-Programm** für Ihren Commodore 128 fertig angepaßt (Bildschirmsteuerung). **Jeweils Originalprodukte!** Jedes Programmpaket enthält außerdem ein ausführliches Handbuch mit kompakter Befehlsübersicht.

Diese Markt&Technik-Softwareprodukte erhalten Sie in den Computer-Abteilungen der Kaufhäuser oder bei Ihrem Computerhändler.

610321

Markt&Technik
UNTERNEHMENSBEREICH
BUCHVERLAG
Hans-Pinsel-Straße 2, 8013 Haar bei München

Spitzen-Software für Commodore 128/128 D

dBASE II, Version 2.41

dBASE II, das meistverkaufte Programm unter den Datenbanksystemen, eröffnet Ihnen optimale Möglichkeiten der Daten- u. Dateihandhabung. Einfach u. schnell können Datenstrukturen definiert, benutzt und geändert werden. Der Datenzugriff erfolgt sequentiell oder nach frei wählbaren Kriterien, die integrierte Kommandosprache ermöglicht den Aufbau kompletter Anwendungen wie Finanzbuchhaltung, Lagerverwaltung, Betriebsabrechnung usw.

dBASE II für den Commodore 128 PC
Bestell-Nr. MS 303 (5¼"-Diskette)

Hardware-Anforderungen: Commodore 128 PC, Diskettenlaufwerk, 80-Zeichen-Monitor, beliebiger Commodore-Drucker oder ein Drucker mit Centronics-Schnittstelle

Für nur DM 199,–* (sFr. 178,–/öS 1890,–*)

***inkl. MwSt. Unverbindliche Preisempfehlung**

Markt&Technik
128er-Software

dBASE™ II
ASHTON-TATE
für den Commodore 128 PC

5¼"-Diskette
im Floppy 1541-Format

Und dazu die weiterführende Literatur:

Zu einem Weltbestseller unter den Datenbanksystemen gehört auch ein klassisches Einführungs- und Nachschlagewerk! Dieses Buch von dem deutschen Erfolgsautor Dr. Peter Albrecht begleitet Sie mit nützlichen Hinweisen bei Ihrer täglichen Arbeit mit **dBASE II.** Schon nach Beherrschung weniger Befehle ist der Einsteiger in der Lage, Dateien zu erstellen, mit Informationen zu laden und auszuwerten.

Best.-Nr. MT 838
ISBN 3-89090-189-1
DM 49,– (sFr. 45,10/öS 382,20)

Erhältlich bei Ihrem Buchhändler.

Sie erhalten jedes **dBASE II-Programm** für Ihren Commodore 128 PC fertig angepaßt (Bildschirmsteuerung). **Jeweils Originalprodukte!** Jedes Programmpaket enthält außerdem ein ausführliches Handbuch mit kompakter Befehlsübersicht.

Diese Markt&Technik-Softwareprodukte erhalten Sie in den Computer-Abteilungen der Kaufhäuser oder bei Ihrem Computerhändler

610322

Markt&Technik
UNTERNEHMENSBEREICH
BUCHVERLAG
Hans-Pinsel-Straße 2, 8013 Haar bei München

Spitzen-Software für Commodore 128/128 D

MULTIPLAN, Version 1.06

Wenn Sie die zeitraubende manuelle Verwaltung tabellarischer Aufstellungen mit Bleistift, Radiergummi und Rechenmaschine satt haben, dann ist MULTIPLAN, das System zur Bearbeitung »elektronischer Datenblätter«, genau das richtige für Sie! Das benutzerfreundliche und leistungsfähige Tabellenkalkulationsprogramm kann bei allen Analyse- und Planungsberechnungen eingesetzt werden wie z. B. Budgetplanungen, Produktkalkulationen, Personalkosten usw. Spezielle Formatierungs-, Aufbereitungs- und Druckanweisungen ermöglichen außerdem optimal aufbereitete Präsentationsunterlagen!

MULTIPLAN für den Commodore 128 PC
Bestell-Nr. MS 203 (5¼"-Diskette)

Hardware-Anforderungen: Commodore 128 PC, Diskettenlaufwerk, 80-Zeichen-Monitor, beliebiger Commodore-Drucker oder ein Drucker mit Centronics-Schnittstelle

Für nur DM 199,–* (sFr. 178,–/öS 1890,–*)

***inkl. MwSt. Unverbindliche Preisempfehlung**

Markt&Technik
128er-Software

MICROSOFT
MULTIPLAN
für den
Commodore 128 PC

5¼"-Diskette
im Floppy 1541-Format

Und dazu die weiterführende Literatur:

Dank seiner Menütechnik ist **MULTIPLAN** sehr schnell erlernbar. Mit diesem Buch von Dr. Peter Albrecht werden Sie Ihre Tabellenkalulation ohne Probleme in den Griff bekommen. Als Nachschlagewerk leistet es auch dem Profi nützliche Dienste.

