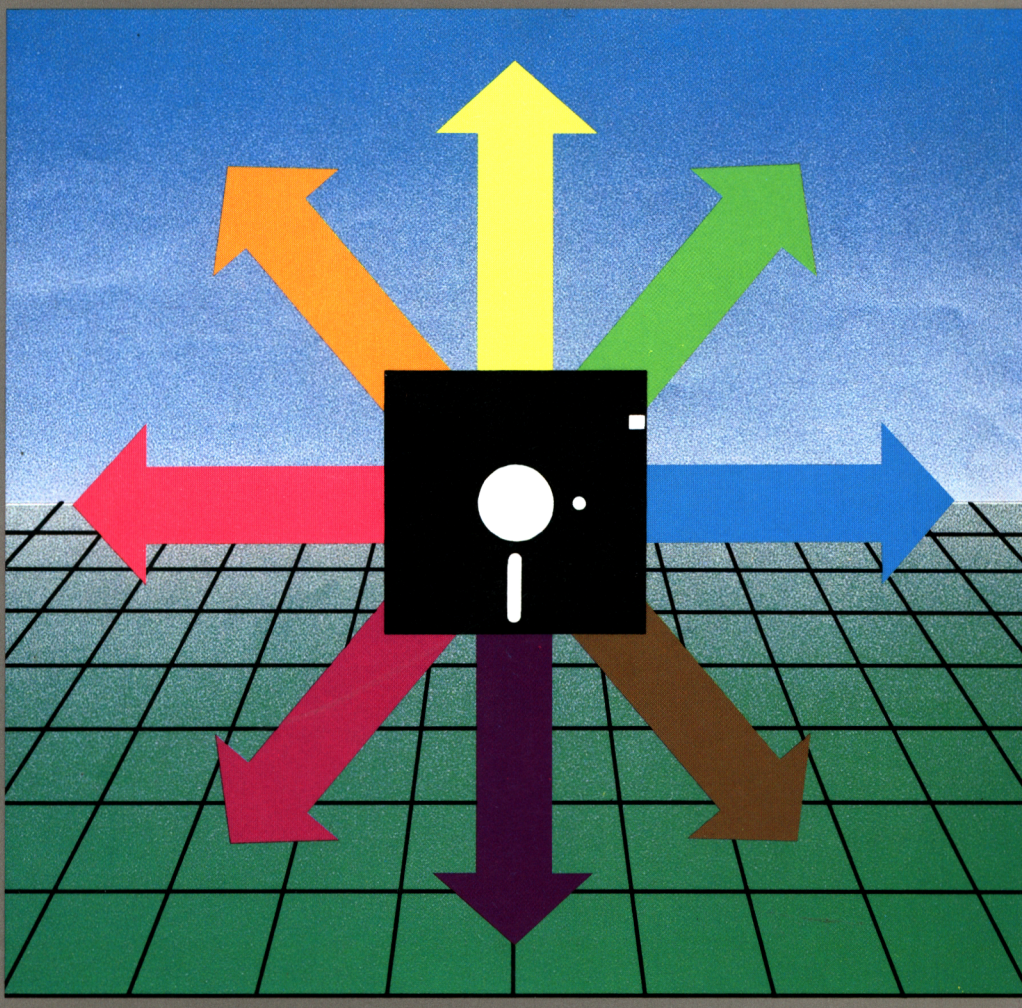
Thom Hogan

CP/M®

Anwenderhandbuch



OSBORNE/McGraw-Hill

Peter Steckel
Schlierbacherweg 18
6145 Lindenfels

Thom Hogan · Osborne CP/M Anwenderhandbuch

Thom Hogan

CP/M®
Anwenderhandbuch

McGraw-Hill Book Company GmbH
Hamburg · New York · St. Louis · San Francisco · Auckland · Bogotá · Guatemala
Johannesburg · Lissabon · London · Madrid · Mexiko · Montreal · New Delhi
Panama · Paris · San Juan · São Paulo · Singapur · Tokio · Toronto

Titel der Originalausgabe:
Osborne CP/M User Guide, Second Edition

Thom Hogan:
CP/M® Anwenderhandbuch
Hamburg: McGraw-Hill Book Company GmbH, 1984
ISBN 3-89028-005-6

Umschlaggestaltung: Conny Zug, Birkenau
Übersetzung: Hans-Herm. Wandke
Satz: Libro Satz J. Witt KG, Frankfurt
Druck und Bindung: Druckerei Bitsch GmbH, Birkenau

Dieses Buch ist
Lore Harp, Carole Ely, Steve Jobs, Steven Wozniak,
Gary Kildall und Seymour Rubenstein gewidmet,
die mir das Rüstzeug gaben,
es zu schreiben.

Folgende Bezeichnungen sind Warenzeichen oder eingetragene Warenzeichen® der nachstehenden Firmen:

Apple® Apple Computer, Inc.

CBM, PET® Commodore Business Machines Inc.

CP/M®, CP/NET®, CP/M-80, CP/M-86, MP/M II Digital Research Corp., Inc.

IBM® IBM

Smartmodem D.C. Hayes Corp.

Softcard Microsoft Inc.

The Source (Servicemark) Source Telecomputing Corporation

WordStar MicroPro International Corporation

Z80® Zilog, Inc.

Inhaltsverzeichnis

Einführung

Ihr Computer ist nicht eine Einheit, sondern ein aufeinander bezogenes System von Geräten und Programmen. Sie müssen diese Komponenten dirigieren, um ein gewünschtes Programm auszuführen. CP/M-80 und CP/M-86 sind Betriebssysteme, die einen großen Teil dieser Aufgabenstellung für Sie erfüllen. CP/M-80 oder CP/M-86 dirigiert die Aktivitäten Ihrer Computer-Komponenten und verwaltet Dateien, die Computerinstruktionen oder Daten enthalten.

Obwohl CP/M-80 wie CP/M-86 komplexe Programme sind, kann man sie ohne vorherige Computererfahrung erlernen. Dieses Buch führt den Anwender-Neuling in das Mikrocomputersystem ein und erläutert darin die CP/M-Funktion.

Kapitel 1 gibt die grundlegenden praktischen Informationen für den Beginn. Kapitel 2 und 3 beschreiben detailliert die CP/M-80- und CP/M-86-Kommandos. Diese Informationen werden Sie jeden Tag brauchen, und wir empfehlen, die Beispiele sorgfältig zu studieren. Diese drei Kapitel geben ein solides Fundament, um zu verstehen, was CP/M ist und wie man es gebraucht.

Kapitel 6 erklärt die Funktionen zweier Verwandter des CP/M – MP/M und CP/NET – und erläutert die Kommandos, die diesen Betriebssystemen gemeinsam sind. Die hauptsächlichen Unterschiede zwischen dem CDOS von Cromemco und dem CP/M-80 von Digital Research werden besonders hervorgehoben, um Cromemco-Operatoren einen leichten Zugang zu diesem Buch zu ermöglichen.

Kapitel 4 und 7 sind für Assembler-Programmierer geschrieben, die CP/M-80 oder CP/M-86 für die eigene Programmentwicklung erweitern wollen. Diese Informationen werden die meisten CP/M-Anwender nicht benötigen, aber sie wurden für eine vollständigere Betrachtung des Systems beigefügt. Wir hoffen, daß die couragierten Leser damit angeregt werden, vollen Gebrauch von den in diesen Kapiteln beschriebenen Dienstprogrammen zu machen.

Kapitel 8 schließlich arbeitet die persönlichen Erfahrungen des Autors mit CP/M-80 heraus und bietet eine Menge hilfreicher Tips.

Eine kommentierte Bibliographie gibt Hinweise auf zusätzliche Literatur, und verschiedene Anhänge bieten praktische Information über CP/M-kompatible Programme, Sprachen und Produkte.

Einige spezielle einführende Bemerkungen, die diese revidierte Ausgabe des Osborne CP/M® Anwenderhandbuches betreffen, sind beigefügt.

Das Hauptanliegen der Neufassung, der Restrukturierung und der Erweiterung des Textes dieser zweiten Ausgabe war es, das Buch akkurater und

vollständiger zu machen. Seit der Erstausgabe wurden CP/M-86, MP/M-86 und MP/M II eingeführt und eine dritte Neubearbeitung von CP/M-80 (CP/M3.0) entworfen. Und wie es bei vielen Werken geschieht, kamen dem Autoren weitere Gedanken, manche Teile des Buches verständlicher zu gestalten. Die CP/M-86-Kommandos und -Informationen wurden in einer Weise in den Text integriert, die es den Benutzern beider Systeme erlaubt, in gleicher Weise Gebrauch von dem Buch zu machen.

Ich habe mich bemüht, aus dieser Revision das vollständigste und genaueste Handbuch für CP/M-Anwender zu machen. Sollten Sie ein Computer-System zur Verfügung haben, lesen Sie dies Buch, während Sie davorsitzen! Probieren Sie Kommandos und Beispiele aus; begnügen Sie sich nicht mit dem Lesen! Sie werden schneller mit CP/M vertraut sein, als Sie erwarteten.

Ein letzter Hinweis: die Ausdrücke „CP/M", „CP/M-80" und „CP/M-86" bezeichnen spezifische (und unterschiedliche) Versionen des Betriebssystems. Wenn wir uns auf CP/M beziehen, meinen wir alle Versionen des CP/M. Wenn wir uns auf CP/M-80 oder CP/M-86 beziehen, meinen wir spezifische Versionen des Betriebssystems.

Bücher sind wie Computer-Programme selten ganz frei von Fehlern. Autor wie Herausgeber bitten Sie um Ihre Kommentare und Kritik.

Dieses Buch ist das Werk des Autors und des Herausgebers; es wurde von Digital Research weder revidiert noch autorisiert noch unterzeichnet.

Kapitel 1
CP/M und Betriebssysteme

CP/M ist ein Platten-Betriebssystem für Mikrocomputer von der Firma Digital Research. Der Name steht für „Control Program/Monitor". Versionen dieses Systems sind für ein breites Spektrum von Mikrocomputern aus einer Anzahl unterschiedlicher Quellen erhältlich. CP/M-80 läßt sich so ziemlich bei allen Mikrocomputern mit 8080- oder Z80-CPU und 8- oder 5¼-Zoll-Disketten-Laufwerken anwenden, CP/M-86 bei fast allen Mikrocomputern mit 8086- oder 8088-CPU und Floppy-Disk.

Geschichte des CP/M

CP/M wurde 1973 von Dr. Gary Kildall entwickelt, der zu dieser Zeit Software-Berater für Intel war. Die früheste Version schrieb er für sein eigenes Experimentalsystem, das eines der ersten Plattenlaufwerke, gebaut von Shugart Associates, einschloß – ein gebrauchtes Laufwerk, das für Gerätetests benutzt worden war, bevor man es Kildall überließ.

Kildall bot diese früheste Version Intel an, aber diese Firma lehnte es ab, das Projekt zu vermarkten oder weiterzuentwickeln. Das war nicht überraschend, denn 1973 und 1974 waren Mikrocomputer eine Seltenheit, und kaum ein Besitzer wußte genau, was er damit anfangen wollte.

Ab 1975 verkauften einige wenige kleine Firmen Mikrocomputer an neugierige Hobbyisten. Als die meisten dieser Firmen Computer mit Platten-Laufwerken bauten, entwickelten sie im allgemeinen auch ihre eigenen Betriebssysteme. Wären diese Pioniere – z. B. Altair, Polymorphic und Processor Technology – in der Lage gewesen, ihre Produkte schnell an den Mann zu bringen, wäre CP/M wohl kaum das „quasi-Standard"-Betriebssystem geworden, das es heutzutage ist.

Indessen jedoch entschieden sich mehrere kleine Mikrocomputer-Hersteller dafür, sich kostspielige Forschung und Entwicklung zu ersparen; sie übernahmen Kildalls CP/M-Betriebssystem für ihre Produkte. Besonders bemerkenswert unter diesen kleineren Firmen waren Tarbell Electronics und Digital Microsystems. Sie waren auch unter den ersten, die funktionierende Plattensysteme lieferten. Da diese Firmen Zusatzkomponenten – solche, die von nahezu jedem System benutzt werden konnten – erstellten, brauchten die Eigner von Altairs, Vectors, Polys und anderen Systemen nicht auf die Hersteller ihrer Computer zu warten, um Laufwerke zu produzieren. Außerdem hatte IMSAI, ein anderer Mikrocomputer-Pionier, Plattensysteme ohne irgendwelche Software ausgeliefert, und nun versprachen sie in Kürze ein fertiges Betriebssystem. Dieses Betriebssystem war dann IMDOS, tatsächlich eine getarnte Version von CP/M.

Ein anderes wichtiges Element in der Geschichte des CP/M ist der Enthu-

siasmus seiner ersten Benutzer. Diese wahren Hobbyisten nahmen häufig in ihrem Streben nach neuem Wissen und neuer Erfahrung unüberwindliche Probleme in Angriff. Theoretisch konnte CP/M-80 jeden auf 8080 oder Z80 basierenden Mikrocomputer mit jedem Plattensystem verbinden; und es tauchte eine Gruppe von Hobbyisten mit „mix and match" (= mische und ordne)-Systemen auf, die Kildalls Produkte testeten. Diese Hobbyisten entwickelten eine Anzahl von Verbesserungen und bildeten, was noch wichtiger war, eine starke und sichtbare Gruppe von Anwendern.

Die Unterstützung einer solchen Anwendergruppe sollte nicht unterschätzt werden. Während der Kinderjahre der Mikrocomputer-Industrie war genaue Produkt-Information nicht fertig zu erhalten. Hersteller gaben Produkte oft mit unvollständiger Information heraus, Computerläden waren noch relativ unbekannt, und in einigen Fällen waren die Anwendergruppen stabiler als die Gesellschaften, die die von den Anwendergruppen erwünschten Produkte entwickelten.

Nachdem die Hersteller begannen, zuverlässige Platten-Laufwerke zu liefern, leiteten Software-Entwickler die nächste entscheidende Phase in der Evolution des CP/M ein. Der Schlüssel, Software-Entwicklung finanziell durchführbar zu machen, ist, Programme zu schreiben, die auf so vielen verschiedenen Mikrocomputern wie möglich laufen können. CP/M-80 war eines der wenigen Betriebssysteme, die auf so ziemlich jedem auf 8080 oder Z80 basierenden Mikrocomputer laufen konnten, und es war nicht an einen einzigen Typ von Platten-Laufwerk gebunden.

Zum Glück für die CP/M-Entwicklung waren die ersten verfügbaren Programme Entwicklungswerkzeuge – Programme zur Generierung anderer Programme. Unter den verschiedenen Entwicklungswerkzeugen, die halfen, CP/M als führendes Betriebssystem für Mikrocomputer zu etablieren, waren CBASIC (und sein Vorgänger EBASIC), Microsoft BASIC und verschiedene spezielle Programme für die Assemblersprache. Diese alle wurden dann benutzt, um Anwenderprogramme wie z. B. generelle Fixpunkt-Routinen, Datenbanken und zentrale Bestandsführungen oder Textverarbeitung zu schreiben.

Die Popularität des CP/M-Betriebssystems wurde Teil eines eskalierenden Musters: CP/M erzeugte Programmiersprachen und Entwicklungswerkzeuge, die dann wieder die Entwicklung von Anwenderprogrammen förderten. Diese CP/M-abhängigen Anwenderprogramme steigerten den CP/M-Umsatz, was dann wieder zu einer höheren Anzahl eingeführter Entwicklungswerkzeuge führte. Diese Aufwärtsspirale des Umsatzes: dadurch mehr Werkzeuge – dadurch mehr Anwenderprogramme – dadurch mehr Umsatz – hat sich jetzt über mehrere Jahre ungehindert fortgesetzt und zeigt keine Anzeichen aufzuhören.

In der Tat war 1981 ein wichtiges Jahr für CP/M. Mit der Einführung neuer Mikrocomputer mit CP/M-Betriebssystem als Standard oder Option durch

Computerriesen wie IBM, Hewlett-Packard und Xerox stieg die Benutzerzahl des CP/M während des Jahres über die Viertelmillionen-Grenze. Es wird geschätzt, daß heute über 300 Computer-Hersteller zusammen mit ihrer Geräteausstattung CP/M-80 oder CP/M-86 anbieten; Besitzer von Mikrocomputern, die keine 8080-, 8086-, 8088- oder Z80-Zentraleinheit haben, können mit der jüngsten Einführung der Microsoft SoftCard für den Apple und der Ankündigung einer zusätzlichen Prozeßplatine für die Commodore CBM- und PET-Computerserie CP/M benutzen.

Nach den bescheidenen Anfängen ist CP/M das am häufigsten benutzte Betriebssystem für Mikrocomputer geworden (und wahrscheinlich für alle Computer, wenn die Anzahl der Installationen, nicht die der Benutzer gezählt wird). Kildalls Original-Betriebssystem hat viele Abwandlungen durchgemacht, aber trotz seiner Einfachheit gibt es über CP/M viel zu lernen.

CP/M-Handbücher

Nun treten Sie auf den Plan. Wahrscheinlich haben Sie dieses Buch gekauft, weil Sie CP/M benötigen, um ein Anwenderprogramm zu fahren. Es mag sich um ein einfaches Textverarbeitungsprogramm oder ein hochentwickeltes Buchungssystem handeln; beide erfordern, daß CP/M verstanden wird.

Digital Researchs CP/M-Handbücher sind nicht für Sie, sondern für professionelle Programmierer geschrieben. Dieses Buch versucht, die Kluft zwischen den Handbüchern von Digital Research und Ihrem Wissen über Computer zu überbrücken.

Die CP/M-Handbücher, die Sie besitzen, werden von Ihrer Version des CP/M abhängen und woher Sie sie bekommen haben. Digital Research verbessert CP/M ständig, und so werden von Zeit zu Zeit neue Versionen herausgebracht. Das jüngst herausgegebene CP/M-80 ist die Version 2.2, davor wurde gewöhnlich die Version 1.4 angeboten. Außerdem vertreiben IBM und andere Firmen jetzt eine spezielle Version des CP/M namens CP/M-86 (weil sie für die 8086- und nicht für die 8080- oder Z80-Zentraleinheit entworfen wurde).

Wenn Sie die Version 2.0 des CP/M-80 oder eine neuere Version (höhere Nummer) gekauft haben, sollten Sie die folgenden Handbücher besitzen:

- An Introduction to CP/M Features und Facilities
- CP/M 2.0 User's Guide
- ED: A Context Editor for the CP/M Disk System
- CP/M Assembler (ASM)
- CP/M Dynamic Debugging Tool (DDT)
- CP/M 2.0 Alteration Guide
- CP/M 2.0 Interface Guide

Benutzer der CP/M-80 Version 1.4 oder einer früheren werden den CP/M 2.0 User's Guide, den CP/M 2.0 Alteration Guide und den CP/M 2.0 Interface Guide nicht besitzen und stattdessen einen Alteration Guide und Interface Guide für die Version 1.4 haben.

Einige Computerhersteller wie Microsoft, CompuPro und Morrow Designs legen jetzt diese sieben Handbücher neu auf und binden sie zu einem. Andere, wie Osborne Computer Corporation und Xerox, benutzen keine Digital Research-Handbücher, sondern liefern solche speziell für ihre eigenen Computer.

CP/M-86-Besitzer erhalten vier Handbücher:

- CP/M-86 Operating System User's Guide
- CP/M-86 Operating System Programmer's Guide
- CP/M-86 Operating System Guide
- CP/M-86 Operating System Command Summary

Käufer von IBM Personal Computern erhalten einen besonders für sie geschriebenen Satz von Handbüchern, der den vorher erwähnten Büchern entspricht.

Wie auch immer, Sie finden vermutlich die Handbücher, die Sie zusammen mit dem CP/M erhalten haben, zu unklar oder verwirrend, daher haben Sie dieses Buch gekauft, um die Informationen zu finden, die Sie für die Benutzung von CP/M brauchen. Dieses Buch erklärt den Inhalt der ersten fünf CP/M-80-2.2-Handbücher sowie des CP/M-86 System Guide im Detail und es faßt die Information der übrigen Handbücher zusammen.

Die Funktionen des CP/M innerhalb eines Mikrocomputer-Systems

Bevor wir weitergehen, ist es wichtig, die von CP/M innerhalb eines Mikrocomputer-Systems unterstützten Funktionen zu verstehen. Wenn Sie wissen, was vorgeht und warum, werden Sie wahrscheinlich weniger Fehler machen.

Wir wollen deshalb die Funktionen des CP/M (oder irgendeines Betriebssystems) im Gebrauch eines Computers beschreiben. Diese Beschreibung setzt ein elementares Verständnis von Mikrocomputern und ihrer Funktionsweise voraus.*

In Abb. 1–1 ist ein Mikrocomputer in einer typischen Konfiguration illustriert. Das System schließt die Zentraleinheit, ein Terminal mit Tastatur und Bildschirm, zwei Plattenlaufwerke und einen Drucker ein.

Dieses System ließe sich in vielfältiger Weise abwandeln. Anstelle eines integrierten Terminals lassen sich Bildschirm und Tastatur trennen. Die Tasta-

* Wenn sie mehr Information über Mikrocomputer-Systeme benötigen, vgl. „An Introduction to Microcomputers: Volume 0 – The Beginner's Book, 3rd. ed." von Adam Osborne und Dave Bunnel. Berkeley: Osborne/McGraw-Hill, 1982.

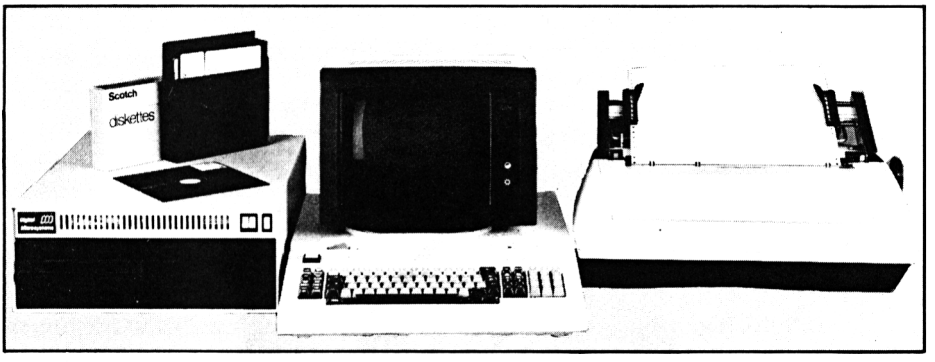


Abb. 1-1. Ein komplettes Mikrocomputer-System

tur könnte ein Teil des Mikrocomputers sein und der Bildschirm eine separate Einheit; oder Tastatur, Bildschirm und Mikrocomputer könnten als eine Einheit integriert sein.

Kleine Systeme können Kassettenbänder und -geräte anstelle von Floppy-Disketten und -Laufwerken verwenden. In den Anfangszeiten benutzten Mikrocomputer-Systeme Lochstreifen, um Informationen zu speichern und erneut zu lesen, und sie benötigten dafür einen Lochstreifenleser und einen Lochstreifenstanzer. Der Gebrauch von Kassetten und Lochstreifen zusammen mit dem CP/M ist nicht mehr üblich, da Platten schneller und zuverlässiger sind.

Mikrocomputer brauchen viel Zeit für die Informationsübertragung zwischen dem Mikroprozessor und verschiedenen anderen Komponenten des Systems. Auch müssen sie die Operationen dieser anderen Komponenten kontrollieren. Dies geschieht durch die Ausführung von Programmen, die kollektiv als ein Betriebssystem bezeichnet werden. CP/M ist solch ein Betriebssystem. Mit geeigneten CP/M-Kommandos kann man Daten von einer Diskette zum Mikrocomputer übertragen, über Drucker ausgeben oder jede Operation ausführen, zu der das Mikrocomputer-System in der Lage ist.

Um diese Funktionen eines Mikrocomputer-Systems für eine breite Steuerung unterschiedlicher Konfigurationen durchführen zu können, ignoriert CP/M (und die meisten anderen Betriebssysteme) die einzelnen Teile, die das Mikrocomputer-System umfaßt, und bezieht sich stattdessen auf logische Einheiten. Mit anderen Worten, statt beispielsweise direkt einen Drucker zu adressieren, nimmt das Betriebssystem an, daß eine List-Einheit vorhanden ist. Gleicherweise nimmt es, statt direkt von einem Lochstreifenleser einzulesen, an, daß die Eingabe von einem Leser kommt.

Der Hersteller Ihres Mikrocomputer-Systems garantiert üblicherweise, daß die wirklichen Geräteeinheiten sauber mit den logischen Einheiten verbunden

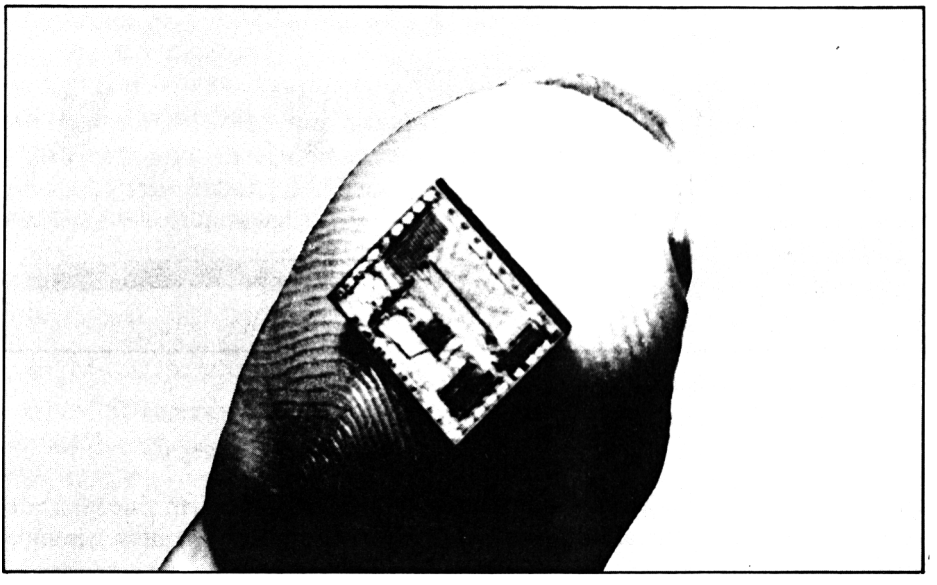


Abb. 1-2. Ein Mikroprozessor-Chip

sind, die CP/M benutzt. Wenn sie jedoch ein System durch „mixing and matching" (= mischen und anpassen) von Komponenten mehrerer Hersteller selbst zusammenstellen, müssen Sie selbst einige Veränderungen des CP/M vornehmen. Diese Programm-Modifikationen sind unsichtbar, sofern sie korrekt durchgeführt werden; wenn sie nicht gemacht werden, wird CP/M möglicherweise unkorrekt arbeiten, wenn überhaupt.

Als Operator eines Mikrocomputer-Systems werden Sie gelegentlich mit Geräten und mit logischen Einheiten zu tun haben. Sie werden beispielsweise die Möglichkeit haben, über Drucker oder Bildschirm auszugeben. Gleicherweise könnte eine Eingabe entweder über Tastatur oder über Telefonleitung von einem ausgelagerten Terminal erfolgen. Solche Auswahlen können leicht durch die dafür vorgesehenen CP/M-Kommandos vorgenommen werden, die wir später in diesem Buch beschreiben. Im Moment geht es lediglich um das Verständnis der generellen Funktion des CP/M. Man braucht nicht die spezifischen Aktivitäten des Betriebssystems zu verstehen, um CP/M zu benutzen.

Ein Betriebssystem wie CP/M ist selbst ein Programm, das von einem Mikrocomputer ausgeführt werden muß. Deshalb muß es in einer Programmiersprache geschrieben sein. Die Programmiersprache, die ein Mikrocomputer versteht, wird durch den in ihm enthaltenen Mikroprozessor (manchmal CPU (Central Processing Unit = zentrale Recheneinheit) genannt) bestimmt.

Ein Mikroprozessor (s. Abb. 1–2) ist eine sehr kleine und unscheinbare Einheit. Er ist die wichtigste Komponente eines Mikrocomputers; er übersetzt

nämlich die Instruktionen, die ein Programm ausmachen und bewirkt die damit verbundenen Aktionen. Einige Mikroprozessoren können CP/M ausführen, andere nicht.

CP/M wurde anfänglich für den von Intel hergestellten Mikroprozessor 8080A geschrieben. Da die Mikroprozessoren 8085 und Z80 ebenfalls 8080A-Instruktionen ausführen, läuft das CP/M-80 auf Mikrocomputern, die einen dieser beiden Mikroprozessoren enthalten. Zusätzlich führte Digital Research Anfang 1981 CP/M-86 für die Mikroprozessoren 8086 und 8088 ein.

Um zwischen den 8080- und 8086-Versionen von CP/M zu unterscheiden, benutzen wir dieselbe Nomenklatur wie Digital Research: CP/M-80 bezieht sich auf die laufenden Versionen der Mikroprozessoren 8080, 8085 und Z80; CP/M-86 auf die der Mikroprozessoren 8086 und 8088.

Typen des CP/M

CP/M kann z. Zt. fast als Standard-Betriebssystem verstanden werden, aber nicht alle CP/Ms sind gleich. CP/M-80 und CP/M-86 unterscheiden sich in den I/O-Instruktionen (Input/Output = Ein-/Ausgabe) für jede Maschine und auch anderweitig.

Außerdem steht die Computer-Technologie nicht still. Die Maschine, die Sie jetzt besitzen, tut viel mehr viel schneller als die Vakuumröhren-Computer aus den 50er Jahren. Die jüngste Einführung von Hart- und Festplattenlaufwerken mag eine der einschneidendsten Entwicklungen werden. Diese neuen Platteneinheiten können viel mehr Information speichern als die geläufige Disketten-Technologie. Während Floppy-Disketten bis zu 2 MB (2 Megabytes = 2 048 000 Zeichen) speichern können, können Hart- und Festplatten bis zu 300 MB speichern.

Da CP/M-80 ursprünglich für Floppy-Disketten-Systeme entworfen wurde, verlangt die Benutzung von Hartplatten-Einheiten Änderungen. Dies ist der hauptsächliche Unterschied zwischen den Versionen 1.4 und 2.2 des CP/M-80 (s. Anhang C – eine komplette Liste der Unterschiede zwischen den Versionen).

Die verschiedenen Typen von CP/M-80 und CP/M-86 spiegeln auch die große Anzahl von Herstellern wider, die CP/M benutzen. Jedermann kann dem CP/M Dienstprogramme oder Verfeinerungen hinzufügen, um die Leistung einer bestimmten Maschine zu erhöhen.

CP/M-Kompatibilität

Leider sind nicht alle CP/Ms kompatibel, und selbst der Grad der Kompatibilität kann unterschiedlich sein. CP/Ms unterscheiden sich hauptsächlich in

- Versionsnummer (1.3, 1.4, 2.0, 2.1, 2.2)
- CP/M-Kernspeicherbelegung (64K CP/M, 48K CP/M)

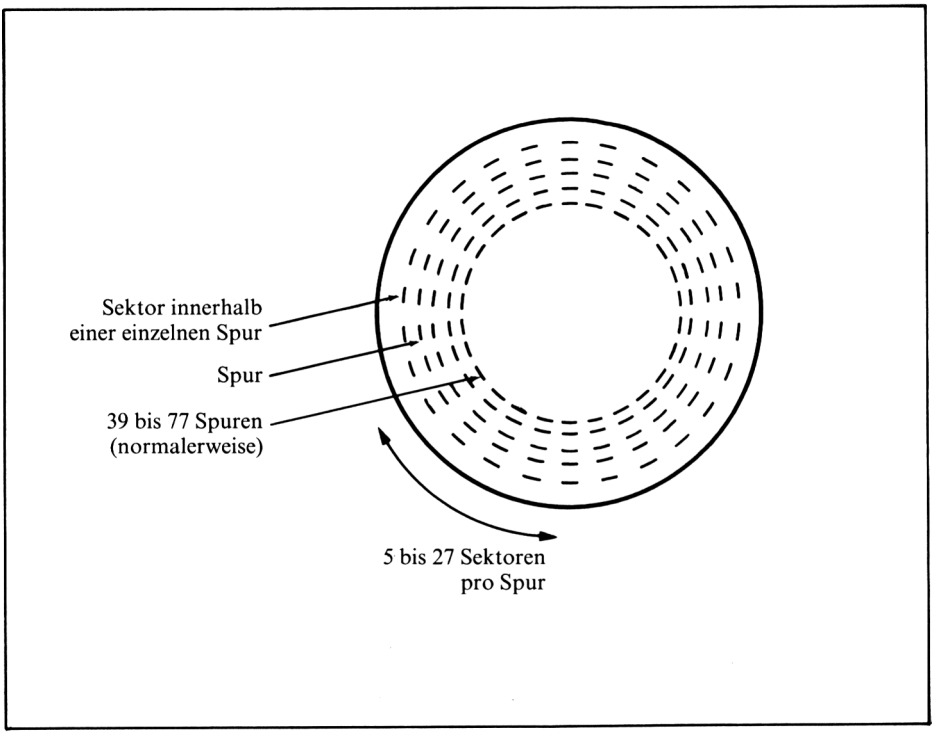


Abb. 1–3. Aufbau einer Disketten-Oberfläche

- Diskettentyp (einfache Dichte, doppelte Dichte, 5¼-Zoll, 8-Zoll)
- dem logischen Layout auf der Diskette (die Art, wie Information auf der Diskette gespeichert wird, unterscheidet sich häufig in der Anzahl der benutzten Sektoren und/oder Spuren)
- Hersteller (normalerweise eine Kombination der vier vorherigen Kriterien)
- Prozessortyp (CP/M-80 gegen CP/M-86).

Es wäre zwecklos, jeden einzelnen CP/M-Typ auflisten zu wollen; die Liste wäre überholt, wenn Sie dieses Buch lesen. Aber trotz der Variationsmöglichkeiten bleibt CP/M doch maschinenunabhängiger als die meisten Betriebssysteme. Durch Verbindung von Computern mit zwei verschiedenen Versionen des CP/M kann normalerweise ein Programm- oder Datenaustausch stattfinden.

CP/M-Versionsnummern

Wie bereits festgestellt gibt es mehrere Versionen des CP/M. Generell ist dieses Buch auf die Version 2.2 des CP/M-80 zugeschnitten, aber Besitzer

früherer oder nachfolgender Versionen werden dieses Buch ebenfalls als nützlich empfinden.

CP/M-80-Versionen sind durch eine Nummer links vom Dezimalpunkt, die sich auf die Gesamtversion bezieht, und eine Nummer rechts vom Dezimalpunkt, die sich auf eine Revision innerhalb der Version bezieht, gekennzeichnet.

Eine zweite Ziffer rechts vom Dezimalpunkt, wie z. B. in 1.42, identifiziert subtile oder maschinenabhängige Modifikationen. Wenn Ihre Version des CP/M-80 nicht 1.4x oder 2.2x ist (wobei „x" eine maschinenabhängige Nummer bedeutet), so bitten Sie Ihren Händler um ein neues Release (= Ausgabe). Z. Zt. existieren folgende Versionen des CP/M-80:

- 1.3 die Original-Version von CP/M-80
- 1.4 eine fehlerfreiere Version des CP/M-80 Release 1
- 2.0 die Original-Release-Version des CP/M-80 Release 2
- 2.1 eine teilweise Überarbeitung der Version 2.0
- 2.2 die aktuelle Revision des CP/M-80 Version 2.

Wenn Sie die Versionen 1.3 oder 2.0 von CP/M-80 benutzen, sollten Sie sich sofort bemühen, daß Ihre Version erneuert wird, denn sonst könnten schwerwiegende Probleme entstehen.

Versionsnummern können von Hersteller zu Hersteller variieren. Einige CP/M-80-Händler gaben mehrere unterschiedliche Revisionen als Version 1.4 heraus, während andere ihre eigenen Updates (= Auffrischungen) mit Nummern wie 1.41 und 1.42 versahen. Jedenfalls zeigen nur die ersten beiden Ziffern in einer Versionsnummer Änderungen des CP/M-80 durch Digital Research an.

Digital-Research – CP/M-Produkte

Da Digital Research CP/M geschrieben hat und es weiter ausbaut, wollen wir andere Produkte mit denen der Digital Research vergleichen. Z. Zt. werden folgende Produkte von Digital Research unterstützt:

8-Zoll-Disketten einfache Dichte (SD).

Diese Diskette benutzt ein vordefiniertes Format, um Informationen zu speichern. Ursprünglich war CP/M-80 für dieses Format konzipiert. Die Versionen 1.4 und 2.2. sind erhältlich und unterscheiden sich hauptsächlich in der maximalen Anzahl von Plattenlaufwerken und der Adressierungs-Kapazität. Unterstützt von Digital Research wird CP/M-80 demnächst auf dem Mikrocomputer eines Intel-MDS-Entwicklungssystems gefahren werden können; wenn Sie jedoch einen anderen Mikrocomputer besitzen, werden Sie Teile des CP/M-80 modifizieren müssen, damit es korrekt arbeitet.

8-Zoll-Disketten doppelte Dichte (DD).

Auf dieser Diskette kann mehr Information gespeichert werden als auf einer von einfacher Dichte.

Vorsicht: Die meisten Implementationen doppelter Dichte sind auf CP/M-80 nicht direkt kompatibel; eine Diskette, die auf dem Laufwerk eines Herstellers erstellt wurde, kann normalerweise nicht eingelegt und auf den Laufwerken anderer Hersteller gelesen werden. Wie bei einfacher Dichte basiert Digital Research's Implementation auf IBM-Spezifikationen; jedenfalls unterstützen, anders als bei einfacher Dichte, sehr wenige Mikrocomputer-Hersteller standardmäßig doppelte Dichte.

Darüberhinaus unterstützt Digital Research die folgenden Betriebssysteme:

MP/M.

Eine Gemeinschaftsbenutzer-Form des CP/M. Statt eines einzigen können bei MP/M mehrere Terminals angeschlossen sein (mehr Information s. Kap. 6). Die laufende Version von Digital Research ist als MP/M II bekannt.

CP/NET.

Eine Multi-Computer-Form von CP/M, die einem Computer erlaubt, die Peripherie eines anderen zu benutzen (Drucker, Plattenlaufwerke usw.). Um CP/NET zu benutzen, muß mindestens einer der Computer mit MP/M ausgestattet sein und alle anderen CP/M benutzen (auch hierzu mehr Details in Kap. 6).

CP/M-86.

Eine spezielle Implementation für Computer mit den Intel-Mikroprozessoren 8086 oder 8088. Außer einer CP/M-86-Version auf 8-Zoll-Disketten einfacher Dichte vertreibt Digital Research auch eine auf 5¼-Zoll-Disketten für das IBM Personal Computer-System.

Gleichlaufende CP/M-86-Systeme

Wie das CP/M-86 wurde auch ein gleichlaufendes CP/M-86 für 8088- oder 8086-CPUs entworfen. Der durch den Titel angezeigte „Gleichlauf" bezieht sich auf die Tatsache, daß auf einem gleichlaufenden CP/M-86 von einem einzelnen Benutzer Mehrfachprozesse durchgeführt werden können. Digital Research liefert gleichlaufende CP/M-86 für IBM-Bildschirmschreiber und Personal Computer-Systeme.

Achtung: Digital Research hat eine lauffähige CP/M-Implementation nur für spezifische Geräte-Konfigurationen parat. Bei Mikrocomputer-Systemen, die Digital Research nicht unterstützt, sind spezielle Instruktionen für Terminal, Modem, Drucker und Plattenlaufwerke notwendig. Nur ein erfahrener Assembler-Programmierer sollte CP/M direkt von Digital Research kaufen. Für die Installation von CP/M können Berater hinzugezogen werden, wenn eine fertige Implementation für Ihre Hardware nicht erhältlich ist.

Lifeboat Associates CP/M-80-Produkte

Zusätzlich zum Verkauf der CP/M-Produkte von Digital Research hat Lifeboat Associates den Teil des CP/M-80 geändert, der Instruktionen für die Plattenlaufwerke und andere Einheiten enthält (BIOS-Sektion s. Kap. 7).

Lifeboat Associates' CP/M-80-Produkte sind für den Radio Shack TRS-80, für Micropolis, North Star, Polymorphic, Altair, Zenith/Heathkit und für eine Anzahl von weiteren Systemen erhältlich.

Die CP/M-80-Software von Lifeboat Associates ist fast zu allen CP/M-80-Systemen kompatibel, nicht aber die Hardware. Deshalb kann das Endresultat enttäuschend für Sie sein. Da die physischen Datenträger, die Disketten, sich unterscheiden können, kann es schwierig sein, Programme oder Daten von einem Mikrocomputer für einen anderen zu kopieren. Auf der anderen Seite können Sie, wenn beide Systeme Modems haben, Programme über Telefonleitung an ein anderes System senden.

Die Wichtigkeit einer Ausstattung, die in der Lage ist, Informationen im Format einer IBM 3740 – 8-Zoll-Diskette einfacher Dichte zu lesen und zu schreiben, sollte nicht unterschätzt werden. Z. Zt. ist dies die einzige physische Möglichkeit, Information zwischen Computern auszutauschen. Auch wenn Sie keine 8-Zoll-Plattenlaufwerke einfacher Dichte besitzen, sollte Ihr Computer irgendwie in der Lage sein, mit anderen Ausstattungen zu kommunizieren. Normalerweise wird ein „modem port" (Modulator/Demodulator-Durchlaß) oder vielleicht ein extra „serial port" Sie befähigen, Ihren Computer mit einem anderen zu verbinden.

Da Lifeboat Associates nur ein Wiederverkäufer von CP/M-80 ist, werden Produkte dieser Firma erst nach Überarbeitung durch Digital Research herausgegeben. Z. B. gab Lifeboat Associates Produkte für Laufwerke mit anderen als den Standard-8-Zoll-Floppies erst ein Jahr nachdem Digital Research die Version 2.2 des CP/M-80 einführte, heraus.

Überhaupt benutzen nicht alle Disketten-Systeme generell alle Dienstprogramme des CP/M-80. So ist z. B. die Formatierung einer Diskette – der Prozeß, die Sektionen einer vorher unbenutzten Diskette so zu kennzeichnen oder zu identifizieren, daß das Betriebssystem erkennen kann, wo die Daten liegen – für das CP/M-80 einfacher Dichte von Lifeboat's North Star nicht vorgesehen.

CP/M-Produkte der Hersteller

Viele Hersteller von Mikrocomputern bieten CP/M-80 oder CP/M-86 als Standard-Betriebssystem für ihre Computer an. Dazu gehören Altos, CompuPro, Digital Microsystems, Dynabyte, Exidy, IMS International, Micromation, Morrow Designs, North Star, Onyx, Osborne und Vector Graphic.

Die CP/M-Produkte der Hersteller unterscheiden sich geringfügig von denen der Lifeboat Associates – alle Hersteller rekonfigurieren Teile des CP/M für ihre eigene Ausstattung. Einige jedenfalls fügen dabei subtile Fallen für den Unvorsichtigen hinzu. Diese Fallen nennen sich oft „features" (= Charakteristika). So enthält z. B. die Version 2.2 des Vector Graphic CP/M-80 einige komplizierte Prüfungen des benutzten Computer-Modells sowie eine Routine für die Prüfung bestimmter Tasten. Diese Zusatzroutinen machen

aber Programme, die kompatibel zum Standard-CP/M-80 sind, inkompatibel zur Implementierung von Vector Graphics.

Mindestens ein Hersteller hat auch die interne Struktur des CP/M-80 modifiziert, um es schneller zu machen. Das ist gewiß von Vorteil, reduziert jedoch die Kompatibilität zu Systemen, die auf einer Standard-Version von CP/M-80 basieren.

Dann haben andere Hersteller wie TEI Betriebssysteme für ihre eigenen Ausstattungen geschrieben, die ähnlich wie CP/M arbeiten. Auch hier mögen signifikante Vorzüge für die Benutzung solcher Systeme bestehen, aber sie sind nicht CP/M; und Benutzer dieser „look-alikes" sollten dies beachten.

Mit der Ankunft des CP/M-86 begannen einige Hersteller, insbesondere CompuPro und IBM, damit, dieses Betriebssystem für ihre eigenen Computer auszulegen. CP/M-86-Diskettendaten sind meistens kompatibel zu solchen des CP/M80, nicht aber Programme, jedenfalls nicht ohne gewisse Konversionen.

Anfang 1982 erschien auch eine andere Mischform des CP/M-Betriebssystems von Digital Research. CompuPro und G&G Engineering führten eine CP/M-MP/M-Version des Betriebssystems ein – eins, auf dem Programme beider Originalversionen (CP/M-80 oder MP/M II) laufen können. Noch einen Schritt weiter ging DEC (Digital Equipment Corporation) mit der Einführung ihres Systems Rainbow 100, das sowohl der Z80- wie auch 8080-CPU entspricht. Das DEC-CP/M kann Programme und Kommandos für CP/M-80 und CP/M-86 ausführen.

Wenn sie also ein nicht-standartisiertes CP/M-Betriebssystem erhalten, sollten Sie das begleitende Handbuch lesen, um zu sehen, wie es sich von den hier beschriebenen Versionen unterscheidet.

CP/M-Look – alikes

Außer dem Computer-Herstellern hat auch eine Anzahl von Software-Firmen CP/M-Look-alikes (= ähnlich aussehende Systeme) produziert. Davon können die meisten für ein weites Spektrum von Mikrocomputern benutzt werden. Einige sind allerdings eingeschränkter als CP/M-80 und erfordern einen Z 80 als Mikroprozessor. Das TP/M von Computer Design Labs ist solch ein Produkt.