Best.-Nr. MT 836
ISBN 3-89090-187-5
DM 49,– (sFr. 45,10/öS 382,20)

Erhältlich bei Ihrem Buchhändler.

Sie erhalten jedes **MULTIPLAN-Programm** für Ihren Commodore 128 PC fertig angepaßt (Bildschirmsteuerung). **Jeweils Originalprodukte!** Jedes Programmpaket enthält außerdem ein ausführliches Handbuch mit kompakter Befehlsübersicht.

Diese Markt & Technik-Softwareprodukte erhalten Sie in den Computer-Abteilungen der Kaufhäuser oder bei Ihrem Computerhändler.

610323

Markt&Technik
UNTERNEHMENSBEREICH
BUCHVERLAG
Hans-Pinsel-Straße 2, 8013 Haar bei München

Bücher zum Commodore 128/128 D

H. Ponnath

Grafik-Programmierung C128

1986, 196 Seiten

inkl. Beispieldiskette

Ein mächtiges Werkzeug hat der Anwender von Computergrafik mit dem Basic 7.0 des Commodore 128 PC in den Händen! Was man damit alles anfangen kann, soll Ihnen dieses Buch zeigen: hochauflösende Grafik, Multicolorbilder, Sprites und Shapes werden anhand von vielen Beispielprogrammen besprochen. Die Videochips und ihre Möglichkeiten sind ebenso Thema wie einige nützliche Assemblerroutinen, die Speicherorganisation, der 80-Zeichen-Bildschirm und vieles andere mehr. Außerdem enthält das Buch eine Diskette mit allen Programmen.

Best.-Nr. MT 90202
ISBN 3-89090-202-2
DM 52,–/sFr. 47,80/öS 405,60

P. Rosenbeck

Das Commodore 128-Handbuch

1985, 383 Seiten

Dieses Buch sagt Ihnen alles, was Sie über Ihren C128 wissen müssen: die Hardware, die drei Betriebssystem-Modi und was die CP/M-Fähigkeit für Ihren Computer bedeutet. Aber Sie werden irgendwann Lust verspüren, tiefer in Ihren C128 einzusteigen. Auch dafür ist gesorgt: an einen Assemblerkurs, der Ihnen zugleich die Funktionsweise des eingebauten Monitors nahebringt, schließen sich Kapitel an, die mit Ihnen auf Entdeckungsreise ins Innere der Maschine gehen. Daß die Reise spannend wird, dafür sorgen die Beispiele, aus denen Sie viel über die Interna des Systems lernen können – bis hin zur Grafik-Programmierung.

Best.-Nr. MT 90195
ISBN 3-89090-195-6
DM 52,–/sFr. 47,80/öS 405,60

J. Hückstädt

BASIC 7.0 auf dem Commodore 128

1985, 239 Seiten

Das neue BASIC 7.0 des C128 eröffnet mit seinen ca. 150 Befehlen ganz neue Dimensionen der BASIC-Programmierung. Es ermöglicht dem Anfänger den einfachen und effektiven Zugriff auf die erstaunlichen Grafik- und Tonmöglichkeiten des C128; der Fortgeschrittene findet die nötigen Informationen für (auch systemnahe) Profi-Programmierung mit strukturierten Sprachmitteln.

An praxisnahen Beispielen (wie z.B. der Dateiverwaltung) zeigt der Autor auf, wie man die für den 128er typischen Merkmale und Eigenschaften (Sprites, Shapes, hochauflösende Grafik, Musikprogrammierung und Geräusche) optimal nutzt!.

Best.-Nr. MT 90149
ISBN 3-89090-170-0
DM 52,–/sFr. 47,80/öS405,60

610324

Markt & Technik-Fachbücher erhalten Sie bei Ihrem Buchhändler

Markt&Technik
UNTERNEHMENSBEREICH
BUCHVERLAG
Hans-Pinsel-Straße 2, 8013 Haar bei München

TOP-ASS

Der ASE-Macro-Assembler für den Commodore 128 PC mit integriertem Editor, Monitor und Linker.

Dieser 6502-Macroassembler setzt neue Maßstäbe. Seine Leistungsfähigkeit wird jeden verwöhnten Maschinenprogrammierer überzeugen:

- integrierter Editor, der schon bei der Eingabe des Quelltextes eine Syntaxüberprüfung vornimmt;
- integrierter Linker, mit dem quellgesteuertes Linken von relokatiblen Modulen möglich ist;
- assemblereigene schnelle und gleichzeitig sehr leistungsfähige Integerarithmetik;
- über 2000 Labels können gleichzeitig verwaltet werden, das heißt Maschinenprogramme bis zu einer Länge von ca. 25 KByte Objektcode können bei Bedarf in einem Rutsch assembliert werden;
- Macros mit beliebig vielen Parametern, Macrobibliotheken, Minimacs, bedingte Assemblierung, Labeleingabe im Dialog, Ausgabe formatierter Assemblerlistings, Ausgabe sortierter Symboltabellen und vieles andere mehr.