Andere populäre CP/M-80-Look-alikes sind SDOS von SD Systems, ursprünglich für den SD-Computer entwickelt und dann separat vermarktet, I/OS (auch unter dem Namen TSA/OS angeboten) von InfoSoft und Turbo-Dos, ein spezielles Look-alike, das sich durch die Geschwindigkeit von Kernspeicher-Platte-Übertragungen auszeichnet. Auch CDOS, das Cromemco-Platte-Betriebssystem, nimmt Kompatibilität zur Version 1.3 des CP/M-80 für sich in Anspruch. Das MULTI/OS, ebenfalls von InfoSoft, ist ein MP/M-Look-alike.

In den meisten Fällen werden diese Look-alikes mit Standard-CP/M-80-Software fahren. Für diejenigen, die einen CP/M-80-Ersatz gekauft haben, ist Kapitel 6 über Look-alikes geschrieben.

Jetzt gibt es auch ein CP/M-86-„work-alike" (= ähnlich arbeitend), das je nachdem als 86-DOS, Microsoft DOS oder IBM-DOS bekannt ist. Ursprünglich entwickelt, um Anwendern des Systems von Seattle Computer Products zu ermöglichen, existierende CP/M-80-Programme schnell auf 8086-Peripherie zu konvertieren, ist das 86-DOS dem CP/M-86 ähnlich. Auch das 86-DOS wird in Kapitel 6 kurz angeschnitten.

Disketten

Das ganze Buch hindurch werden wir uns auf „Floppy-Disketten" und „Plattenlaufwerke" beziehen. Eine Floppy-Diskette ist der dünne bewegliche Datenträger für ein Plattenlaufwerk, das den mechanischen Anteil stellt. Plattenlaufwerke, die mit polierten Metall-„platters" anstelle von Floppy-Disketten arbeiten, werden in diesem Buch als „Hartplatten-Laufwerke" abgehandelt (denn der Datenträger ist hart). Zur Präzisierung wollen wir Floppy-Disketten als „Disketten", Plattenlaufwerke als „Laufwerke" und Hartplatten-Laufwerke als „Hartplatten" bezeichnen.

Das Rückgrat eines Platten-Betriebssystems ist natürlich die Platte selbst. Die Vorstellung mag schwierig sein, daß 150 eng beschriebene Schreibmaschinenseiten (das Ergebnis von 30 Stunden Tipparbeit) auf einer kleinen magnetischen Diskette gespeichert werden können. Leider kann man leicht vergessen, die notwendigen Schritte zu unternehmen, um die auf der Diskette gespeicherte Information zu sichern. Wir wollen deshalb unsere ausführlichen Informationen über CP/M für einen Schnellkursus über Typen, Pflege und Gebrauch von Disketten unterbrechen.

Disketten-Vergleich

Wenn Sie in einen Computerladen gehen um eine Diskette zu kaufen, wird der Verkäufer Sie fragen, welche Art Sie benötigen. Um eine Vorstellung der Möglichkeiten zu gewinnen, betrachten Sie die folgenden Merkmale:

- 8-Zoll- gegen 5¼-Zoll-Disketten
- einseitige gegen doppelseitige Disketten
- Disketten einfacher gegen Disketten doppelter Dichte
- Disketten mit weichen gegen Disketten mit harten Sektoren
- Disketten mit 10 gegen Disketten mit 16 harten Sektoren
- Schreibschutzsperre oder nicht

Das ist eine verwirrende Reihe von Auswahlmöglichkeiten, und es gibt so viele Handelszeichen wie Typen. Wir könnten nicht annähernd alle Kombinationen auflisten. Tatsächlich publizieren Diskettenhersteller lange Listen

von kompatiblen Disketten, Computern und Laufwerken. Prüfen Sie in einem angesehenen Computerladen, welche Disketten Sie für Ihren Mikrocomputer gebrauchen können. Besser noch, fragen Sie Ihren Computer-Verkäufer.

Ein kurzer Überblick über die populären Typen von Disketten und Mikrocomputern wird in Anhang E gegeben.

Disketten-Beschreibung

Wie beschrieben sind Disketten flexibel, und deshalb werden sie manchmal Floppies genannt. Sollte gerade eine Extra-Diskette zur Hand sein, so nehmen Sie sie, denn wir machen jetzt eine „Disketten-Tour".

Zunächst bemerken wir, daß die Diskette in einem stabilen Papierumschlag liegt. Dieser schützt ungefähr zwei Drittel der Diskette vor solchen Datenkillern wie Schmutz, Flüssigkeit oder Daumenabdrücken. Da der Umschlag aber recht dünn ist, ist auch der Schutz begrenzt – seien Sie also vorsichtig! Viele Disketten-Hersteller drucken Behandlungshinweise auf die Rückseite dieses Umschlags. Lesen Sie jede Information auf Ihrer Diskette; eines Tages kann sich daraus der Unterschied ergeben, ob Sie viele Stunden lang neu eintippen oder einen angenehmen Abend zuhause verbringen.

Nehmen Sie die Diskette sorgfältig aus dem Umschlag (sie gleitet rechts heraus). Wenn es aussieht, als müßten Sie etwas aufschneiden, so verwechseln Sie die Diskettenhülle mit dem Umschlag. Die Diskette hat eine quadratische Vinyl-Hülle, die eine dünne runde Scheibe schützt. Die Hülle hat (wie auch die Platte) ein rundes Loch in der Mitte und einen länglichen Ausschnitt an einer Kante (s. Abb. 1–4), außerdem noch ein kleines Loch an einer Seite des Zentrierungsloches.

Diese Teile werden wie folgt bezeichnet:

Zentrierungsloch.

Der Plattenlauf-Mechanismus verbindet sich durch dieses Loch mit der Diskette und läßt sie innerhalb der Hülle rotieren.

Indexloch.

Das Laufwerk findet hier den Anfangssektor (und im Falle von Hartsektoren-Disketten jeden einzelnen Sektor) für jede Spur der Diskette. Wenn sie sich an dieser Stelle eine senkrechte Linie auf der Plattenoberfläche denken, wartet das Laufwerk auf diese Startlinie und zählt von da an die Informationszeichen.

Zugriffsloch.

Durch diesen Ausschnitt kommt der Schreib-/Lesekopf des Laufwerks mit der magnetischen Oberfläche der Diskette in Kontakt. Er bewegt sich in dieser Öffnung zurück und vorwärts, von Spur zu Spur. Beachten Sie, daß sich auf beiden Seiten der Diskette Zugriffslöcher befinden.

Raste.

Dies ist eine Schreibschutzsperre. Auf Diskette schreiben heißt Information

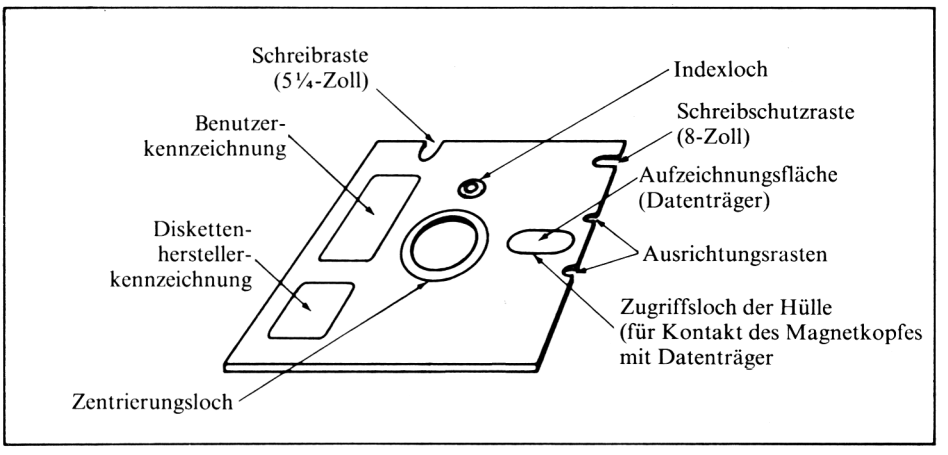


Abb. 1–4. Eine typische Diskette

hinzufügen. Hier kann Verwirrung entstehen: auf 8-Zoll-Disketten kann man schreiben, wenn die Raste bedeckt ist, sonst nicht. Bei 5¼-Zoll-Disketten ist es umgekehrt.

Außerdem befindet sich die 8-Zoll-Schreibschutzraste in der Nähe des Zugriffsloches, die der 5¼-Zoll-Diskette dagegen an der Seite (s. Abb. 1–4).

Als ob dies nicht genug wäre, kaufen einige Hersteller Disketten ohne Schreibschutzraste. Software-Händler, die Programme auf 5¼-Zoll-Disketten liefern, benutzen normalerweise diese Disketten, damit niemand Änderungen auf den Originaldisketten vornehmen kann.

Behandlung der Disketten

Nun, da Sie mit den Teilen der Diskette vertraut sind, ist es Zeit zu lernen, wie sie zu behandeln ist.

Zunächst die Verbote:

Berühren Sie nie die Diskettenoberfläche!

Wenn nötig, können Sie die Vinylhülle anfassen, aber niemals die magnetische Oberfläche der Diskette. Egal, wie sauber Ihre Hände sind, selbst der kleinste Rückstand kann Ihren Computer daran hindern, bestimmte Daten zu lesen.

Halten Sie Disketten von Magneten fern!

Das scheint überflüssig, nicht wahr? An Ihrem Arbeitsplatz befinden sich keine Magnete? Überlegen Sie genau! Wenn Ihr Computer ein Videoterminal hat (eines mit Bildröhre), haben Sie einen Magneten. Auch starke elektrische Felder wirken magnetisch. Die Lautsprecher in einem Klangsystem enthalten mit hoher Wahrscheinlichkeit Magnete. Tatsächlich reichen die magnetischen

Felder der meisten Geräte nicht aus, um Disketteninformation zu löschen, es sei denn durch direkten Kontakt. Dennoch ist es eine gute Regel, Disketten mindestens 30 cm von allem Magnetischen fernzuhalten. Auch ist es, entgegen landläufiger Ansicht, das Beste, Datenträger in Metallboxen und nicht in Plastikmagazinen aufzubewahren. Plastik hält unerwartete Felder nicht auf, wohl aber Metall.

Biegen Sie Ihre Disketten nicht!

Die Informationen auf einer Diskette sind sehr eng gespeichert. Überlegen Sie einen Moment: 2 000 000 Zeichen (auf einigen Disketten) in einem Bereich von zwei Kreisflächen mit je 8 Zoll Durchmesser. Jeder Kniff auf der Diskette kann bewirken, daß der Lesekopf des Laufwerkes den Kontakt mit der Plattenoberfläche verliert und eine Menge Information dadurch verloren geht.

Bewahren Sie Disketten nicht in unsauberer Umgebung auf!

Wenn Sie Mitarbeitern erlauben, zu essen, zu trinken oder zu rauchen, während sie am Computer sitzen, laden Sie das Mißgeschick zu sich ein. Filmemacher verwenden routinemäßig Cola, um Geräusche von magnetischen Tonspuren zu entfernen. Sollten Sie versehentlich ein Glas Coke über Ihrer Diskette verschütten, werden Sie wahrscheinlich ihre gesamte Information löschen.

Lassen Sie beim Ausschalten nie eine Diskette im Computer!

Nehmen Sie Disketten vor dem Ein- oder Ausschalten von Computer oder Laufwerken heraus. In den meisten Fällen wird nichts passieren, wenn Sie diese Regel nicht einhalten. Es könnte jedoch eine Energiespitze den magnetischen Kopf des Laufwerks treffen und dieser unsinnige Information mit verderblichen Folgen schreiben.

Lassen Sie Ihre Disketten nicht zu voll werden!

Viele Programme benutzen eine Diskette zur Datenspeicherung oder -generierung. Wenn diese temporären oder neuen Daten nicht mehr auf die Diskette passen, könnten Sie sie alle verlieren.

Zwei Faktoren begrenzen die Diskettenkapazität: die Anzahl der Dateien und die Anzahl der Zeichen. In einigen Implementationen erlaubt CP/M nur 32 Dateien, während manche der Versionen 2.2 von CP/M-80 bis zu 8192 Dateien auf einer Hartplatte ermöglichen. Die meisten Versionen von CP/M-80 werden 64 oder 128 Dateien auf einer Floppy-Diskette, 256, 512 oder 1 024 Dateien auf einer Hartplatte erlauben.

Die Anzahl der Zeichen auf einer Diskette wird durch das Laufwerk bestimmt und bewegt sich bei Floppies zwischen 80 000 und 2 000 000 Zeichen. Hartplatten-Laufwerke können min. 5 MB und max. 30 oder mehr MB speichern.

Nun einige Gebote:

Legen Sie Disketten immer langsam und vorsichtig ein!

Viele der 5¼-Zoll-Laufwerke haben eine sehr kleine Ausrichtungstoleranz zwischen Diskette und Magnetkopf. Besonders Micropolis-Laufwerke können eine hastig eingelegte Diskette falsch zentrieren, denn sie positionieren diese selbst. Disketten sind zerbrechlich; sie sollten sorgfältig und überlegt gehandhabt werden. Die Sekunde, die man durch Hast beim Einlegen spart, kann Stunden kosten, wenn man verlorene Information neu eingeben muß.

Etikettieren Sie alle Ihre Disketten!

Kaum etwas ist frustrierender, als fünfzig identisch aussehende Disketten zu haben und ihre Inhalte nicht zu kennen. Disketten haben, besonders wenn Sie den später in diesem Kapitel gegebenen Sicherungsratschlägen folgen, die Tendenz, sich zu vermehren wie Kaninchen. Nehmen Sie dieses Buch! Sein Text würde gerade auf eine Diskette hoher Kapazität passen. Nun sollen aber Kopien der letzten drei Revisionen zuzüglich ihrer Backups (= Sicherungen) aufbewahrt werden. Zusätzlich belegt das Text-Editor-Programm den größten Teil einer weiteren Diskette. Zählen Sie die Programmdisketten, eine Reihe weiterer Disketten für wichtige Aufzeichnungen und Daten sowie einige für Spiele und Hobbies hinzu, so können Sie sich den Stapel von Papierumschlägen vorstellen, der sich ansammeln kann.

Also aufgepaßt! Etikettieren Sie Ihre Disketten! Entwickeln Sie ein System, die älteren Versionen oder Kopien Ihrer Daten von den laufenden (in Gebrauch befindlichen) zu unterscheiden. Das Label (= Etikett) sollte Ihren Namen und das Datum, den Namen und die Versionsnummer des Betriebssystems sowie eine Kurzbeschreibung der Platteninhalte enthalten.

Halten Sie Backup- (= Sicherungs-) und selten benutzte Disketten vom Computer fern!

Bewahren Sie nur die Disketten beim Computer auf, die Sie ständig gebrauchen. Warum sollten Sie es sich unbedingt schwer machen, die benötigte Diskette aufzufinden?

Stellen Sie sicher, daß Ihre Disketten in korrekter Lage aufbewahrt werden!

Wie Schallplatten werden auch Disketten beschädigt, wenn man sie für längere Zeit anders als horizontal oder exakt vertikal aufbewahrt. Bei Schräglage wird die Schwerkraft die Diskette mit der Zeit leicht biegen. Wenn Disketten auch nicht wellig werden wie Schallplatten, werden sie doch von einem bestimmten Grad an leicht geknifft. Wenn sie Disketten horizontal lagern, türmen Sie nicht zuviele aufeinander. Hersteller empfehlen, nicht mehr als zehn Disketten übereinander zu stapeln.

Pflegen Sie Ihr Equipment (= Geräte-Ausstattung)!

Laufwerke sind nicht so schwierig zu pflegen wie Autos; sie brauchen nicht alle 15 000 km einen Ölwechsel. Auch kann ein Disketten-Equipment überge-

pflegt werden. Konstante Korrektur des Mechanismus kann die in die Laufwerke eingebauten Toleranzen zunichte machen. Ständiges Reinigen der Magnetköpfe kann für sie schädlicher sein als normale Abnutzung. Benutzen Sie eine Reinigungsdiskette (bei Ihrem Händler erhältlich) einmal in drei Monaten oder wenn Sie Fehler im Plattenlaufwerk befürchten. Die neuesten Laufwerke benötigen Pflege nur noch einmal im Jahr, und im Falle der neueren Winchester-Technologie ist bei Hartplattenlaufwerken nichts mehr zu korrigieren – der Mechanismus der Einheit ist versiegelt.

Kaufen Sie Qualitätsdisketten!

Versandhausdisketten liegen preislich unter dem, was die meisten Händler bezahlen. Bei diesem Preis sind die Disketten nicht verifiziert oder auf ihre Fähigkeit, Daten zu speichern oder wiederzugeben, geprüft. Man kann die Qualitätsunterschiede zwischen Disketten hören; eine Diskette niederer Qualität wird raspelnde Geräusche von sich geben. Sie werden Ihre persönlichen oder geschäftlichen Aufzeichnungen nicht auf Papierservietten führen, so vertrauen Sie auch Ihre wertvollen Informationen nicht deren Computer-Äquivalent an!

Einlegen und Herausnehmen von Disketten

Entnehmen Sie dem Handbuch Ihres Mikrocomputers, wie eine Diskette ins Laufwerk gehört. Bei horizontal montierten Laufwerken halten Sie die Diskette (achten Sie darauf, nicht die Magnetoberfläche zu berühren) auch auf horizontaler Ebene. Die Label-Seite sollte oben liegen. Legen Sie das Ende mit dem Zugriffsloch (dem länglichen) zuerst ins Laufwerk (s. Abb. 1–5). Die Schrift auf dem Label (= Kennzeichnung) wird normalerweise von Ihnen wegzeigen, wenn Sie die Diskette einlegen. Beispiele, für die die oben beschriebene Handhabung richtig ist, sind Apple-, IBM- und Osborne-Computer.

Doch auch hier gibt es Ausnahmen. Altos z. B. montiert seine Laufwerke umgekehrt in das Chassis, um die Handhabung zu erleichtern. Die Label-Seite zeigt dann nach unten.

Vertikal montierte Laufwerke sind etwas schwieriger zu beschreiben, denn die Kennzeichnung kann entweder nach links oder nach rechts zeigen, wenn Sie eine Diskette einlegen. Wenn Sie eine Diskette verkehrt herum einlegen, kann sie beschädigt werden; im Zweifel schauen Sie auf alle Fälle noch einmal im Handbuch für Ihren Mikrocomputer nach.

Nahezu alle Laufwerke zeigen dem Benutzer durch ein kleines rotes Licht, die Plattenaktivitätslampe, an, wenn gerade auf eine Diskette zugegriffen wird. Legen Sie keine Diskette ins Laufwerk oder nehmen eine heraus, wenn diese Lampe brennt; es sei denn, daß das Computer-Handbuch etwas gegenteiliges sagt. Bei vielen Systemen kann dieses bewirken, daß Information quer über die Diskettenoberfläche geschrieben wird. Natürlich kann die Integrität der Diskettendaten besonders dann beschädigt werden, wenn der Computer gerade schreibt (mehr als beim Lesen).

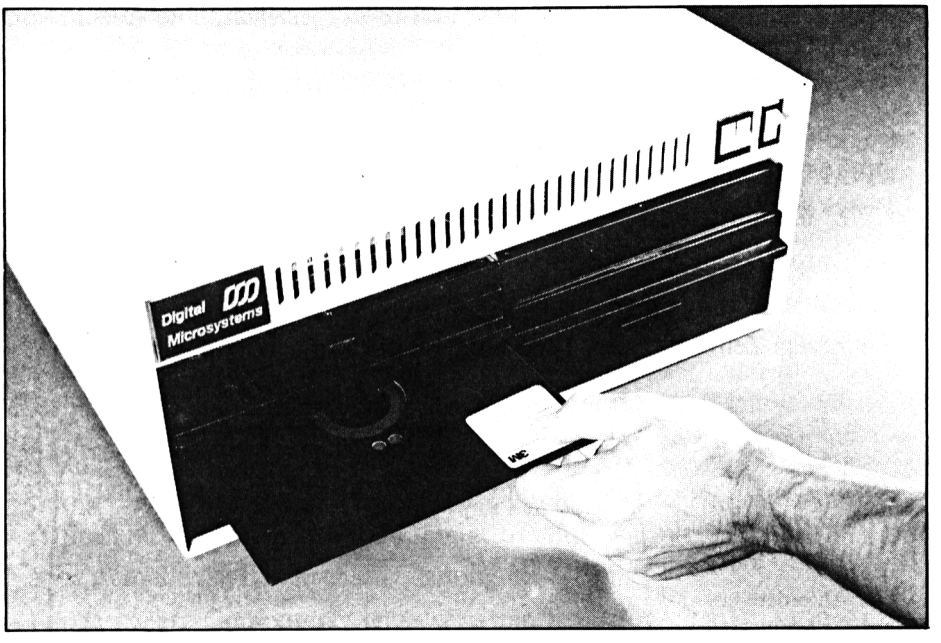


Abb. 1–5. Einlegen einer Diskette in ein horizontal montiertes Laufwerk

Die vernünftige Behandlung von Disketten ist eine Sache des gesunden Menschenverstandes. Obwohl Disketten dem Computer-Novizen neu sind, gibt es keine Entschuldigung dafür, die Folgen eines Mißbrauchs zu ignorieren. CP/M kann nur so gut sein wie die Umgebung, die Sie ihm schaffen. Pflegen Sie deshalb das Equipment und behandeln Sie Ihre Diskettendaten sorgfältig.

Wie man CP/M startet

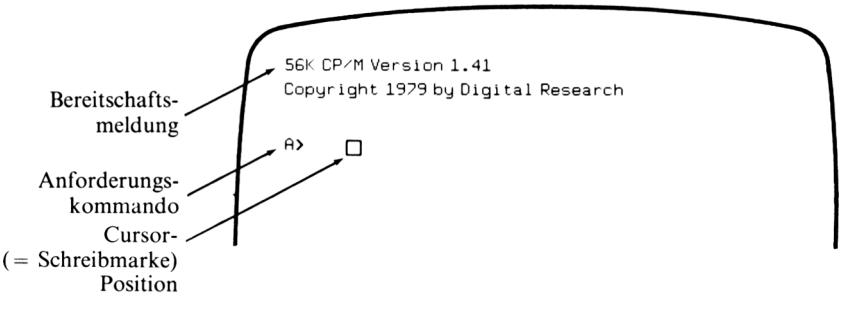
Wenn eine CP/M-Version in Ihren Computer geladen wird, geschieht mehreres. Ein Kaltstartlader überträgt es als Speicherinhalt der Diskette in den Kernspeicher Ihres Computers. Dieser Lader ist von Maschine zu Maschine und möglicherweise zwischen den einzelnen CP/M-Versionen verschieden, aber seine Funktion ist immer die, CP/M zu laden. Der Kaltstartlader ist ein getrenntes Programm, das zusammen mit CP/M auf der Diskette gespeichert ist.

Warum lädt man etwas in den Computer, um etwas anderes zu laden? Die Gründe sind komplex (erläutert in Kap. 7), aber in erster Linie ermöglicht es Maschinenunabhängigkeit, was bedeutet, daß völlig unterschiedliche Computer die gleichen Laufwerke und die gleiche Kopie von CP/M benutzen können.

Nachdem der Kaltstartlader CP/M anfordert, geschieht eine Anzahl von Dingen in folgender Reihenfolge:

- CP/M wird in den Kernspeicher Ihres Computers geladen
- die Computer-Kontrolle wird an CP/M gegeben
- CP/M führt verschiedene Initialisierungs-Operationen aus
- CP/M übermittelt eine Bereitschaftsmeldung auf Ihrem Bildschirm
- die Anforderung A erscheint auf dem Bildschirm
- schließlich wartet CP/M auf Ihr Kommando.

Zu diesem Zeitpunkt sieht Ihr Sichtgerät etwa folgendermaßen aus:



Wie Sie sehen, tut CP/M nichts Mysteriöses, wenn Sie Ihren Computer starten. Das Schwirren und Klicken der Laufwerke zeigt an, daß CP/M in den Kernspeicher übertragen wird und alles in Ordnung ist.

Als nächstes sei die Initialisierung von CP/M als physikalischer Prozeß betrachtet. Da jedes Computersystem verschieden ist, wollen wir in generellen Begriffen sprechen. Wenn Sie sich über die korrekte Prozedur nicht sicher sind, lesen Sie die Begleithandbücher!

Zuerst müssen einige periphere Einheiten eingeschaltet werden. Wenn Sie ein Hartplatten-Laufwerk haben, schalten Sie zunächst einmal dieses ein, denn es braucht meist mehrere Minuten, um „auf Touren zu kommen" und benutzt werden zu können.

Wenn Sie Floppy-Laufwerke haben, legen Sie noch keine Diskette ein (wenn Ihnen Ihre Computer-Handbücher dies nicht vorschreiben). Wenn Sie ein Diskette einlegen und danach den Computer ein- oder ausschalten, kann ein kleiner transienter Stromstoß an den Schreib/Lese-Kopf erfolgen, der Disketteninformation löscht oder ändert.

Schalten Sie den Computer ein. In einigen Fällen versucht er sofort, Programme von der Diskette zu laden. Legen Sie keine Diskette ins Laufwerk, während er gerade zuzugreifen versucht.

Die meisten Hersteller sehen für den Start eine Spezial-Routine vor. Wenn Sie solche Maschinen „hochfahren", geben sie über Bildschirm eine Nachricht aus und warten auf Ihre Instruktion. Einige erwarten, daß Sie mehrere Male die RETURN-Taste drücken, um die Geschwindigkeit zu bestimmen, in der die Tastatur Zeichen sendet. Normalerweise ist die Instruktion ein einfacher Buchstabe (wie „B" für „boot"). Bei diesen Systemen schalten Sie den Strom ein, legen Ihre Diskette ins erste Laufwerk und drücken die „B"-Taste oder die Instruktion, die Ihr System benötigt.

Hier ist ein Überblick über die besprochenen Schritte, ein System zu starten:

1. Schalten Sie den Strom der peripheren Einheiten ein
2. Legen Sie die Systemdiskette ein (einige Systeme)
3. Schalten Sie den Computer ein
4. Legen Sie die Systemdiskette ein (die übrigen Systeme)
5. Drücken Sie die CARRIAGE RETURN-Taste (einige Systeme)
6. Tippen Sie die Startinstruktion (einige Systeme – normalerweise „B")
7. Erwarten Sie die Bereitschaftsmeldung des CP/M.

Einige neuere Computersysteme erwarten beim Einschalten automatisch eine CP/M-Diskette. Für diese Systeme ist der normale Prozeß des Hochfahrens (start-up) wie folgt abgekürzt:

1. Peripherie einschalten
2. Diskette einlegen
3. Computer einschalten
4. CP/M-Bereitschaftsmeldung erwarten

Entnehmen Sie Ihrem Computer-Handbuch die Startprozedur Ihres Systems.

Täglich wiederkehrende Vorgänge

Für den täglichen Betrieb benötigen Sie Kenntnisse über einige Routinevorgänge:

RESET (Computer auf Grundstellung):
Das Drücken des Panik-Knopfes

Es gibt einige verhängnisvolle Fehler, die die Daten auf Ihren Disketten in Gefahr bringen können. „Fatal errors" (= Fehler, die nicht wiedergutzumachen sind) erhalten Sie sicher, wenn Sie den Strom abschalten oder den RESET-Knopf drücken, während die Laufwerke Information auf Diskette schreiben. Leider läßt sich nicht sagen, man solle grundsätzlich nicht den Strom abschalten oder RESET drücken. So muß man sich möglicherweise entscheiden, RESET zu drücken, um eine „Endlos-Schleife" kontinuierlich

ausgeführter Instruktionen abzubrechen. Gründlich ausgetestete Programme bleiben selten in solch einer Schleife stecken, aber neu entwickelte Programme tun dies gelegentlich.

Wenn eine Endlos-Schleife die Instruktion enthält, Information auf Diskette zu schreiben, könnte sich die Plattenaktivitäts-Lampe einschalten und anbleiben. Nachdem Sie einige Minuten darauf gewartet haben, daß etwas geschieht, könnten Sie die Panik-Taste drücken. Stellen Sie sicher, daß Ihr Computer in einer Endlos-Schleife kreist, bevor Sie etwas unternehmen. Gute Programme werden periodische Meldungen herausgeben wie

I'M WORKING oder THIS MAY TAKE A WHILE.

Wenn auch diese Meldungen nicht ganz narrensicher sind, beruhigen sie doch etwas, wenn sich die Lampe einschaltet und anbleibt. Wenn Sie völlig davon überzeugt sind, daß Ihr Computer in einer Endlos-Schleife kreist, drücken Sie RESET. Fast alle Computer haben diesen Knopf, bei manchen ist er jedoch auf der Rückseite versteckt. Bei den meisten hat er den gleichen Effekt, als ob man den Computer aus- und dann wieder einschaltet. Jedoch wird der Stromfluß zu keinem Gerät unterbrochen. Einige Computer behalten sogar das Programm oder Daten im Kernspeicher, aber normalerweise ist dies nicht der Fall.

Wenn Ihre Handbücher nicht klar zum Ausdruck bringen, was nach dem Drücken von RESET zu unternehmen ist, rufen Sie Ihren Computerladen an, bevor Sie experimentieren. Sollten Sie wichtige Daten im Computer haben und sie schützen wollen, sollten Sie CP/M-80 oder ein anderes Programm nicht neu laden, denn das würde die Daten wahrscheinlich zerstören. Bei CP/M-86 beträgt diese Wahrscheinlichkeit etwa 50%, je nach Computer. Trotzdem ist es am besten, das Schlimmste anzunehmen und nicht automatisch davon auszugehen, daß man die Information im Kernspeicher erhalten kann. Seien Sie in diesem Fall lieber freudig überrascht. Die Chance, die Information im Kernspeicher zu behalten, mag gering sein, aber besser als gar keine. Unternehmen Sie nichts nach dem Drücken von RESET, bevor Sie sicher sind, daß die Information im Kernspeicher verloren ist.

Plattensicherung

So ziemlich jede Ausführung von CP/M hat ein Kopier-Programm (s. Kap. 5). Bevor Sie Ihre Arbeit am Computer beenden, machen Sie Kopien von allen Platten, die Sie geändert oder aktualisiert haben; dieses Vorgehen wird „backup" genannt.

Hier sind unsere Vorschläge zur Plattensicherung:

1. Beschriften Sie alle Ihre Disketten, wobei sich das Original von der Kopie unterscheiden sollte. Ein Weg ist die „Vater-Sohn"-Methode. Das Original heißt „Vater", die Kopie heißt „Sohn". Sollten Sie noch eine weitere Kopie für Archivierungszwecke benötigen, so ist das Original der Großvater. Es

besteht jedoch die Gefahr, daß sich Gewöhnung an unregelmäßige Sicherung einschleicht.

Eine weit bessere Methode ist deshalb die rotierende Backup-Prozedur. Sie haben eine Diskette für jeden Werktag. Montagabend kopieren Sie die Mo- auf die Di-, Dienstagabend die Di- auf die Mi-Diskette usw. Vorausgesetzt, die Disketten sind sorgfältig etikettiert, sind Sie in jedem Fall sicher, die richtige Diskette zu benutzen und Sie haben somit eine natürliche Backup-Folge.

2. Disketten halten nicht ewig, allein schon, weil der physische Kontakt zur Hülle zu stark ist. Auch ein schlecht ausgerichtetes Laufwerk wird gelegentlich zu Schreibfehlern führen.

   Die richtige Einschätzung der Lebensdauer einer Diskette ist daher äußerst wichtig. Es gibt Hersteller, die behaupten, man könne ihre Disketten ein Jahr lang kontinuierlich benutzen. Wir empfehlen jedoch, falls Sie die rotierende Backup-Prozedur anwenden, die Disketten nach 6 bis 12 Monaten auszutauschen. Das ist eine sehr vorsichtige Arbeitsweise, denn jede Diskette wurde dann nur 27 bis 54 Tage lang benutzt. Wenn man jedoch den Diskettenpreis mit den Kosten einer Neueingabe der Daten vergleicht, erscheint dieses Vorgehen weise.

3. Benutzen Sie nie die Programme auf ihren Original-Disketten! Wenn Sie ein neues Programm (oder ein CP/M-Update) auf einer Diskette erhalten, machen Sie eine Kopie, etikettieren Sie die Original-Diskette als „Master" und verwahren Sie sie gut. Es wäre töricht, mit dem Original zu arbeiten. Eines Tages könnte Ihr Computer müßig stehen, während Sie auf eine Ersatz-Diskette warten. Die meisten Software-Verkäufer geben ein Update auch nur gegen Rückgabe des Originals heraus, also bewahren Sie es an einem sicheren Platz auf.

Abschließen des Systems

Zuerst haben Sie gelernt, Ihr Computer-System hochzufahren. Die Prozedur des Abschließens ist nicht einfach der umgekehrte Vorgang.

Erstens: Benutzen Sie den normalen Abschluß eines jeden Programmes. Das ist sehr wichtig, denn viele Programme schließen das Schreiben von Information auf Diskette nicht richtig ab, wenn man ihnen nicht die korrekte Nachricht eingibt.

Zweitens: Nehmen Sie alle Disketten aus ihren Laufwerken; danach schalten Sie Laufwerke, Terminal, Drucker und sonstige Peripherie aus, den Computer zuletzt. Sollten Sie Hartplattenlaufwerke haben, schalten Sie sie vor dem Mikrocomputer aus, damit Ihnen keine Informationen verlorengehen. Halten Sie sich aber in jedem Fall an die Anweisungen Ihres Handbuches.

Kapitel 2
CP/M – Eingebaute Kommandos

In Kapitel 1 wurde beschrieben, wie CP/M zu starten ist. Nun ist es an der Zeit, daß Sie CP/M für sich arbeiten lassen. Dieses Kapitel untersucht die eingebauten Kommandos in CP/M-80 und CP/M-86.

Kommandos sind Instruktionen

Kommandos sind Instruktionen an CP/M. Wenn CP/M bereit ist, ein Kommando entgegenzunehmen, schreibt es eine Anforderung (normalerweise A>). Damit CP/M ein Kommando ausführt, tippen Sie es ein und drücken Sie die CARRIAGE RETURN-Taste.

Alle Versionen von CP/M haben zwei Arten von Kommandos: eingebaute und sogenannte transiente (flüchtige) Kommandos.

Der Unterschied ist subtil. Die Kurzprogramme, die eingebaute Kommandos ausführen, befinden sich zusammen mit CP/M ständig im Kernspeicher. Die Programme, die transiente Kommandos ausführen, werden beim Start von CP/M nicht automatisch in den Kernspeicher geladen. Ein transientes Kommando führt dazu, daß CP/M ein Programm von der Systemdiskette lädt und ausführt. Sonst befinden sich transiente Programme nicht im Kernspeicher.

Eingebaute und transiente Kommandos unterscheiden sich also folgendermaßen:

- Ein eingebautes Kommando wird von CP/M sofort ohne vorherige Instruktionen von der Diskette ausgeführt.
- Ein transientes Kommando erfordert vor jeder Anwendung, daß eine Reihe von Instruktionen von der Systemdiskette in den Kernspeicher geladen wird. Eingebaute wie transiente Kommandoprogramme werden durch das Eintippen eines Befehls wie LOAD oder DIR als Antwort auf die CP/M-Anforderung aufgerufen.

Kommandos arbeiten mit Plattendateien.

CP/M-Kommandos greifen auf Informationen zu, die in Dateien auf Disketten gespeichert sind und verändern diese. Eine Datei ist jede Information, die als einzelne Einheit unter einem eindeutigen Namen gespeichert ist. Die Länge einer Datei kann von 0 Zeichen bis hin zur maximalen Kapazität einer Diskette variieren. Hier sind einige Beispiele von Dateigrößen:

- ein einzelnes Programm
- alle Daten für ein Programm
- eine komplette Adressenliste

- eine einzelne Liste
- eine große Gruppe von Standardbriefen
- ein Kapitel dieses Buches
- der gesamte Inhalt dieses Buches.

Es gibt keine Regel dafür, welche Information man in einer Datei speichern kann. Man definiert die Form des Inhalts, wenn man die Datei erstellt.

Alle Dateien bestehen aus Feldern: einzelnen Wörtern, Zahlen oder anderen Informationseinheiten in zweckmäßiger Größe. Man kann Dateien auch in Sätze (= records) untergliedern, um eine andere Größenordnung von Information widerzuspiegeln.

Wie Dateien in Sätze und diese wiederum in Felder unterteilt werden, wird bei der Erstellung bestimmt. Nehmen Sie z. B. eine Adreßliste. Alle Namen und Adressen können eine einzelne Datei bilden, aber jeder einzelne Name samt Adresse könnte als ein Satz bezeichnet werden, wobei dann der Name oder jeweils eine Adreßzeile jeweils ein Feld ergäben:

```
    FILE: [= = = = = = = NAME AND ADDRESS FILE = = = = = = =]
 RECORDS: [- - name/address/city - -] [- - name/address/city - -]
  FIELDS: [name]  [address]  [city]  [name]  [address]  [city]
```

Zusammenfassung der eingebauten Kommandos

CP/M-80 Version 1.4 kennt sechs, die Version 2.2 sieben und CP/M-86 wiederum sechs eingebaute Kommandos:

CP/M-80 Version 1.4	CP/M-80 Version 2.2	CP-M-86
DIR	DIR	DIR
TYPE	TYPE	TYPE
ERA	ERA	ERA
REN	REN	REN
SAVE	SAVE	
d:	d:	d:
	USER	USER

Zusätzlich zu den grundlegenden eingebauten Kommandos gibt es verschiedene direkt interpretierte Zeilenausgabe-Kommandos. Es sind:

^C ^E ^H ^J ^M ^P ^R ^S ^U ^X

Wir werden jedes dieser Kommandos später in diesem Kapitel detailliert beschreiben; jedenfalls sehen Sie, daß nur wenige zu erlernen sind.

Wie man Kommandos eingibt

Bevor die Funktion eines jeden Kommandos beschrieben wird, müssen Sie die von uns benutzten Bezeichnungen der Befehle und Tastenanschläge kennen:

Drücken	bedeutet das Tippen einer einzelnen Taste
Tippen	bedeutet das Drücken einer Folge von Tasten
<cr>	bedeutet das Drücken von CARRIAGE RETURN (auf manchen Maschinen auch ENTER)
^X	bedeutet, das Kontrollzeichen „X" (CNTRL-X) einzugeben

Unterstreichungen werden angewandt, um die Benutzer-Eingabe von der Computer-Ausgabe zu unterscheiden (wenn nötig); in den angegebenen Beispielen ist Ihre Eingabe **unterstrichen**.

Beachten Sie, daß die CARRIAGE RETURN-Taste durch das Symbol <cr> repräsentiert wird. Ein typisches Kommando-Beispiel wäre:

```
DIR <cr>
```

Das bedeutet, daß man die Buchstaben „D", „I" und „R" eingibt und danach die CARRIAGE RETURN-Taste drückt. Sie wird auch mit anderen Abkürzungen wie RETURN, CARR RET, CR oder ENTER bezeichnet.

Kontrollzeichen haben besondere Bedeutung für Computer. So wie man die SHIFT-Taste drückt, um einen Großbuchstaben auf einer Schreibmaschine zu erzeugen, wird die CONTROL-Taste gedrückt, um ein Kontrollzeichen einzugeben. (s. Abb. 2-1). ^C bedeutet also, die CONTROL-Taste zu drücken, zu halten, den Buchstaben „C" zu drücken und die CONTROL-Taste wieder loszulassen.

Die Control-Taste ist oft als CTRL, als CTL oder manchmal auch als ALT beschriftet; man findet sie fast immer in der unteren linken Ecke der Tastatur nahe der linken SHIFT-Taste.

Ihre Konsole listet sowohl Ihre Eingabe als auch die Antwort des Computers. Dieses Buch benutzt folgende Kennzeichnung, um zwischen den Ausgaben des Computers und den auf der Tastatur getippten Eingabedaten zu unterscheiden: die Eingabe ist unterstrichen, die Computerausgaben nicht.

Hier ist ein Beispiel:

```
A>DIR B:<cr>
A:STAT  COM | MBASIC4 COM | BACKUP COM |  MBASIC5  COM
A:CBAS2 COM |  CRUN2  COM |   XREF  COM |     PIP   COM
A>MBASIC5 <cr>
```

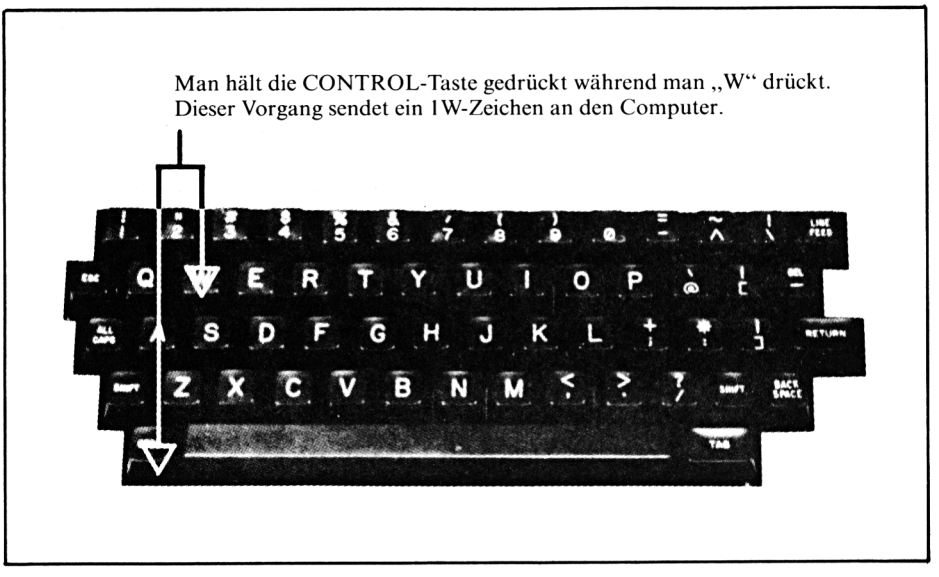


Abb. 2–1. Eingabe eines Kontrollzeichens

```
Microsoft BASIC version 5.3
copyright 1977, 1978, 1979, 1980
by Microsoft. Inc.
14,568 bytes free
OK
```

In diesem Beispiel gibt der Computer auf dem Bildschirm (oder Drucker) die Anforderung A> aus. Sie tippen DIR<cr>. Der Computer antwortet mit einer Auflistung von acht Dateinamen und einem weiteren A>. Danach tippen Sie MBASIC5<cr>, und eine 5-Zeilen-Meldung zeigt an, daß Microsoft-BASIC geladen und bereit ist.

Das Leerzeichen, das durch Drücken der Leertaste (SPACE BAR) getippt wird, wird in Kommandos häufig benötigt. Es ist in einem Computersystem so wichtig wie irgendein anderes Zeichen. Tatsächlich waren frühe Versionen des CP/M-80 recht pingelig in bezug auf Leerzeichen, die Version 2.2 allerdings weniger. Geben Sie die Leerzeichen in Beispielen mit ein.

Groß- und Kleinbuchstaben werden in Kommandozeilen und anderen Beispielen von Anwendereingaben benutzt, um die festen und die veränderlichen Anteile der Eingabe zu unterscheiden. Den Großbuchstaben-Anteil müssen Sie exakt so eintippen, wie er erscheint; dieser unveränderliche Teil Ihrer Eingabe ist normalerweise das Kommando. Im Falle des Kleinbuchstaben-Anteils müssen Sie die Information Ihrer Wahl unterstützen; dies ist der variable Teil Ihrer Eingabe. So ist z. B.

```
DIR *.typ <cr>
```

eine Form der Directory-Kommandos

Sie müssen im o.a. Kommando „DIR" eingeben, aber „typ" müssen Sie durch den Dateityp Ihrer Wahl ersetzen. Hier sind einige gültige Beispiele für den o.g. Befehl:

```
DIR *.ASM <cr>
DIR *.BAS <cr>
DIR *.COM <cr>
```

CP/M – Datei-Konventionen

Jede CP/M-80- oder CP/M-86-Datei wird durch einen Dateinamen und einen Dateityp identifiziert. Diese Identifikationen werden jetzt im Detail besprochen.

Datei-Namen

Jede CP/M-80- oder CP/M-86-Datei hat einen eindeutigen Namen, der aus einem bis zu acht Zeichen besteht. Normalerweise werden für Dateinamen nur Großbuchstaben und Ziffern benutzt, obwohl es – z. B. im Microsoft-BASIC – möglich ist, eine Datei mit einem Namen aus Kleinbuchstaben oder sogar ohne Namen zu erstellen.

CP/M-80 wie CP/M-86 konvertieren in einer Kommandozeile (die einem A> folgt), automatisch Klein- in Großbuchstaben und werden einen leeren Dateinamen nicht akzeptieren. Verschiedene Zeichen dürfen in einem Dateinamen nicht verwendet werden; und zwar:

< > . , ; : = ? * [ ] ( ) / <TAB>

Da CP/M jedes dieser Zeichen in spezieller Weise benutzt, sind sie in Dateinamen ungültig.

Auch Kontrollzeichen (oder andere Zeichen, die auf dem Bildschirm nicht erscheinen) dürfen Sie in einem Dateinamen nicht benutzen. Hier sind einige Beispiele gültiger Dateinamen:

FILENAME	
IS-FILE	
OH!	
87654321	
A+B+C	Diese Kombination wird nicht empfohlen, denn das Pluszeichen kann gelegentlich falsch verstanden werden.