Außerdem wird der ASE-Macroassembler von einem sehr guten Monitor und einem Relativlader unterstützt, der relokatible Module an beliebige Speicheradressen laden kann und endlich Schluß macht mit den Dutzenden Maschinenprogrammen auf Diskette, die sich nur durch ihre Startadresse unterscheiden!

Lernen Sie es kennen, das TOP-ASS Assembler-Entwicklungssystem! Es lohnt sich!

Best.-Nr. MD 253A **DM 89,–**
inkl. MwSt. Unverbindliche Preisempfehlung.

PROTEXT

Die Profi-Textverarbeitung mit vollautomatischer Silbentrennung, integrierter Tabellenkalkulation und Zusatzprogramm zum Überprüfen der Rechtschreibung für den Commodore128 PC.

PROTEXT ist ein leicht bedienbares Textprogramm mit hoher Leistungsfähigkeit. Eingebaute Hilfefunktionen ermöglichen eine schnelle Einarbeitung. Mit **PROTEXT** sind daher auch Anfänger in der Lage, alle Vorteile eines professionellen Textprogramms zu nutzen. Überzeugen Sie sich selbst!

Was **PROTEXT** alles kann:

- Farbkombination für Hintergrund und Schrift (Vordergrund) frei wählbar;
- formatierte Ausgabe auf Bildschirm und Drucker mit programmierbaren Haltepunkten über serielle, V24- oder zwei Software-Centronics-Schnittstellen;
- vielfältige Formatanweisungen: linker/rechter Rand, vollautomatische Silbentrennung, Kopf-/Fußzeilen, Fußnoten, Zentrieren usw.;
- schnelle selbstlernende Textkorrektur mit deutschem (ca. 25000 Worte) Grundwortschatz sowie neun Kundenbibliotheken, die in Text umgewandelt, bearbeitet, ergänzt, sortiert und ausdruckbar sind;
- Textübertragung per DFÜ mit Space-Optimierung und automatischer Fehlerkorrektur;
- leistungsfähige Rechenmöglichkeiten mit Zeilenmarkierung (Rechentabulator), Kolonnenverarbeitung, progr. Tabellenkalkulation und Taschenrechner.

Hardwareanforderungen: C128 oder C128D, 80-Zeichen-Monitor, Commodore-Drucker oder Drucker mit Centronics-Schnittstelle.

Best.-Nr. MD 254A **DM 89,–** inkl. MwSt. Unverbindliche Preisempfehlung.

610325

TOP-ASS und PROTEXT erhalten Sie in den Computer-Abteilungen der Kaufhäuser und in Computershops.

Markt&Technik
UNTERNEHMENSBEREICH
BUCHVERLAG
Hans-Pinsel-Straße 2, 8013 Haar bei München

Prof. Dr. Wolf-Jürgen Becker

C 128
Alles über CP/M 3.0

Der Autor:

Prof. Dr. rer. nat. WOLF-JÜRGEN BECKER, Jahrgang 1940, Studium an der Technischen Hochschule in Darmstadt, Promotion auf dem Gebiet der experimentellen Festkörperphysik. Seit 1970 mit der industriellen Betriebsmeßtechnik und deren Entwicklung und Konstruktion bei verschiedenen Meßgeräteherstellern beschäftigt. Seit 1968 Leiter des Fachgebietes Meßtechnik im Fachbereich Elektrotechnik an der Gesamthochschule/Universität Kassel.

Commodore liefert zum C 128 das Profi-Betriebssystem CP/M 3.0 bzw. CP/M Plus auf Diskette mit. Dadurch hat der Benutzer die Möglichkeit, seinen Commodore 128 als professionellen Personal Computer zu nutzen, weil er auf eine der umfangreichsten Softwarebibliotheken der Welt zugreifen kann. Von Standardsoftware bis hin zu speziellen Branchenlösungen wird diese jedem Anwendungswunsch gerecht.
In diesem Handbuch findet der Anwender alle Informationen, die er für den erfolgreichen Einstieg in das Betriebssystem CP/M 3.0 benötigt. Die einzelnen Befehle werden im logischen Zusammenhang mit den wesentlichen Optionen ausgeführt. Anhand von Beispielen werden die Funktionen erläutert. Ein ausführlicher Nachschlageteil enthält systemspezifische Informationen für den Programmierer.
Aus dem Inhalt:

- Einführung in CP/M 3.0
- CP/M-Befehle und deren Funktion
- Struktur von CP/M auf dem C 128
- CP/M-Disketten im C-128-Format
- Arbeiten mit CP/M anhand ausgesuchter Softwarebeispiele (Nevada Fortran und Turbo-Pascal)

Hard- und Softwareanforderungen:
Commodore 128 oder Commodore 128 D mit CP/M-3.0-Betriebssystem.

ISB N 3-89090-370-3

DM 52,-
sFr. 47,80
öS 405,60