Die folgenden Dateinamen sind alle ungültig:

ISFILENAME	zu viele Zeichen
IS, FILE	enthält ein unerlaubtes Zeichen

<FILE>	enthält unerlaubte Zeichen
IS) FILE	enthält ein unerlaubtes Zeichen
FILE^C	enthält ein Kontrollzeichen
WORD*	enthält ein unerlaubtes Zeichen

Ein Dateiname sollte möglichst der Bedeutung entsprechen und eindeutig sein. Ihr Lohnprogramm könnte z. B. PAYROLL heißen. Es vereinfacht „PR" zu nennen, könnte vielleicht Tippanschläge sparen, einen möglichen Käufer Ihrer Programme jedoch irritieren.

Datei-Typen

Es ist auch vernünftig, Dateien so zu identifizieren, daß daraus die weitere Benutzung hervorgeht, denn unterschiedliche Informationstypen werden auch unterschiedlich gehandhabt. Bestimmte Operationen für Programmdateien könnten bei irrtümlicher Verwendung Inhalte von Datendateien zerstören.

Um dies zu verhindern, verlassen sich CP/M-80 und CP/M-86 auf den Dateitypen, um die Funktion der Datei zum Ausdruck zu bringen. Der Dateityp besteht aus bis zu drei Zeichen und wird vom Dateinamen durch einen Punkt getrennt.

Hier ist eine umfassende Liste der normalerweise von CP/M- und MP/M-Anwendern benutzten Dateitypen:

.ASC	Datei mit ASCII-Text
.ASM	Quellprogramm in Assembler-Sprache (CP/M-80)
.A86	Quellprogramm in Assembler-Sprache (CP/M-86)
.BAK	Backup-Datei (= Sicherungs-Duplikat)
.BAS	BASIC-Quellprogramm
.BRS	Banked resident system process (MP/M) (= gebunkerter Systemprozeß)
.C	C-Quellprogramm
.CAL	SuperCalc-Datei
.CMD	direkt ausführbares transientes Programm (CP/M-86)
.COB	COBOL-Quellprogramm
.COM	direkt ausführbares transientes Programm (CP/M-80)
.DAT	Daten-Datei
.DOC	Dokumentations-Datei (Text-Datei)
.FOR	FORTAN-Quellprogramm
.HEX	Intel HEX-Format – Objektcode-Datei (CP/M-80)
.H86	HEX-Format – Objektcode-Datei (CP/M-86)
.INT	BASIC-Zwischencode – Programm-Datei (CBASIC)
.IRL	Indexed Relocatable Library (= indizierte Bibliotheks-Datei)
.LST	Assembler-Druck-Datei (CP/M-86)
.MAC	Makro-Assembler-Quellprogramm
.OBJ	Maschinencode-Datei (Objektcode)

.OVR	Overlay-Datei (= Überlagerungs-Datei, wird von MicroPro und Sorcim benutzt)
.PAS	Pascal-Quellprogramm
.PCO	Pascal-Laufzeit-Programm (Sorcim Pascal/M)
.PLI	PL/I-Quellprogramm
.PRL	Page-Relocatable File (MP/M) (= um Kernspeicher-Seiten selbstverschiebliche Datei)
.PRN	Assembler-Druck-Datei
.REL	Relocatable (= selbstverschiebliche) Maschinencode-Datei
.RSP	resident-system process file (= residente System-Prozeß-Datei)
.SRC	Quellprogramm (von der CP/M-Anwendergruppe)
.SPR	System-Prozeß-Datei
.SUB	Kommando-Datei für SUBMIT-Programm
.TEX	Dokumentations-Datei (TEX file)
.TXT	Dokumentations-Datei (text file)
.$$$	temporäre Datei; unbenutzbar, wenn nicht gesichert

Wenn es auch noch andere Typen gibt, macht doch diese Liste 95% der gebräuchlichen aus. Wie Sie sehen, helfen einige Typen besondere Programme zu identifizieren und außerdem Dateien zu benutzen oder zu drucken.

Sie können auch Ihre eigenen Datei-Typen kreieren, z. B. die Versionsnummer eines Programms in den Stadien seiner Entwicklung. Angenommen, Sie arbeiten an einem BASIC-Programm namens THEBEST, könnte Ihre Liste von Dateien auf einer Platte wie folgt aussehen:

```
THEBEST.010
THEBEST.020
THEBEST.030
THEBEST.040
THEBEST.BAS
```

THEBEST.BAS repräsentiert das Endprodukt und jeder der anderen Typen ein unterschiedliches Entwicklungsstadium des Programms. Dateitypen können null bis drei Zeichen lang sein oder auch nichtvorhanden sein, d. h. der Dateityp kann manchmal – nicht immer, sondern programmabhängig – ausgelassen werden.

Wenn auch CP/M-80 und CP/M-86 den Dateityp in den meisten Fällen ignorieren, so fordern doch viele Systemprogramme spezifische Dateitypen. So braucht z. B. der CBASIC-Compiler für Quellprogramme den Typus „BAS", während die mit CBASIC kompilierten Programme den Typus „INT" benötigen. Aber diese Typen sind nur für CBASIC bedeutsam, CP/M-80 und CP/M-86 kennen nichts dergleichen.

In der frühen Dokumentation von Digital Research (übrigens auch der ersten Revision dieses Buches) werden oft Dateitypen als Datei-Erweiterungen beschrieben, wie es damals auch gedacht war. Jedoch bezeichnet der

Begriff „Typ" den Zusammenhang genauer, und so wollen wir ihn in diesem Buch benutzen. Sollten Sie in anderen Dokumentationen den Begriff „extensions" (= Erweiterungen) finden, so tauschen Sie ihn einfach gegen „types" (= Typen) aus.

Verbindung von Dateinamen und Dateitypen

Manchmal können sie in einem Kommando eine Datei einfach durch ihren Namen (der 8-Zeichen-Begriff) spezifizieren. Mitunter brauchen Sie aber auch Namen und Typ. Diese sind grundsätzlich durch einen Punkt getrennt, z. B.:

```
WATNOWMY.LUV
ROBOT.1
MICRO.SFT
THX1138.PIK
KINGAND.EYE
```

Es sind also weder acht Zeichen im Dateinamen noch drei im -typen verlangt. Ein Dateiname muß mindestens aus einem Zeichen bestehen, aber Dateien müssen keinen Typ haben. Mit anderen Worten, Sie können Dateinamen aus 1 bis 8 und Dateitypen aus 0 bis 3 Zeichen haben. Sowohl CP/M-80 als auch CP/M-86 ignorieren in einem Namen jedes Zeichen nach dem achten und in einem Typen jedes nach dem dritten.

```
CPMUSERGUIDE.BOOK
```

würde also interpretiert als

```
CPMUSERG.BOO
```

Ein Vorbehalt: Computersprachen sind noch nicht ausreichend standardisiert für allgemein gültige Definitionen. Manche Programmierer verstehen unter „Dateiname" noch die gesamte Kombination von Name, Punkt und Typ. Auch Begriffe wie „Dateireferenz" (= file reference), „Dateierweiterung" (= file extension), „primärer Name" (= primary name) oder „sekundärer Name" (= secondary name) werden in der Literatur gebraucht. Wir werden uns jedoch auf die Form

TYPE filename.typ <cr>

beschränken.

Sie würden also das Kommando TYPE, ein Leerzeichen, den Dateinamen, einen Punkt, den Dateityp und dann CARRIAGE RETURN eingeben. Beachten Sie dabei die Schreibweise dieses Buches für Kommandos (s. S. ■)!

Laufwerks-Bezeichnungen

CP/M benennt die verfügbaren Laufwerke mit „A", „B", „C", „D" usw. in alphabetischer Reihenfolge. Das erste oder primäre Laufwerk hat die Bezeichnung „A". Wenn allerdings Hartplatten- und Floppylaufwerke in einem

System gemischt sind, verwenden manche Hersteller für die ersteren die Buchstaben „M", „N", „O", „P" usw. Prüfen Sie dies in Ihrem Handbuch nach, wenn Ihr System mit gemischten Laufwerken arbeitet.

Da Dateinamen ebenfalls Buchstaben enthalten, wird ein Doppelpunkt (:) nach der Laufwerksbezeichnung hinzugefügt, z. B.

A:

In diesem Buch wie in allen CP/M-Handbüchern beziehen sich A:, B:, C: usw. durchweg auf Plattenlaufwerke. Das vorherige Beispiel würde also unter Einschluß einer Laufwerksbezeichnung so geschrieben werden:

TYPE d:filename.typ <cr>.

In diesem Falle würden Sie also das Kommando TYPE, ein Leerzeichen, eine gültige Laufwerksbezeichnung, einen Doppelpunkt, einen Dateinamen, einen Punkt, einen Dateityp und danach ein CARRIAGE RETURN eingeben.

Andere Plattenbetriebssysteme benutzen gelegentlich Ziffern zur Bezeichnung der Laufwerke. Es ist also vernünftig, Ihre Laufwerke CP/M nach CP/M zu bezeichnen.

Für den Start von CP/M-80 oder CP/M-86 werden Sie fast immer das Laufwerk „A" benutzen. Sollten Sie nichts anderes eingeben, wird „A:" als aktives Laufwerk für alle Kommandos und Programme angenommen. Wenn Sie Laufwerk „A" benutzen, müssen Sie die Laufwerksbezeichnung nicht eingeben.

Wildcard-Referenzen

Es kann vorkommen, daß Sie den Namen oder Typen einer Datei, die Sie benutzen wollen, nicht kennen; oder Sie wollen zur gleichen Zeit mehr als eine Datei spezifizieren. Digital Research hat dafür Wildcard-Referenzen, manchmal auch „vieldeutige Datei-Referenzen" (= ambiguous file references) genannt, vorgesehen. Der Stern (*) ersetzt, wenn er an Stelle eines Namens oder Typens gebraucht wird, den gesamten Anteil des Dateinamens oder -typens oder beides. Das Fragezeichen (?) in einem Namen oder Typen ersetzt ein einzelnes Zeichen innerhalb des Namens oder Typens.

In Handbüchern von Digital Research werden vieldeutige Dateinamen auch als „afn" (ambiguous file name) abgekürzt, während komplette Dateinamen als „ufn" (unambiguous file name) bezeichnet werden.

Viele CP/M-Kommandos erwarten den ufn, also den kompletten Dateinamen (Name und Typ) in der Kommandozeile für einen korrekten Aufruf.

Ansonsten kann ein Stern die Zeichengruppe an jeder Seite des Punktes in einem kompletten Dateinamen ersetzen. Es sind daher alle folgenden Beispiele als afn-Spezifikationen gültig:

*.BAS	Alle Dateien mit dem Typen BAS
THEBEST.*	Alle Dateien mit dem Namen THEBEST
.	Alle Dateien

Wozu das gut sein soll? Betrachten Sie das Kommando DIRECTORY (= Inhaltsverzeichnis)! Wenn Sie die Namen aller BASIC-Programme auf Laufwerk „A" erfahren wollen, tippen Sie die folgende Kommandozeile:

DIR *.BAS <cr> (DIR für Directory)

Sollten Sie alle löschen wollen, tippen Sie

ERA *.BAS <cr> (ERA für Erase).

Sicher haben Sie verstanden, daß man das Fragezeichen ähnlich wie den Stern gebrauchen kann. Wenn z. B. PROGRAM1.ASM die einzige Datei dieses Typs auf Laufwerk „A" wäre, würden die folgenden Kommandos von CP/M identisch behandelt:

ERA *.ASM <cr>	lösche alle Dateien des Typs ASM
ERA PROGRAM1.ASM <cr>	lösche die Datei PROGRAM1.ASM
ERA ???????1.ASM <cr>	lösche alle Daten des Typs ASM mit einer „1" an achter Stelle des Namens

Jedes Fragezeichen steht für eine Position, und „?1.ASM" oder „???????1.ASM" ist für CP/M nicht dasselbe.

Die meisten CP/M-Systeme haben zwei oder mehr Laufwerke. Sie erinnern sich, daß CP/M ein Laufwerk durch einen Buchstaben, gefolgt von einem Doppelpunkt, identifiziert. Das Kommando

A>DIR B:*.* <cr>

listet die Namen aller Dateien in Laufwerk „B". Die Anforderung A> zeigt an, daß Laufwerk „A" angenommen wird. Wenn wir nicht explizit „B" angegeben hätten, wären aufgrund der Annahme von „A" alle Dateien dieses Laufwerks angezeigt worden.

Eingebaute Kommandos

Eingebaute Kommandos wurden in diesem Kapitel schon kurz erwähnt. Im Folgenden wird jedes dieser Kommandos detailliert beschrieben. Auf Unterschiede im Gebrauch zwischen CP/M-80 Version 1.4, CP/M-80 Version 2.2 und CP/M-86 wird genau hingewiesen.

DIR – Liste das Directory (= Inhaltsverzeichnis)

Das Kommando DIR listet das Inhaltsverzeichnis der Dateien auf einer Diskette. Es können alle Dateinamen aufgelistet werden, Disketten-Directories können durch Wildcard-Zeichen auf Gruppen ähnlicher Dateinamen durchsucht werden oder der Benutzer kann eine einzelne Datei auffinden.

Frühe Versionen des CP/M-80 (1.3 und 1.4) geben die Inhaltsverzeichnisse einer einzelnen Spalte pro Zeile mit jeweils einem Dateinamen und -typ wieder, dem die Laufwerks-Spezifikation vorausgeht, z. B.:

```
A>DIR B:<cr>
B:DOCUMENT MAR
B:DOCUMENT APR
B:DOCUMENT JUN
A>
```

Die Version 2.2 von CP/M-80 und CP/M-86 listen Inhaltsverzeichnisse mit vier Dateien pro Zeile, getrennt durch eine vertikale Linie, z. B.:

```
A>DIR B:<cr>
B:DOCUMENT MAR | DOCUMENT APR | DOCUMENT JUN | DOCUMENT JUL
B:DOCUMENT AUG | DOCUMENT SEP
A>
```

Einige Versionen 2.2. des CP/M-80 listen wegen der um die Hälfte reduzierten Bildschirmbreite nur zwei Spalten je Zeile (so auch Osborne 1).

Es gibt folgende gültige Versionen des Kommandos DIR:

DIR d: <cr>
Listet das Inhaltsverzeichnis alle auf der Diskette des angegebenen Laufwerks (d:) vorhandenen Dateien. Sollte keine Laufwerks-Spezifikation angegeben sein, wird das Standardlaufwerk angenommen.

DIR d:filename.typ <cr>
Listet das Inhaltsverzeichnis aller Dateien auf Laufwerk d, die sich mit dem spezifizierten Dateinamen und -typ decken. Sie können die Wildcard-Spezifikatoren „*“ und „?“ benutzen, um Dateigruppen aufzufinden.

Es folgen Beispiele des Kommandos DIR und der resultierenden Inhaltsverzeichnisse:

```
A>DIR B:*.BAS<cr>
B:FRESHWTR.BAS
B:SALTWTR.BAS
A>
```

Dieses Beispiel zeigt ein Inhaltsverzeichnis im List-Format der Version 1.4 des CP/M-80. Das Kommando DIR B:*.BAS listet alle Dateien des Typs „BAS“ auf Laufwerk „B“.

```
A>DIR FI??.*<cr>
A:FISH  TRT | FISH  BAS | FISH      CRP | FISH      ING
A:FISH  EEE | FILE  BAS
A>
```

Dieses Beispiel zeigt ein Inhaltsverzeichnis im Format der Version 2.2. DIR FI??.* listet alle Dateinamen beliebigen Typs auf dem Standard-Laufwerk (i.A. „A:", die mit FI anfangen und zwei weitere Zeichen haben.

Fehlermeldungen

Bei Eingabe des Kommandos DIR gibt es mehrere Fehlermöglichkeiten mit entsprechenden CP/M-Kommentaren, z. B.:

NO FILE, NOT FOUND oder FILE NOT FOUND

Die Diskette enthält die angegebene(n) Datei(en) nicht. Prüfen Sie, ob das Kommando korrekt eingegeben wurde; wenn ja, ist (/sind) sie nicht vorhanden.

BDOS ERR ON d: (d: = Laufwerksbezeichnung)

CP/M konnte im angegebenen Laufwerk keine Diskette finden, die Diskette ist falsch formatiert, die Laufwerke sind ausgeschaltet oder die Tür des Laufwerks ist nicht geschlossen. Stellen Sie sicher, daß die Diskette korrekt eingelegt ist. Sollte noch die Meldung „BAD SECTOR" folgen, werden Sie nach aller Wahrscheinlichkeit eine schlechte Diskette haben oder sie ist verkehrt herum eingelegt.

DIT?

Jede Fehlermeldung mit einem Fragezeichen am Ende zeigt an, daß CP/M das eingegebene Kommando nicht auffinden konnte.

In diesem speziellen Fall wurde DIT statt DIR eingetippt. Prüfen Sie Ihre Eingabe sorgfältig auf Tippfehler, wenn Sie eine Fehlermeldung mit Fragezeichen am Ende sehen.

ERA – Lösche eine Datei (CP/M)

ERAQ – Lösche eine Datei mit Abfrage (nur MP/M)

ERA steht für „erase" (= lösche). Bei diesem Kommando müssen Sie die Namen der zu löschenden Dateien unmittelbar nach dem Befehlscode tippen. Da äußerst selten alle Dateien von einer Diskette entfernt werden sollen, werden CP/M-80 und CP/M-86 bei einer solchen Eingabe nachfragen (nicht so die Version 1.3 von CP/M-80).

Es ist eine gute Angewohnheit, vor dem Löschen das Inhaltsverzeichnis zu prüfen. Verifizieren Sie also zuerst durch DIR, daß sich die Datei(en), die Sie entfernen wollen, tatsächlich auf der Diskette befinden. Dann benutzen Sie das Kommando ERA für die Löschung. Schließlich versichern Sie sich wieder durch DIR, daß CP/M die ausgewählte(n) Datei(en) korrekt gelöscht hat. Nur so haben Sie eine ordnungsgemäße Kontrolle über den Löschvorgang.

Anwender des MP/M können ERAQ als spezielle Version des ERA benutzen, um vor jeder Dateilöschung eine Rückfrage zu erhalten. Obwohl im MP/M auch ERA verfügbar ist, sollte man bei dieser CP/M-Version ERAQ benutzen.

Im CP/M ist nur eine Form des Kommandos ERA erlaubt:

ERA d:filename.typ <cr>

Löscht alle Dateien der Diskette auf Laufwerk d:, die in Dateiname und -typ übereinstimmen. Die Wildcard-Zeichen „*" und „?" können benutzt werden, um gleichzeitig mehrere Dateien zu spezifizieren.

MP/M-Anwender können zusätzlich folgende Form benutzen:

ERAQ d:filename.typ <cr>

Löscht wie ERA, jedoch erst nach Benutzer-Bestätigung.

Hier einige Beispiele für das Kommando ERA:

```
A>DIR C:<cr>
C: QUALITY  CTL|MIND  CTL|WEIGHT  CTL|THOUGHT  CTL
A>ERA C:QUALITY.CTL<cr>
A>DIR C:<cr>
C: MIND  CTL|WEIGHT  CTL|THOUGHT  CTL
```

Im obigen Beispiel lassen wir zunächst das Inhaltsverzeichnis in Laufwerk „C" auflisten, also vier Dateien. Dann lassen wir durch CP/M die Datei QUALITY.CTL löschen. Da außer A > keine Meldung diese Aktion anzeigt, lassen wir das Inhaltsverzeichnis noch einmal auflisten, um die Löschung zu bestätigen.

```
A>DIR B:*.BAS<cr>
B: NOW  BAS|THEN  BAS|ALWAYS  BAS
A>ERA B:*.BAS<cr>
A>DIR B:*.BAS<cr>
NOT FOUND
A>
```

In diesem Beispiel finden wir auf „B" drei Dateien des Typs „BAS" und lassen sie löschen. Ein weiteres DIR bestätigt, daß auf Laufwerk „B" keine Datei dieses Typs übriggeblieben ist.

```
B>DIR <cr>
B: SOHO  NY|CHELSEA  NY|UPPREAST NY|UPPRWEST NY
B>ERA *.*<cr>
ALL FILES (Y/N)?Y <cr>
B>DIR <cr>
NO FILE
B>
```

Dieses Beispiel zeigt das Löschen aller Dateien durch CP/M-80 oder CP/M-86. Beachten Sie die Nachricht „ALL FILES (Y/N)?" und die folgende Anwender-Antwort. Wenn Sie nicht mit „Y" antworten, werden keine Dateien von der Diskette gelöscht, hier jedoch alle.

Das folgende Beispiel läßt sich nur mit MP/M anwenden:

```
AØ>ERAQ SNWAKEEN.*<cr>
A: SNWAKEEN VLY?y
A: SNWAKEEN RVR?n
A: SNWAKEEN CTY?n
AØ>DIR SNWAKEEN.*<cr>
A: SNWAKEEN RVR | SNWAKEEN CTY
AØ>
```

Hier haben wir ERAQ (Löschen mit Nachfrage) angewandt. Jede nach Name und Typ in Frage kommende Datei wird einer erneuten Überlegung anheimgestellt. Die erste Datei haben wir gelöscht, die nächsten beiden nicht. Ein DIR bestätigt unsere Aktionen.

Die bei ERA möglichen Fehlermeldungen sind im wesentlichen dieselben wie bei DIR, also:

NO FILE, NOT FOUND oder FILE NOT FOUND
CP/M oder MP/M konnte die spezifizierte Datei nicht finden. Wenn Sie z. B. ERA KNOW statt ERA NOW tippen, könnte diese Meldung erscheinen. Sollten Sie jedoch auch eine Datei namens KNOW haben, so würde diese gelöscht. Zählen Sie also nicht auf diese Fehlermeldung, um Eingabefehler aufzufangen.

BDOS ERR ON d: (d: = Laufwerks-Identifikator)
CP/M konnte keine Diskette auf dem Laufwerk finden, sie wurde falsch eingelegt, die Tür des Laufwerks ist offen oder der Strom ist nicht eingeschaltet.

ERAQ?
Das Fragezeichen zeigt wiederum an, daß CP/M das Kommando nicht finden konnte. Im vorliegenden Fall haben Sie entweder versucht, ein MP/M-Kommando für ein CP/M-Betriebssystem zu benutzen oder die Datei befindet sich nicht auf dem Standardlaufwerk.

REN – Benenne eine Datei neu

Dateien können durch das Kommando REN (Rename) neu benannt werden. Der alte sowie der neue Name der Datei muß explizit angegeben werden; Sie können nicht mit den Wildcard-Identifikatoren „*“ und „?“ Dateigruppen auf einmal neu benennen.

Fast immer, wenn der Computer wie im Rename-Kommando eine Äquivalenzanweisung verarbeitet, steht die neue Formation bzw. das Resultat links vom Gleichheitszeichen und der alte Zustand rechts. Das läßt sich so ausdrücken:

NEU = ALT

Beim Rename-Kommando kann für eine oder beide Dateinamen ein Laufwerk-Spezifikator angegeben werden. Wenn er für einen Namen angegeben wird, gilt er für beide. Wenn beiden Dateinamen Spezifikatoren zugeordnet sind, müssen sie gleich sein. Wenn keiner angegeben ist, wird die Diskette auf dem Standardlaufwerk durchsucht.

Hier eine Form des REN-Kommandos:

```
A>DIR<cr>
A: HOLDEN   ONE | PHOEBE   TWO | 19EIGHTY FOR
A>REN NOVEL.ONE = HOLDEN.ONE<cr>
A>DIR<cr>
 A: NOVEL   ONE | PHOEBE   TWO | 19EIGHTY FOR
 A>
```

In diesem Beispiel wird die Datei HOLDEN.ONE in NOVEL.ONE umbenannt.

Das Kommando REN kennt verschiedene Fehlermeldungen:

FILENAME?
Sie benutzten irrtümlich einen afn (ambiguous file name = mehrdeutiger Dateiname) in der Kommandozeile. Wenn sie z. B. REN TALL.CAR= SHORT???.CAR eingeben, wird die Fehlermeldung SHORT???.CAR? erscheinen.

NO FILE
Die angegebene Datei existiert nicht. Überprüfen Sie Ihre Eingabe.

FILE EXISTS
Der neue Name existiert bereits auf der Diskette. Dies wird von CP/M als Fehler betrachtet. Wenn Sie eine existierende Datei durch eine neue Version ersetzen wollen, benennen Sie die alte um oder löschen Sie sie. Sie können auch das in Kapitel 3 beschriebene Dienstprogramm PIP benutzen.

BDOS ERR ON d: (d: = Laufwerksbezeichnung)
CP/M konnte entweder die Diskette nicht finden oder das Laufwerk nicht aktivieren. Vergewissern Sie sich, daß die Diskette richtig eingelegt, das Laufwerk eingeschaltet und die Laufwerkstür korrekt geschlossen ist.

SAVE – Sichere den Inhalt des Speichers auf eine Diskette

Eine rauhe Methode, im CP/M-80 Speicherinhalte zu bewahren, ist das Kommando SAVE. Im CP/M-86 steht es nicht zur Verfügung.

SAVE übermittelt die Inhalte, die als „Transient Program Area" (= nichtresidenter Programmbereich) bekannt sind, an eine Diskettendatei mit frei wählbarem Namen und Typ. Man muß CP/M die Anzahl der Seiten (= Pages), also der 256-Zeichen-Blöcke, die man sichern will, sowie Namen und Typ der Datei angeben. Wählen Sie den Dateinamen sorgfältig: SAVE löscht jede vorhandene Datei gleichen Namens, bevor es eine neue kreiert.

Sie werden das Kommando SAVE kaum anwenden, wenn Sie CP/M-80 nur für ausgetestete Programme oder Standard-Software benutzen. Andererseits werden Assembler-Programmierer dieses Kommando häufig gebrauchen. Aus Kontinuitätsgründen wird SAVE an dieser Stelle des Buches vorgestellt: sollten Sie mit der Assembler-Sprache nicht vertraut sein oder weitere Hintergrundinformation wünschen, lesen Sie Kapitel 4, bevor Sie mit dieser Beschreibung fortfahren. Da in Verbindung mit SAVE normalerweise DDT, das Testhilfedienstprogramm, benutzt wird, haben wir es auch in unserem Beispiel verwandt. Auch DDT wird in Kapitel 4 – Assembler-Dienstprogramme – diskutiert.

Die erlaubte Form des Kommandos SAVE ist

SAVE ## d:filename.typ <cr>

Dabei repräsentiert ## die Anzahl der Seiten (Blöcke von je 256 Zeichen Speicherinhalt), die auf Diskette gespeichert werden sollen.

Um SAVE durchführen zu können, muß man die Anzahl der Speicherseiten kennen, die gesichert werden sollen. Die erste zu sichernde Seite beginnt stets an der hexadezimalen Adresse 0100. Die zu sichernde Anzahl von Seiten muß dezimal angegeben werden.

Anfänger stoßen oft auf Probleme in Bezug auf das Verständnis des Konzeptes der „Speicherseite" und verstehen nicht, wieso dezimal 256 äquivalent zu hexadezimal 0100 ist. Die Zahlenkonvertierung zwischen dezimal und hexadezimal ist nicht leicht für jemanden mit „Mathe-Angst".

Das hexadezimale Zahlensystem (meist „hex" abgekürzt) beruht auf der Basis 15. Das heißt, eine Stelle kann 16 Ziffern haben. Zur Basis 10 zählen wir

0 1 2 3 4 5 6 7 8 9,

zur Basis 16 (hex math)

0 1 2 3 4 5 6 7 8 9 A B C D E F.

Der Buchstabe „A" repräsentiert unsere normale Zahl 10, „B" 11 usw. Im Dezimalsystem bedeutet 111

```
1 an   1er-Stelle =   1
1 an  10er-Stelle =  10
1 an 100er-Stelle = 100
        Summe     = 111
```

Im Hexadezimalsystem bedeutet 111

```
1 an   1er-Stelle =   1
1 an  16er-Stelle =  16
1 an 256er-Stelle = 256
        Summe     = 273 (zur Basis 10).
```

Was hat das jedoch mit dem Kommando SAVE zu tun?

Sie erinnern sich, daß hex 0100 einer Speicherseite (dec 256) entspricht, hex

0200 somit zwei Speicherseiten. Da wir bei SAVE nur die Anzahl der Seiten berücksichtigen, können wir die beiden letzten Stellen vernachlässigen.

Lassen Sie uns die Anzahl der Seiten berechnen, wenn wir die Speicherinformation zwischen den Adressen hex 0100 und hex 2785 sichern wollen.

1. Vergessen Sie die erste Adresse, hex 0100; SAVE startet hier grundsätzlich.
2. Streichen Sie die letzten beiden Ziffern der höheren Adresse. In unserem Beispiel wird aus 2785 eine 27.
3. Konvertieren Sie die verbleibende Zahl in ihr dezimales Äquivalent. In unserem Beispiel ergibt also

   7 an 1er-Stelle = 7
   2 an 16er-Stelle = 32
   Summe = 39 (zur Basis 10).
4. Schritt 3 ergibt das jeweils richtige Resultat bis auf einen Spezialfall: wenn die beiden letzten Ziffern in Schritt 2 hex 00 sind, wird man in Schritt 3 eine 1 subtrahieren. Wäre beispielsweise die höhere Adresse hex 2700 anstatt hex 2785, so wäre das korrekte Ergebnis 39 – 1 = dezimal 38 Seiten für SAVE.

Nun, da Sie wissen, wie man hexadezimale Zahlen in dezimale konvertiert, lassen Sie uns ein Beispiel für den Gebrauch von SAVE betrachten.

```
A>DDT CURSEAND.HEX<cr>
DDT VERS 2.2
NEXT PC
1100 0100
-
```

In diesem Beispiel benutzen wir DDT (wird in Kapitel 4 besprochen), um eine Datei namens CURSEAND.HEX zu laden. Die Zahlen unter „NEXT" und „PC" sind wichtig: die erste zeigt auf die um 1 höhere als die letzte von der Datei genutzte Adresse, die zweite zeigt die Startadresse der Datei an. Wenden wir die obigen Regeln an, um die Anzahl der Speicherseiten zu ermitteln, die gesichert werden sollen, so kommen wir auf 16 (1 + 16 – 1). Deshalb würden wir das folgende SAVE-Kommando verwenden:

```
A>SAVE 16 CURSEAND.COM<cr>
A>
```

Fehlermeldungen

FILENAME?

Sie haben die Anzahl der zu sichernden Seiten nicht angegeben, einen Wildcard-Identifikator benutzt oder das Wort SAVE falsch geschrieben. Prüfen Sie die Genauigkeit Ihrer Eingabe bevor Sie fortfahren. Dummerweise

können Sie wegen der Art, wie manche CP/M-80-Systeme den Speicher während eines SAVE gebrauchen, nicht davon ausgehen, daß die Daten, die Sie sichern wollten, noch vorhanden sind. Kurz gesagt, Sie müssen zuerst die gewünschte Information im Speicher wiederherstellen, bevor Sie das Kommando erneut einzugeben versuchen.

NO SPACE oder DISK FULL

Auf der Diskette befinden sich bereits zu viele Dateien oder es ist kein Platz vorhanden, um die neue Information zu speichern. Löschen Sie unnötige Dateien auf der Diskette oder nehmen Sie eine andere.

TYPE – Liste eine Datei mit ASCII-Information

Wenn Sie eine Datei haben, die aus druckbaren Zeichen besteht (also Daten im Gegensatz zu Computerinstruktionen), können Sie durch CP/M mit dem Kommando TYPE ihre Inhalte auf Ihrer Konsole ausgeben.

TYPE arbeitet normalerweise mit Dateien der Typen „BAS", „ASM", „BAK", „HEX", „DOC", „TXT" bzw. überhaupt solchen, die ASCII-Text oder -Daten enthalten. ASCII ist ein Zeichenerkennungs- und Speicherschema für Computer; jedes Zeichen hat eine eindeutige Entsprechung, die gespeichert oder benutzt werden kann.

Sie können auch mit einem Text-Editor oder einem Textverarbeitungsprogramm Einblick in Dateiinhalte nehmen; aber nicht jeder CP/M-Benutzer hat diese zur Verfügung. Wenn Sie lediglich einen kleinen Ausschnitt Ihres Programms oder Ihrer Datei sehen wollen, warum sollten Sie den Text-Editor bemühen? Das Kommando TYPE ist in alle Versionen des CP/M eingebaut und immer verfügbar.

Es gibt dafür nur eine Form:

TYPE d:filename.typ <cr>

Listet die Inhalte der spezifizierten Datei.

Ein Anwendungsbeispiel wäre:

```
A>TYPE O.POS<cr>
Had Dick Seeker been present, and had he been able to see
inside the NASCOM Cray-1 computer, he would have seen a
string of bits spelling out his name, computer style. But
the National Security Computer Center was 15ØØ miles away:
in fact, Dick wasn't even aware of its existence.
```

Im obigen Beispiel forderte der Benutzer die Textauflistung der Datei O.POS. Wäre dies eine lange Datei – länger, als der Bildschirm auf einmal hätte wiedergeben können – hätte ein S benutzt werden können, um die Auflistung zeitweilig zu unterbrechen. Für den Abbruch der Textausgabe drücke man irgendeine Taste oder C.

MP/M-Benutzer können (wie im folgenden Beispiel gezeigt) eine zweistellige Zahl angeben, um TYPE anzuweisen, wieviele Zeilen jeweils auf der Konsole erscheinen sollen.

TYPE d:filename.typ P## <cr>
Die Nummernzeichen zeigen die jeweilige Zeilenanzahl an.

Wenn die MP/M-Form des Kommandos benutzt wird, listet der Computer die dem P folgende Anzahl von Zeilen und wartet dann auf CARRIAGE RETURN. Auf diese Weise kann ein MP/M-Benutzer einen Text listen, ohne befürchten zu müssen, daß er vom Bildschirm verschwindet, bevor er ausgewertet wurde (die normale Spezifikation wäre P23 für 24-Zeilen-Terminals).

Fehlermeldungen

FILENAME?
Diese Fehlermeldung erscheint, wenn die genannte Datei nicht existiert oder das Kommando falsch buchstabiert wurde. Prüfen Sie Ihre Eingabe und versuchen Sie es neu.

BDOS ERR ON d: (d: = Laufwerksbezeichnung)
CP/M konnte die Diskette oder das Laufwerk nicht finden.

Bedeutungslose Anzeige (manchmal von Klingelzeichen begleitet)
Sie haben TYPE mit einer Datei versucht, die keinen ASCII-Code oder innerhalb des Textes Maschinencode enthält. Im letzteren Fall starten Sie CP/M neu, um sicherzustellen, daß Sie die Kernspeicherinhalte Ihres Computers nicht geändert haben.

USER (= Benutzer) – Wechsel der zum gegenwärtigen Zeitpunkt eingetragenen Benutzernummer

CP/M-80 Version 2.2, CP/M-86 und MP/M enthalten das spezielle Kommando USER. Dieses Kommando erlaubt Ihnen, eine Zahl zwischen 0 und 15 zu spezifizieren, die jeder Datei, die Sie erstellen, hinzugefügt wird.

Wenn Sie einen Kaltstart mit CP/M-80 oder CP/M-86 durchführen, wird die Anwendernummer 0 angenommen; wenn Sie MP/M kaltstarten, wird die Konsolnummer als Benutzernummer angenommen. Ihre Diskettenoperationen werden sich lediglich auf Dateien des Benutzerbereichs 0 beziehen (oder eine andere Zahl, wenn Sie in MP/M eine andere Konsole benutzen). Mit anderen Worten, wenn Sie nach einem Kaltstart eine neue Datei namens JUNK.AGN sichern, wird sie immer im Inhaltsverzeichnis des Benutzerbereichs 0 erscheinen. Geben Sie vorher USER 2 <cr> ein, wird sie nur im Inhaltsverzeichnis des Benutzerbereiches 2 zu finden sein.

„Benutzerbereiche" sind imaginär. Jede Diskettendatei ist mit einer Benutzernummer assoziiert, die auf der Diskette als zusätzliche Information über die Datei gespeichert ist. Darum ist es nicht notwendig, auf der Diskette Raum für jeden Benutzerbereich freizuhalten.

Das USER-Kommando ist von minimalem Wert für die Ausführung von Standardprogrammen. Wenn sich jedoch Benutzer Laufwerke teilen, wie im MP/M, kann ein Anwender seine Dateien nach Bereich 1 und ein anderer nach Bereich 2 sichern, und beide können Dateien im Benutzerbereich 0 teilen.

Die einzig erlaubte Form des USER-Kommandos ist

USER # # <cr> (wobei # # eine Zahl zwischen 0 und 15 ist)

Wenn Sie mit mehreren Benutzerbereichen arbeiten, funktioniert auch das Kommando ERA unterschiedlich. Z. B. wird ERA *.* <cr> nur alle Dateien der derzeitig gewählten Benutzernummer löschen. Es gibt keinen Weg, alle Dateien auf der Diskette mit einem Kommando zu löschen, wenn sie nicht demselben Benutzerbereich zugehören.

Nahezu ausnahmslos wird jede Diskette, die Sie von einem Software-Hersteller erhalten, alle Dateien im Benutzerbereich 0 haben.

Fehlermeldungen

? oder #?

Sie haben keine oder eine Zahl größer als 15 spezifiziert oder USER falsch geschrieben. Prüfen Sie Ihre Eingabe und versuchen Sie neu.

FILENAME? oder NO FILE

Diese Meldungen können erscheinen, wenn Sie Benutzerbereiche wechseln und dann versuchen, auf Programme oder Daten eines anderen Bereiches zuzugreifen.

d: – Wechsle das Standard-Laufwerk

Bei einem CP/M-System mit zwei oder mehr Laufwerken wird das derzeitig aktive (Standard-) durch Tippen des Buchstabens für das gewünschte Laufwerk ausgetauscht, gefolgt von Doppelpunkt (:) und <cr>.

```
B:<cr>
C:<cr>
L:<cr>
```

Und um auf Laufwerk „A" zurückzukommen, tippen Sie

```
A:<cr>
```

Wenn Sie das Standardlaufwerk wechseln, wechselt CP/M den Anforderungsbuchstaben Auf B: <cr> werden CP/M-80 und CP/M-86 mit der Anforderung B> reagieren; MP/M würde mit #B> antworten, wobei das Nummernzeichen den Benutzerbereich repräsentiert.

Wenn Sie eine Datei auf dem Standardlaufwerk auswählen, brauchen Sie dessen Kennbuchstaben nicht mit dem Dateinamen anzugeben, aber bei allen anderen Dateireferenzen müssen Sie das gewünschte Laufwerk spezifizieren.

Fehlermeldungen

BDOS ERR ON d: SELECT

Diese Fehlermeldung wird erscheinen, wenn Sie versuchen, zu einem Laufwerk zu wechseln, das nicht existiert oder von CP/M nicht gefunden wird (vielleicht ist der Strom aus oder die Tür offen).

Zeilenausgabe-Kommandos

Mit Zeilenausgabe-Kommandos können Sie Tippfehler während der Eingabe von Kommandozeilen korrigieren und erhalten Ihnen eine gewisse Kontrolle über die Konsolanzeige (Ausgabe). Die Zeilenausgabe-Kommandos werden nach Funktion gegliedert in Tabelle 2-1 aufgeführt.

Die Kommandos CONTROL-P (^P), CONTROL-S (^S) und normalerweise auch CONTROL-H (^H) sind jederzeit anwendbar. Die anderen Zeilenausgabe-Kommandos müssen normalerweise innerhalb einer CP/M-Kommandozeile erscheinen, d. h. nach der CP/M-Anforderung und vor dem CARRIAGE RETURN.

Zeilenausgabe-Kommandos werden oft bei der Eingabe für transiente Programme benutzt. Ihre Nützlichkeit hängt dabei vom Programm ab.

^C – CONTROL-C – Neustart CP/M-80

Ein Warmstart stellt interne Information für CP/M-80 bis zu einem vordefinieren Status wieder her, ohne Programme oder Daten im Kernspeicher zu zerstören. Ein Kaltstart beginnt bei Null und zerstört vorherige Programme im Kernspeicher.

Ein Kaltstart heißt oft „cold boot", ein Warmstart „warm boot". Im CP/M-80 kennt der Warmstart zwei Hauptanwendungen:

- Disketten beim Auswechseln aus einem oder mehreren Laufwerken neu einzutragen.
- Ein laufendes transientes Programm zu unterbrechen und zur Kommando-Ebene von CP/M-80 zurückzukehren.

Sie können Probleme bekommen, wenn Sie eine Datei nicht zum rechten Augenblick abschließen, also dem CP/M-80 melden, daß das Programm die Datei nicht mehr braucht (CLOSE). Diese Probleme können Sie vermeiden, indem Sie ein Programm in jedem Fall normal beenden (wenn das Programm die Option QUIT hat, ist es besser, sie zu benutzen, als RESET oder ^C zu drücken).

Wenn Sie Disketten wechseln, ohne dies CP/M-80 oder CP/M-86 mitzuteilen, können unvorhersehbare Folgen eintreten. Drücken Sie immer CONTROL-C, wenn Sie eine andere Diskette ins Laufwerk legen, es sei denn, die Diskette wird direkt vom Laufwerk **angefordert**. Im letzteren Fall müssen Sie

annehmen, daß der Programmierer den Diskettenwechsel mit den entsprechenden Anweisungen versehen hat, um CP/M-80 davon zu informieren.

CP/M-86 bearbeitet ein CONTROL-C etwas anders als CP/M-80. Es läßt mehrere gleichzeitig arbeitende Programme im Kernspeicher zu und führt Buch über die Reihenfolge, in der sie gestartet wurde. Bei jedem ^C wird das zuletzt aufgerufene Programm gestoppt. Nach dem Stop des letzten Programms startet ^C CP/M-86 neu, ähnlich wie CP/M-80. Es ist wichtig zu wissen, daß in CP/M-86 Disketten nicht ohne Programmanforderung gewechselt werden können, es sei denn, Sie geben ^C ein, wenn gerade kein Programm läuft.

MP/M arbeitet noch anders. CONTROL-C stoppt hier nur die Ausführung des gerade laufenden Programms. Da bei MP/M mehrere Benutzer an einen Computer angeschlossen sein können, ist ein spezielles Kommando notwendig, um bestimmte Disketten in Grundstellung zu bringen, d. h. die Arbeit eines Benutzers neu zu starten, ohne die anderen in Mitleidenschaft zu ziehen. Dieses Kommando ist DSKRESET.

DSKRESET ⟨cr⟩	startet alle Disketten neu
DSKRESET d: ⟨cr⟩	startet die Diskette in Laufwerk d: neu
DSKRESET d:, d: ⟨cr⟩	startet mehrere Laufwerke neu (Laufwerksbezeichnungen werden durch Kommata getrennt)

Ein DSKRESET kann zurückgewiesen werden, wenn mehrere Benutzer auf dieselbe Diskette zugreifen. In diesem Fall wird eine Meldung anzeigen, welche Konsolen und welche Programme diese Diskette benutzen. Ein DSKRESET sollte ausgeführt werden, bevor man eine Diskette aus dem Laufwerk nimmt, andernfalls könnten die Daten anderer Benutzer zerstört werden.

Fehlermeldungen

BDOS ERR ON d: R/O (d: = Laufwerksbezeichnung)

CP/M erkennt gewöhnlich eine schreibgeschützte Diskette. Wenn das der Fall ist, setzt es für diese Diskette das Read-Only-Attribut (Nur-Lesen). Sollte dennoch ein Programm versuchen, auf diese Diskette zu schreiben, erscheint die o.a. Fehlermeldung. Der einzige Weg, diesen Fehler zu beheben ohne Disketteninformation zu zerstören, ist, ^C zu tippen (ohne daß ein Programm läuft, wenn Sie CP/M-86 benutzen).

Der erfahrene Anwender bemerkt, daß die Fehlmeldung erscheint, wenn CP/M in Disketten den „read-only"-Schalter entdeckt. Die Existenz dieser Fehlermeldung zeigt, daß man Disketten zu jeder Zeit mit Schreibschutzsperre versehen kann, solange man sie nur lesen will. Wenn Sie jedoch Ihr System in üblicher Weise benutzen, ist es selten, daß Sie auf eine Diskette nicht auch schreiben werden. Insofern ist dieser Vorschlag vielleicht nicht ganz ungefährlich.

Tabelle 2–1. Zeilenausgabe-Kommandos – nach Funktionen gruppiert

Funktion	Kommandos
Beende die Kommandozeile	CARRIAGE RETURN (<cr>) LINE FEED (<lf>) CONTROL-J CONTROL-M
Lösche die Kommandozeile (Zeilenabbruch)	CONTROL-U CONTROL-X
Radiere ein Zeichen in der Kommandozeile aus	BACKSPACE (<bs>) DELETE (<del>) RUBOUT (<rub>) CONTROL-H
Kontrolliere die Anzeige (Display)	CONTROL-E CONTROL-R CONTROL-S
Kontrolliere den Drucker	CONTROL-P
Starte CP/M-80 neu Beende die Funktion des CP/M-86	CONTROL-C

^E – CONTROL-E – Setze die Eingabe auf der nächsten Zeile fort

Um ein langes Kommando auf der nächsten Zeile der Konsole fortzusetzen, tippen Sie ^E. Dies bewegt den Cursor (= Schreibmarke) zum Anfang der nächsten Zeile. Wenn Sie dann schließlich <cr> drücken, wird das gesamte Kommando als eine Einheit interpretiert, obwohl es auf dem Bildschirm auf mehreren Zeilen erscheinen kann. Das Zeichen CONTROL-E selbst wird nicht als Teil der Kommandozeile angesehen.

^H – CONTROL-H oder BACKSPACE – Lösche das letzte Zeichen

Die Version 2.2 des CP/M-80 sowie CP/M-86 gestatten ein ^H oder BACKSPACE, um vor einem <cr> einfache Tippfehler zu korrigieren.

Wenn Ihre Tastatur „BACKSPACE" oder „BS" enthält, können Sie diese Tasten benutzen; in jedem Fall können Sie CONTROL-H eingeben. Es sind lediglich verschiedene Namen für dieselbe Funktion.

CONTROL-H arbeitet ähnlich wie RUBOUT bzw. DELETE; jedoch löscht es das ungewünschte Zeichen vom Bildschirm, während RUBOUT bzw. DELETE es auf dem Bildschirm stehenlassen und verdoppeln.

^H oder BS ist für Video-Anlage eher geeignet als für Drucker. Zwar werden CP/M-80 und CP/M-86 auch im letzteren Fall korrekt reagieren, nur der Drucker wird möglicherweise streiken.

DELETE bzw. RUBOUT – Streiche ein Zeichen und verdopple es

Die DELETE-Taste eliminiert das letzte ungestrichene Zeichen in der Kommandozeile und verdoppelt (wiederholt) es. Dazu drücken Sie je nach Tastatur „DELETE", „DEL", „RUBOUT" oder „RUB".

Den Versionen 1.3 und 1.4 des CP/M-80 fehlt normalerweise die unter „^H" beschriebene Möglichkeit des Löschens durch Zurücksetzen. Stattdessen führen diese Versionen fast immer die DELETE-Funktion mit Verdopplung des gelöschten Zeichens aus. Wenn Sie also BLA und dann DEL oder RUB eingeben, wird entweder

```
BLAA      gelöschte Zeichen werden wiederholt
```

oder

```
BL        gelöschte Zeichen sind verschwunden
```

erscheinen.

Wiederholte Zeichen mögen EDV-Neulingen etwas seltsam vorkommen. Als CP/M-80 zuerst entwickelt wurde, war die grundlegende Konsoleinheit der meisten Mikrocomputer ein Fernschreiber oder vergleichbarer Drucker. Diese können nicht zurücksetzen. Um also zu zeigen, daß ein Zeichen gelöscht wurde, ließ Digital Research es in den frühesten Versionen des CP/M-80 verdoppeln.

Die Zeiten haben sich geändert, und heutzutage benutzen die meisten Computer hochschnelle Videokonsolen. Diese können zurücksetzen und löschen. Digital Research wird diesbezüglich eine Korrekturmöglichkeit für jeden Anwender der Versionen 1.3 und 1.4 des CP/M-80 vorsehen.

^J und ^M – CONTROL-J und CONTROL-M – LINE FEED und CARRIAGE RETURN <cr> – Zeilenende

Wenn Sie die Version 2.2. des CP/M-80 oder das CP/M-86 benutzen, können Sie ^J oder die Taste LINE FEED für ein Zeilenende (CARRIAGE RETURN) einsetzen. Eine LINE FEED-Taste kann auch mit „LF" markiert sein.

Als Kommando für CP/M-80 und CP/M-86 bewirkt ^M exakt das Gleiche wie ein <cr>.

^P – CONTROL-P – Weise die Ausgabe dem Drucker zu

Um den Drucker einzubeziehen, drücken Sie ein ^P. Wenn Ihre Version des CP/M in die Konfiguration einen Drucker einbezieht, wird die gesamte Konsolanzeige auch an diesen übermittelt.

CONTROL-P arbeitet wie ein Ein-/Aus-Schalter: bei der ersten Eingabe wird der Drucker einbezogen, bei der zweiten wieder abgehängt.

Bedenken Sie, daß der Drucker den Bildschirm imitiert, wenn Sie ihn mit ^P einschalten. Wenn nicht jemand Ihrer CP/M-Version ein Drucker-Interface (= Anpaßschaltung) hinzugefügt hat, werden Sie höchstwahrscheinlich die folgenden kleinen Unbequemlichkeiten tolerieren müssen:

1. Ihr Drucker wird Seitenwechsel ignorieren und einfach Zeile für Zeile ausgeben.
2. Jeder Kontroll-Code im Text, z. B. ein Kommando „Lösche Bildschirm" (= CLEAR SCREEN), kann einen widersprüchlichen Effekt auf den Drucker haben. Viele Computer benutzen das ASCII-FORM FEED-Zeichen (Formularvorschub bwz. Seitenwechsel), um den Bildschirm zu löschen; es würde dann also bei jedem CLEAR SCREEN ein Formularvorschub auf dem Drucker vollzogen.
3. Wenn Sie nicht gerade einen Schnelldrucker haben, verlangsamt sich die Konsolausgabe. Jedesmal, wenn der Computer ein Zeichen senden möchte und den Drucker „busy" (= in Arbeit) findet, wartet er.

Einige Programme, wie z. B. WordStar, hängen den Drucker automatisch ab, ob Sie es wollen oder nicht. Sie benutzen andere Kommandos, um Information an den Drucker zu senden.

Wenn Sie Ihre Programme als fertige Standard-Pakete gekauft haben, werden diese den Drucker nach Notwendigkeit an- und aussetzen. Geben Sie bei der Ausführung eines solchen Programmes kein ^P ein, wenn das Systemhandbuch es nicht vorschreibt; wenn irgend möglich, überlassen Sie es dem Computer.

Fehlermeldungen

Wenn Sie ein ^P eingeben und die Zeichen auf dem Bildschirm doppelt auftreten, der Drucker aber nichts ausgibt, wenden Sie sich an die Firma, die Ihnen das System verkauft hat.

```
A>DIR^P<cr>
AA:: SSTTUUTTTTEERR..TTXXTT||RREEPPEEAATT..DDOOCC
AA>>
```

Verdoppelte Zeichen zeigen an, daß CP/M nicht mitgeteilt wurde, wohin Zeichen für die Listeinheit (normalerweise der Drucker) zu senden sind. Der

BIOS-Sektion des CP/M müssen Instruktionen hinzugefügt werden, um Information sauber auf dem Drucker auszugeben.

Es kann auch passieren, daß das System nach einem ^P blockiert, also die weitere Zeichenannahme verweigert. Dann ist gewöhnlich der Drucker nicht korrekt angeschlossen (prüfen Sie, ob der Stecker in der Dose ist) oder ausgeschaltet.

^R – CONTROL-R – Wiederhole die laufende Kommandozeile

Wenn Sie im Tippen nicht geübt sind und eines der frühen CP/M-Systeme haben, das gelöschte Zeichen verdoppelt, kann es sein, daß ein Kommando

```
DIR B:BASIC???.*
```

mehr aussieht wie

```
DIBBR BAA:BASIXXC?????.*
```

Wenn Sie solche Kommandozeilen häufig lesen müssen, beginnen Sie vielleicht, Ihren Computer zu hassen. Hier hilft CONTROL-R. Sie geben es ein, und es erscheint eine neue korrekte Kommandozeile unter dem getippten Original. Diese neue Zeile löscht alle Zeichen, die verdoppelt anstatt gelöscht waren, z. B.:

```
DIBBR BAA:BASIXXC?????.*^R
DIR B:BASIC???.*
```

^R arbeitet nicht mit Version 1.3 des CP/M-80; Anwender der Version 2.2 oder des CP/M-86 werden es wahrscheinlich kaum brauchen.

^S – CONTROL-S – Unterbreche die Anzeige

CP/M-80 und CP/M-86 können die Konsolanzeige **unterbrechen**. Alles stoppt wie ein Standbild in einem Film, bis Sie befehlen, weiterzuarbeiten. Sie lassen durch ein ^S den Bildschirm pausieren. Mit einem weiteren ^S oder irgendeinem Zeichen außer ^C wird die Arbeit fortgesetzt. Tatsächlich stoppt CONTROL-S den Computer und daher auch die Anzeige.

Trotzdem ist dies nicht die beste Möglichkeit, einem Überlauf auf dem Bildschirm zu begegnen. Meist wird man durch den Überlauf überrascht und reagiert zu langsam, um das Bild exakt an der gewünschten Stelle anzuhalten. „Gute" Programme arbeiten mit formatisierter Bildschirmausgabe und versuchen, niemals mehr Information auf dem Bildschirm unterzubringen, als dieser verkraften kann. „Gute" Programme schließen auch einen automatischen Stop am Ende eines Bildschirms voll Information ein und erwarten ein CARRIAGE RETURN (oder eine andere Taste) zur Fortsetzung der Ausgabe. Kurz, wenn Sie während einer Programmausführung ^S drücken müssen, sehen Sie sich nach besseren Programmen um oder bitten Sie Ihren Händler, Ihr Programm zu modifizieren.

^U/^X – CONTROL-U/CONTROL-X – Brich das laufende Kommando ab

Sollten Sie einmal versehentlich Kauderwelsch tippen, werden Sie nicht nervös, es passiert uns allen gelegentlich. Vielleicht war Ihre Eingabe:

```
DIRG A:THOMKJLSK.SAB = SLIUF
```

Das ergibt überhaupt keinen Sinn. Anstelle eines BACKSPACE oder DELETE mag es einfacher sein, die Eingabe durch ^U neu zu starten. CP/M ignoriert das bis dahin Eingegebene und geht zum Beginn der nächsten Zeile über. Ans Ende der gelöschten Zeile setzt es ein Nummernzeichen (#).

In der Version 1.3 des CP/M-80 benutzen Sie ^U, bei allen anderen ^X, um die abgebrochene Zeile zu löschen, oder ^S, um sie auf dem Bildschirm zu erhalten.

Kapitel 3
CP/M – Transiente Kommandos

Im letzten Kapitel erläuterten wir die eingebauten Kommandos des CP/M. CP/M-80 und CP/M-86 ermöglichen es außerdem, diese Basiskommandos durch eine Reihe zusätzlicher Programme zu erweitern, die sich wie Kommandos verhalten. Dieses Kapitel beschreibt einige dieser Programme, insbesondere diejenigen, die normalerweise von CP/M-80 und CP/M-86 unterstützt werden.

Was ist ein Programm?

Sie wissen aus Kapitel 2, daß eingebaute Kommandos die Worte sind, die CP/M mit den ständig im Kernspeicher befindlichen Instruktionen interpretieren kann, während transiente, also nicht fest eingebaute Kommandos (normalerweise Programme genannt) durch das Tippen eines Ausdrucks aufgerufen werden, wie ein Kommando wirkt und CP/M instruiert, sich die weiteren Befehle von der Diskette zu holen.

Es fragt sich also, woher CP/M weiß, welche weiteren Instruktionen auf der Diskette verfügbar sind.

Wenn sie auf Anforderung des CP/M-80 etwas tippen, das nicht als Kommando erkannt wird, wird das Disketten-Inhaltsverzeichnis auf eine Datei des Typs „.COM" durchsucht, deren Name mit dem ersten getippten Wort übereinstimmt. Nehmen wir an, Sie tippen

BUY <cr>

Nach der Feststellung, daß BUY kein eingebautes Kommando ist, sucht CP/M-80 nach einer Datei BUY.COM auf dem z. Zt. aktiven Laufwerk. Ist sie dort nicht vorhanden, gibt CP/M-80 die Fehlermeldung BUY? aus. Wenn die Datei jedoch existiert, lädt CP/M-80 ihre Inhalte in den Kernspeicher und übergibt die Kontrolle an ihre erste Instruktion.

CP/M-86 arbeitet fast genauso wie CP/M-80 – der einzige Unterschied ist, daß es den Dateityp „.CMD" benutzt. Im obigen Beispiel würde CP/M-86 also die Datei BUY.CMD suchen. Es gibt einen guten Grund für die in CP/M-80 und CP/M-86 unterschiedlichen Dateitypen. Da beide CP/M-Betriebssysteme die gleiche Speichermethode für Disketten benutzen, ist es möglich und in einigen Fällen ratsam, beide – CP/M-80- und CP/M-86-Programme – auf dieselbe Diskette zu speichern. Da der 8086-Chip, für das CP/M-86 geschrieben wurde, keine Programme für die 8080-Familie, die von CP/M-80 unterstützt wird, verarbeiten kann, modifizierte Digital Research beim Entwurf des CP/M-86 den Typ der Kommandodateien, um Konfusion zu vermeiden.

Sie werden sofort den Wert der Einrichtung transienter Kommandos im CP/M erkennen. Sollte die mit dem „Kommando" aufgerufene Datei Com-

puterinstruktionen enthalten, sieht es so aus, als ob Sie ein gültiges Kommando getippt hätten, obwohl die Ausführung wegen des notwendigen Diskettenzugriffs natürlich etwas länger auf sich warten läßt.

Was ist also ein Programm, das wie ein Kommando arbeitet? Zum gegenwärtigen Zeitpunkt wollen wir es im Kontext zu CP/M wie folgt definieren: Ein Programm ist eine Folge von Instruktionen in einer Diskettendatei des Typs „.COM" für CP/M-80 bzw. „.CMD" für CP/M-86. Es wird durch Tippen des Dateinamens aufgerufen. So ein Programm wird häufig **transientes Programm** oder **transientes Kommando** genannt.

Das bedeutet, daß, nachdem ein Programm einmal von der Diskette in den Kernspeicher des Computers geladen wurde, dieser – abgesehen von Ausnahmezuständen – zeitweilig die Instruktionen des Programmes ausführt, ohne auf CP/M zurückzugreifen. Ein hypothetisches Programm „DISPLAY.COM" (oder „—.CMD") könnte den Computer anweisen, die folgenden Schritte zu unternehmen:

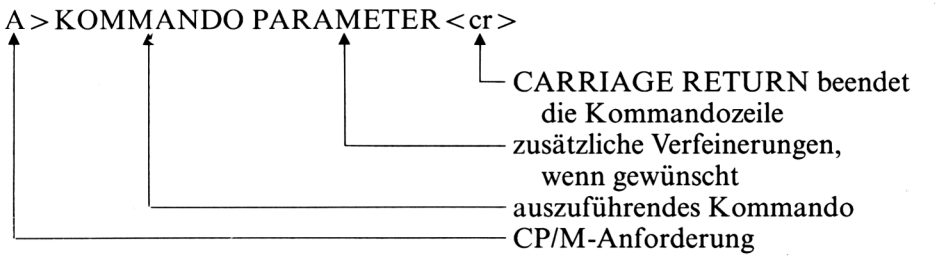
1. Frage nach dem Namen der auszugebenden Datei
2. Suche die Datei auf der Diskette
3. Lies ein Zeichen aus der Datei
4. Gib das Zeichen auf der Konsole aus
5. Wenn noch Zeichen übrig sind, gehe zurück nach Schritt 3
6. Übergib die Steuerung an CP/M

Aufruf eines Programms

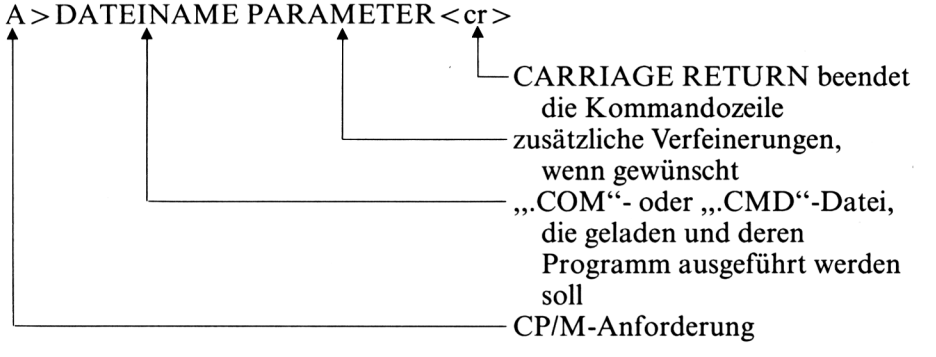
Die Kommandozeile bis CARRIAGE RETURN ist alles, was sie nach der CP/M-Anforderung (A>) tippen. CP/M erlaubt Kommandozeilen mit der maximalen Länge von 127 Bytes; Sie können also bis zu 127 Zeichen für ein einzelnes Kommando eingeben. Verblüfft es Sie, daß CP/M über den Dateinamen, der im allgemeinen acht Stellen lang ist, 119 Zeichen mehr akzeptiert?

So wie die Arbeitsweise einiger Kommandos durch das Hinzufügen erweiterter Angaben verfeinert werden kann, so können auch Programme mit zusätzlicher Information aufgerufen werden. So verfeinert z. B. DIR B:*.BAS gegenüber DIR *.BAS das Kommando DIR. Da es an einem besseren Namen oder auch nur einem eingeführten Standard mangelt, wollen wir diese Angaben „zusätzliche Informationsparameter" nennen. Pro Kommandozeile sind mehr als ein Parameter möglich.

Lassen Sie uns kurz wiederholen. Eine Generalisierung des Formates einer Kommandozeile sieht folgendermaßen aus:



Genauso können wir ein transientes Kommando bzw. Programm generalisieren:



Was sind nun die Parameter, die wir dem Programm mitgeben? Die Antwort darauf ist nicht einfach: einige Programme akzeptieren keine zusätzlichen Parameter, andere haben die Möglichkeit, wieder andere erfordern sie. Die Anzahl zu übergebender Parameter kann zwischen einem und so vielen liegen, wie die Kommandozeile zu fassen vermag.

Dieses Kapitel beschreibt eine Auswahl transienter CP/M-Kommandos. „Transient" bedeutet, daß sich die Kommandos (es sind tatsächlich Programme) nicht permanent im Kernspeicher befinden, sondern auf Diskette geführt werden. Wir benutzen den Begriff „transientes Kommando" in Übereinstimmung mit der Beschreibung dieser Programme für CP/M-80 und CP/M-86 von Digital Research.

Bevor wir weitergehen, müssen wir einen Begriff neu definieren, den wir im ganzen ersten Teil des Buches benutzt haben.

Es besteht der Trend, schnellere Laufwerke mit Platten größerer Kapazität in Mikrocomputer-Systeme einzubeziehen. Diese neueren Laufwerke arbeiten mit dicken starren Metallplatten, die nicht die dünne flexible Plastikoberfläche der Diskette haben. Sie werden Hartplatten, starre Platten, Winchester-Platten (ein Spitzname, den sie nach Einführung der 3030-Hartplatte durch IBM erhielten) oder einfach Platten genannt. Bis jetzt haben wir in diesem Buch fast ausschließlich von „Disketten" gesprochen. Da jedoch die

Version 2.2. des CP/M-80 sowie CP/M-86 entworfen wurden, um Hartplatten einzubeziehen, werden wir unsere Terminologie leicht ändern. Über den ganzen Rest dieses Buches wird häufig der Begriff „Platte" benutzt werden, und damit sind sowohl Floppy-Disketten als auch Hartplatten gemeint. Frühere Hinweise über Disketten in diesem Buch sind generell auch für Hartplatten anwendbar.

Nun haben Sie die nötigen Informationen um fortzufahren. Während wir die Liste der eingebauten Kommandos erschöpfend behandelt haben, haben wir auf die transienten Kommandos lediglich hingewiesen. Im Rest dieses Kapitels werden wir auf einige wichtige transiente Kommandos eingehen, die Digital Research für CP/M-80 und CP/M-86 entwickelt hat. Kapitel 4 befaßt sich mit den übrigen Programmen von Digital Research, die für Sie wichtig sein könnten.

Format-Beschreibungen
Transienter Programme

Wir haben in Kapitel 2 ein strukturiertes Format benutzt, um jedes eingeführte Kommando zu beschreiben. In diesem Kommando wollen wir es folgendermaßen tun:

- *Programm-Name*
  Kurzbeschreibung der Programmfunktion, gefolgt von der Anwendung.
- *Kommandozeilen-Beschreibung*
  Format . . . . . Beschreibung
  .
  .
  .
  Format . . . . . Beschreibung
- *Zweck der Kommandozeile*
- *Anwendungsbeispiele mit Computer-Anzeige*
- *Andere Eingaben als die Kommandozeile*

Anmerkung: Wir haben Beispiele aus einer Vielzahl von CP/M-Ausführungen gewählt, obwohl wir die Auswahl an normaler Anwendung orientiert haben. Möglicherweise wird Ihre Ausführung des CP/M Meldungen oder Information nicht ganz genauso anzeigen wie hier, aber das sollte Sie nicht irritieren. Wenn jedoch die Information, die Sie erhalten, von dieser Interpretation so radikal verschieden zu sein scheint, daß Sie sie nicht in Einklang bringen können, oder wenn Sie anscheinend unsinnige oder nicht zu den Beispielen passende Meldungen empfangen, sprechen Sie mit Ihrem Händler, um die Abweichungen zu klären, bevor Sie fortfahren.

Am Ende dieses Kapitels werden Sie auch einen Abschnitt mit den meisten CP/M-Fehlermeldungen finden, die Sie bei der Erprobung der in diesem Kapitel eingeführten Kommandos erhalten können.

Haushaltungs-Dienstprogramme

Für Computer-Benutzer bedeutet **Haushaltung** die Pflege der Geräte, innerhalb derer ein Programm mit seinen Daten operiert. Wohlgepflegte Arbeitsmittel sichern vernünftige Speicherung und Wiedererlangung von Information; sie wird so angeordnet, daß sie leicht wiedergefunden werden kann. Betrachten wir einen Ablageordner. Wir organisieren darin Daten für einen leichten Zugriff. Aber was geschieht, wenn er voll ist? Wenn wir in Eile sind – werden wir trotzdem alles in angemessener Ordnung halten? Wie finden wir einen Posten wieder, der nur zeitweilig auf unserem Schreibtisch lag? Die Antwort auf diese Fragen schließt jede Haushaltungsroutine zur Aufrechterhaltung der Vollständigkeit der im Ordner enthaltenen Information ein.

Betrachten wir einmal eine Platte als elektronischen Datenordner. Wie jeder Ordner kann sie voll werden, in Unordnung geraten oder Information vermissen lassen.

Selbstverständlich wird einige Haushaltungsroutine benötigt, um diese elektronische Ablage, unsere Platten, in Ordnung zu halten. Sie muß folgende Funktion erfüllen:

- den derzeitigen Status einer Platte aufzuzeigen
- die Information auf der Platte neu zu organisieren
- Dateien zu erweitern

Aber wir haben ja bereits einige in Kapitel 2 aufgeführte Kommandos für diese Funktionen, die sich nach der folgenden Tabelle kategorisieren lassen:

Kommando	Gebrauch		
	Status	Neuordnung	Erweiterung
DIR	x		
ERA		x	
REN		x	
SAVE		x	x
TYPE	x		
USER		x	

Dennoch genügen diese Kommandos nicht, um eine Platte zu ändern oder eine fundamentale Neuordnung zu schaffen. Deshalb erstellte Digital Research eine Anzahl von Haushaltungs-Dienstprogrammen, um Disketten in Ordnung zu halten. Darin sind folgende transiente Programme eingeschlossen:

Programm	CP/M-80	CP/M-86
STAT	STAT.COM	STAT.CMD
PIP	PIP.COM	PIP.CMD.
ED	ED.COM	ED.CMD
DUMP	DUMP.COM	DUMP.CMD
SYSGEN	SYSGEN.COM	LDCOPY.CMD
MOVCPM	MOVCPM.COM	n/a

Wie die eingebauten können auch die transienten Haushaltungs-Kommandos nach Funktionen kategorisiert werden:

Kommando	Gebrauch		
	Status	Neuordnung	Erweiterung
STAT	x	x	x
PIP		x	x
ED		x	x
DUMP	x		
SYSGEN		x	x
LDCOPY		x	x
MOVCPM		x	x

SYSGEN, LDCOPY und MOVCPM werden in Kapitel 5, STAT, PIP, ED und DUMP in diesem Kapitel beschrieben.

Die Beschreibung von STAT ist wegen der zwei unterschiedlichen Anwendungen dieses Programmes in zwei Abschnitte unterteilt.

Der Abschnitt „Dateien-Statistik" beschreibt den Dateien-, der Abschnitt „Geräte-Statistik" den Geräte-Haushalt. Geräte-Einheiten wurden in Kapitel 1 aufgeführt.

Die Berichterstattung über PIP ist in drei Abschnitte unterteilt. Diese entsprechen den natürlichen Unterschieden in Bezug auf Ursprünge und Bestimmung der Daten, die durch PIP verbunden werden können.

STAT – Dateien-Statistik

STAT, Abkürzung für STATISTICS (= Statistik), vermittelt Informationen über eine Datei oder eine Dateiengruppe. Die Beschreibung von STAT für andere technische Einheiten als Plattenlaufwerke werden im Folgenden beschrieben.

STAT-Kommandos übermitteln entweder Information über den Umfang oder Aufbau einer oder mehrerer Dateien auf einer Platte, oder sie ändern

deren Merkmale. Die Dateien werden in den Parametern der Kommandozeile spezifiziert. Die **Dateigröße** bezieht sich auf die Anzahl der für sie reservierten Bytes. **Ungenutzter Speicherplatz** bezeichnet die Anzahl von Bytes, die auf einer Platte noch zur Speicherung von Dateien zur Verfügung stehen. **Datei-** oder **Plattenmerkmale** sind eine bestimmte Folge von Kennzeichen, die sich einer Datei oder Diskette zuordnen lassen.

Vorstellung von STAT und seiner Terminologie

Der Umfang einer Datei oder der ungenutzte Speicherplatz einer Platte wird von STAT in Bytes angegeben. Wie wir in Kapitel 1 besprochen haben, ist ein Byte die Speichereinheit für ein einzelnes Zeichen, und ein KB (Kilobyte) entspricht $1024 = 2^{10}$Bytes.

Wenn STAT meldet, daß auf einer Platte 34 K übrig sind, können wir noch 34 816 (1024 × 34) Bytes speichern.

Erfahrung lehrt am besten, die Dateiinformationen von STAT zu interpretieren. Wenn Sie STAT oft gebrauchen und die Listen vergleichen, werden Sie ein Gespür für Speichergrößen entwickeln. Z. B. entsprechen 34 K ungefähr 10 Schreibmaschinenseiten (angenommen 65 Zeichen pro Zeile und 54 Zeilen pro Seite).

Benutzer des CP/M-80 Version 2.2 oder des CP/M-86 können Dateien zwei Paare von Attributen zuordnen: R/O oder R/W und DIR oder SYS.

R/O Read-only- (schreibgeschützte) Datei.
Sie können eine R/O-Datei weder verändern noch durch ERA löschen. Der R/O-Status schützt vor irrtümlichen Löschungen wertvoller Dateien.

R/W Read-Write- (Lies/schreib-) Datei.
Eine R/W-Datei können Sie ändern oder löschen – es sei denn, die ganze Datei ist schreibgeschützt. Das ist das normale und damit das Standardattribut einer Datei (es muß nicht explizit angegeben werden).

SYS System-Datei.
Eine SYS-Datei erscheint nicht in der Anzeige eines Platteninhaltsverzeichnisses. Von einer SYS-Datei können Sie Informationen wiedererlangen, neue speichern, sie löschen oder in jeder üblichen Weise gebrauchen; im Inhaltsverzeichnis wird sie aber nicht erscheinen, nur durch STAT. In der Version 2.2. des CP/M-80 sind SYS-Dateien für alle Benutzerbereiche verfügbar, in späteren Versionen dieses Systems nur für Benutzerbereich 0.

DIR Directory- (Inhaltsverzeichnis-) Datei.
Eine DIR-Datei erscheint in allen Anzeigen des Inhaltsverzeichnisses der gegenwärtigen Benutzerbereiche. Es ist das normale Dateiattribut.

Die Partner eines jeden Attributenpaares schließen sich gegenseitig aus; weder können Dateien das R/W- und R/O- noch das SYS- und DIR-Attribut besitzen.

Wenn Sie versuchen, auf eine R/O-Platte oder -Datei zu schreiben, wird die Fehlermeldung BDOS ERR ON d:R/O erscheinen (d: = Laufwerksbezeichnung). Selbst wenn eine Platte nicht schreibgeschützt ist, kann diese Meldung durch STAT bei einem gesetzten R/O-Schalter vorkommen. Das bedeutet, daß Sie Disketten ausgetauscht haben, ohne es CP/M mitzuteilen, aber CP/M den Tausch erkannt hat. Um den R/W-Status der neuen Diskette zu bestätigen, tippen Sie einfach CONTROL-C.

Wenn eine Diskette durch eine Schreibschutz-Raste gesperrt ist, erscheint sie in der STAT-Liste trotzdem als R/W, denn CP/M hat keine Möglichkeit, den Status der Schreibschutz-Raste festzustellen, bevor Sie einen Schreibversuch unternommen haben.

Die Information von STAT über Dateiumfang, -attribute und Speicherplatz werden für folgende Funktionen benötigt:

- zu prüfen, wieviel Platz auf einer Diskette übrig ist
- zu prüfen, wieviel Platz von einer Dateiengruppe belegt ist
- zu prüfen, wieviel Platz von einer einzelnen Datei belegt ist
- einer Dateiengruppe R/O- bzw. R/W-Attribute zuzuordnen
- einer einzelnen Datei R/O- bzw. R/W-Attribute zuzuordnen
- einer Dateiengruppe die Attribute DIR oder SYS zuzuordnen
- einer einzelnen Datei die Attribute DIR oder SYS zuzuordnen.

Um zu erkennen, welche STAT-Kommandozeile für jeden der aufgelisteten Zwecke einzugeben ist, lesen Sie die nächsten Seiten sorgfältig. Testen Sie die Beispiele an Ihrem Computer und versuchen Sie auch, die Fehlermeldungen hervorzurufen. Überlegen Sie anhand der gerade aufgelisteten Gebrauchsmöglichkeiten von STAT, was Sie mit der angezeigten Information anfangen können.

Gebrauch von STAT für Dateien

STAT <cr>

Zeige das Attribut und den freien Speicherplatz auf den seit dem letzten Kalt- oder Warmstart im Zugriff befindlichen Platten an.

Beispiele:

```
A>STAT<cr>                    Kommandozeile
A: R/W, SPACE: 2K             Meldung von STAT

A>STAT<cr>                    Kommandozeile
A: R/W, SPACE: 2K             Meldungen von STAT
B: R/O, SPACE 12ØK
A>
```

(Die Version 1.3 des CP/M-80 meldet nur den Status des z. Zt. eingetragenen Laufwerks.) Beachten Sie, daß in den obigen Beispielen zuerst der Schreibschutz-Status gemeldet wird, danach der freie Speicherplatz.

Spezielle Eingaben:
Keine

STAT d: <cr>
Melde den freien Speicherplatz der Platte auf Laufwerk d:

Beispiel:

```
A>STAT B:<cr>                    Kommandozeile
BYTES REMAINING ON               Meldung von STAT
B: 170K
A>
```

Hier fragen Sie nach der Statistik des zweiten Laufwerks, des Laufwerks „B:".

Spezielle Eingaben:
Keine

STAT d:filename.typ <cr>

Zeige den von der angegebenen Datei (oder mehreren) auf Laufwerk d: besetzten Speicherplatz an. Dateinamen und -Erweiterungen können die mehrdeutigen Zeichen (* und ?) enthalten. Die Angabe des Laufwerkskennzeichens ist wahlfrei; wenn sie unterlassen ist, wird das derzeitig aktive Laufwerk angenommen.

Beispiele:

```
A>STAT JOAN.ARC<cr>

 RECS  BYTS  EX  ACC    D:FILENAME.TYP
    5   2K    1  R/W    A:JOAN.ARC

A>STAT B:JOAN.ARC<cr>

 RECS  BYTS  EX  ACC    D:FILENAME.TYP
   10   4K    1  R/W    B:JOAN.ARC

A>STAT *.COM<cr>

 RECS  BYTS  EX  ACC    D:FILENAME.TYP
    4   2K    1  R/W    A:DUMP.COM
   48   6K    1  R/W    A:(ED.COM)
   56   8K    1  R/O    A:PIP.COM
   24   4K    1  R/W    A:STAT.COM
   10   2K    1  R/W    A:SUBMIT.COM
BYTES REMAINING ON A: 218K
```

```
A>STAT B:EXAMPLE.?X?<cr>

 RECS  BYTS   EX   ACC     D:FILENAME.TYP
   48    6K    1   R/W     B:EXAMPLE.TXT
   24    4K    1   R/W     B:EXAMPLE/EXT
A>
```

Spezielle Eingaben:
Keine

In den letzten Beispielen wurden einige neue Begriffe in den von STAT gedruckten Kopfzeilen angeführt.

RECS
Die Anzahl der von der Datei belegten Sektoren. CP/M speichert Information in Einheiten von 128 Bytes.

BYTES
Die Länge der Datei in Kilobytes (1 K = 1024 = 2^{10}Bytes).

EX
Die Anzahl physischer „Extents" (= Erweiterungen bzw. Abschnitte) der Datei. Extents sind eine weitere Möglichkeit für CP/M, eine Datei zu führen, aber für die meisten Benutzer ist dies unerheblich.

ACC
Das Zugriffsattribut der Datei, R/W oder R/O. Die Version 1.4 und frühere Versionen des CP/M-80 schließen diese Funktion nicht ein.

D:FILENAME.TYP
Die Spalten-Überschrift für Laufwerk und Dateinamen auf der Anzeige. Sie entfällt ab Version 2.2 des CP/M-80.

A: (ED.COM)
Ein eingeklammerter Dateinname zeigt an, daß das Attribut SYS gesetzt ist. Es ist nur für 2.2 und neuere Versionen des CP/M-80 verfügbar.

STAT d:filename.typ $atr <cr>

Sie können das Attribut „atr" (R/O, R/W, DIR oder SYS) mit einem vorangehenden Dollarzeichen ($) angeben. Dateiname und -typ können die Wildcard-Spezifikatoren „*" und „?" enthalten. Das d: ist wahlfrei; wenn nicht angegeben, wird das derzeitige Standardlaufwerk angenommen.

Beispiele:

```
A>STAT BROTHER.AND $R/O<cr>
BROTHER.AND SET TO R/O
A>

A>STAT B:BROTHER.AND $SYS<cr>
B:BROTHER.AND SET TO SYS
A>
```

```
A>STAT *.* $R/W<cr>
BROTHER.AND SET TO R/W
SISTER.TOO SET TO R/W
A>
```

Spezielle Eingaben:
Keine

STAT – Einheitenstatistik

STAT gibt ebenfalls Auskunft über die physischen und logischen Einheiten von CP/M. In Kapitel 1 führten wir kurz die physischen Einheiten als die Geräte ein, aus denen Sie Ihr Mikrocomputer-System zusammensetzen können. Der Begriff „logische Einheit" betrifft die generellen Funktionen Ihres Mikrocomputers, während „physische Einheit" ein spezifisches Gerät bezeichnet, das eine solche Funktion ausführt. CP/M erfordert die Auswahl einer physischen Einheit für die Funktion jeder logischen Einheit. Wir übermitteln dem Computer unsere Auswahl durch das STAT-Kommando (für Einheiten).

Es gibt in CP/M vier logische Einheiten:

CON:
Gib Kommandos ein und zeige Information an; Funktion der Operatorkonsole

RDR:
Empfange Information; Funktion des Lochstreifenlesers

PUN:
Sende Information; Funktion des Lochstreifenstanzers

LST:
Liste (drucke) Information; List-Funktion.

Es sind 12 physische Einheiten möglich:

TTY:
langsame Konsolanzeige (Fernschreiber)

CRT:
schnelle Konsolanzeige (cathode ray tube = Datensichtgerät)

BAT:
Einheit für Stapelverarbeitung

UC1:
Vom Benutzer definierte Konsole

PTR:
Lochstreifenleser

PTP:
Lochstreifenstanzer

UR1:
Leser #1

UR2:
Leser #2

UP1:
Stanzer #1

UP2:
Stanzer #2

LPT:
Zeilendrucker

UL1:
Listeinheit

Anmerkung: Der Doppelpunkt (:) ist Bestandteil der Namen logischer wie physischer Einheiten und muß jeweils angegeben werden.

Sechzehn Zuordnungen physischer zu logischen Einheiten sind möglich.

Logisch	**Physisch**
CON: kann übernommen werden von	TTY:, CRT: BAT:, UC1:
RDR: kann übernommen werden von	TTY:, PTR:, UR1:, UR2:
PUN: kann übernommen werden von	TTY: PTP:, UP1:, UP2:
LST: kann übernommen werden von	TTY:, CRT:, LPT:, UL1:

Ihr Mikrocomputer könnte für andere physische Einheiten als den standardmäßigen TTY: generiert sein. Dies wurde höchstwahrscheinlich von Ihrem CP/M-Händler durchgeführt. Wenn Sie zwei Drucker haben, könnte einer als LPT:- der andere als UL1:-Einheit generiert sein. Einmal getroffen sind solche Zuordnungen generell konstant; sie reflektieren die jeweils notwendige Programmierung, um jede Einheit mit dem Mikrocomputer zu verbinden. In unserem Beispiel können die beiden physischen Einheiten LPT: und UL1: die Funktion der logischen Einheit LST: übernehmen.

Unglücklicherweise sind die meisten Computerhersteller in Bezug auf die Zuordnung physischer Einheiten nicht den Empfehlungen von Digital Research gefolgt und Digital Research hat standhaft die ursprünglichen Einheitennamen aufrechterhalten, trotz wesentlicher Wandlungen in der Technologie von Terminals, Modems und Druckern. Hier ist der normale und terminologisch verwirrende Aufbau eines typischen Mikrocomputers:

Logisch	**Physisch**	**Tatsächliche Einheit**
CON:	TTY:	hochgeschwindes Videoterminal
RDR:	PTR:	Modem-Empfangskanal (Übermittlungseinheit)
PUN:	PTP:	Modem-Sendekanal (Tel-TV-Video)
LST:	LPT:	Drucker

Sie sehen also, daß die Information dieser Tabelle sich nicht mehr mit den vorherigen Beschreibungen deckt (z. B. bezeichnete TTY: ursprünglich eine langsame Konsoleinheit, nicht ein hochschnelles Videoterminal). Lochstreifenleser und -stanzer werden kaum noch gebraucht, und die Abkürzungen PUN:, RDR:, PTR: und PTP: beziehen sich eigentlich auf vergangene Zeiten. Lassen Sie sich also nicht stören, wenn Ihre Gerätezuordnung nicht den CP/M-Definitionen entspricht.

Beim ersten Start Ihres Computers wird die Standard-Gerätezuordnung vorgenommen. Sie ist von Ihrem Software-Händler generiert und variiert von System zu System. Diese Standardwerte sollten den Zuordnungen entsprechen, die Sie am häufigsten benutzen. Ein Kaltstart setzt die Gerätezuordnungen auf Standard, ein Warmstart berührt sie in keiner Weise.

Wir benutzen STAT für Einheiten

- um die derzeitige Gerätezuordnung zu erfahren
- um die möglichen Gerätezuordnungen zu erfahren
- um logischen Einheiten physische zuzuordnen.

Einige STAT-Kommandozeilen für Geräte fordern Information auf Plattenlaufwerken an. Diese speziellen Kommandozeilen benutzen wir

- um den gegenwärtigen Status eines Laufwerks zu erfahren
- um den gegenwärtigen Status der Benutzerbereiche auf einer Platte zu erfahren
- um eine Platte vor irrtümlichen Schreibbefehlen zu schützen.

Um zu entscheiden, welche Form der STAT-Kommandozeile Sie für welchen Zweck wählen sollten, lesen Sie die folgenden Seiten sorgfältig. Überprüfen Sie jede Kommandozeile, ihren Zweck und die resultierende Computer-Anzeige. Bedenken Sie, was Sie mit der Information der genannten STAT-Variationen anfangen können.

STAT DEV: <cr>

Zeige die laufende Gerätezuordnung an.

Beispiel

```
A>STAT DEV:<cr>
CON: IS CRT:
RDR: IS UR1:
PUN: IS UP2:
LST: IS UL1:
A>
```

Beachten Sie, daß die linke Spalte die logische, die rechte Spalte die korrespondierende physische Einheit bezeichnet.

Spezielle Eingaben:
Keine

STAT VAL: <cr>

Zeige die möglichen Zuordnungen physischer zu logischen Einheiten und einen gekürzten Überblick über STAT-Kommandoteile an.

Beispiel

```
A>STAT VAL:<cr>
   TEMP R/O DISK: D: = R/O
   SET INDICATOR: D:FILENAME.TYP $R/O $R/W $SYS $DIR
   DISK STATUS  : DSK: D:DSK:
   USER STATUS  : USR:
IOBYTE ASSIGN:
   CON:  = TTY:CRT:BAT:UC1:
   RDR:  = TTY:PTR:UR1:UR2:
   PUN:  = TTY:PTP:UP1:UP2:
   LST:  = TTY:CRT:LPT:UL1:
```

Anmerkung: Wenn Sie die Versionen 1.4 oder 1.3 des CP/M-80 benutzen, erscheinen nur die letzten vier Zeilen.

Die unterschiedlichen ersten Zeilen, die STAT präsentiert, geben die möglichen Einheiten von STAT-Kommandos an. Es sind

STAT D:=R/O	um einen temporären Schreibschutz für eine Platte zu bewirken
STAT D:FILENAME.TYP R $ATR	um ein Attribut zu setzen
STAT D:DSK:	um eine Laufwerksstatistik zu sehen
STAT D:USR:	um die Statistik über den Gebrauch eines Benutzerbereichs zu sehen

Spezielle Eingaben:
Keine

STAT log: = phy: <cr>

Ordne die spezifizierte physische der spezifizierten logischen Einheit zu.

Beispiel

```
A>STAT CON:=CRT:,LST:UL1:<cr>
A>
```

Beachten Sie, daß mit einem Kommando mehrere Gerätezuordnungen getroffen werden können, wobei jede durch ein Komma abgetrennt wird.

Beachten Sie außerdem, daß CP/M keine Meldung ausgibt, um Sie zu informieren, daß die Zuordnung vorgenommen wurde. Wenn Ihr System nach einer neuen Gerätezuordnung nicht länger arbeitet, ist es möglich, daß Sie für die Konsolfunktion eine nicht vorhandene Einheit angegeben haben.

Spezielle Eingaben:
Keine

STAT USR:<cr>

Zeige die laufende Benutzernummer an und liste alle Benutzernummern, für die Dateien vorhanden sind.

Beispiel

```
A>STAT USR:<cr>
ACTIVE USER: Ø
ACTIVE FILES: Ø 1 3
A>
```

Die obige Meldung zeigt, daß Sie z. Zt. im Benutzerbereich 0 arbeiten, aber auf Ihrer Platte Dateien in den Bereichen 0, 1 und 3 haben. Diese Funktion ist in den Versionen 1.3 und 1.4 des CP/M-80 nicht verfügbar.

Spezielle Eingaben:
Keine

STAT d:DSK:<cr>

Zeige an, wie Daten auf der Platte in Laufwerk d: gespeichert sind.

Beispiel

```
A>STAT B:DSK:<cr>
B: DRIVE CHARACTERISTICS
     4Ø96: 128 BYTE RECORD CAPACITY
      512: KILOBYTE DRIVE CAPACITY

      128: 32 BYTE DIRECTORY ENTRIES
      128: CHECKED DIRECTORY ENTRIES
      128: RECORDS/EXTENT
       16: RECORDS/BLOCK
       58: SECTORS/TRACK
        2: RESERVED TRACKS
A>
```

Das d: (Laufwerkskennzeichnung) ist wahlweise; wenn er nicht angegeben ist, wird Information für alle Platten angezeigt, auf die seit dem letzten Kalt- oder Warmstart zugegriffen wurde.

Die wichtigste Information der Anzeige ist „KILOBYTE DRIVE CAPACITY", die den freien Speicherplatz auf einer leeren Platte anzeigt, und „32 BYTE DIRECTORY ENTRIES", die die maximale Anzahl von Dateien angibt, die Sie auf der Platte speichern können (diese Anzahl wird sich reduzieren, wenn Sie Dateien mit mehr als einem Speicherbereich haben).

Die Information der STAT d:DSK:-Anzeige wird sich nicht ändern, solange Sie CP/M nicht modifizieren. Diese Information reflektiert Designentscheidungen darüber, wie Daten auf den Platten zu arrangieren sind.

Die folgenden Definitionen erklären jede der STAT-Anzeigen, die Ihnen auf dem CRT: präsentiert werden:

128 BYTE RECORD CAPACITY
Die maximale Anzahl von Sektoren (128-Byte-Sätzen), die Sie auf der Platte speichern können.

KILOBYTE DRIVE CAPACITY
Die maximale Anzahl von K, die Sie auf der Platte speichern können.

32 BYTE DIRECTORY ENTRIES
Die maximale Anzahl von Dateien, die Sie auf der Platte speichern können.

CHECKED DIRECTORY ENTRIES
Normalerweise das gleiche wie „32 BYTE DIRECTORY ENTRIES" für Laufwerke mit auswechselbaren Speichermedien (Disketten); normalerweise „0" für Laufwerke mit nicht-auswechselbaren Medien (versiegelte Hartplatten).

RECORDS/EXTENT
Die maximale Anzahl von Sätzen für einen Eingang der Directory.

RECORDS/BLOCK
Der minimale Platten-Speicherbereich, der einer Datei zugewiesen werden kann.

SECTORS/TRACK
Die Anzahl von Sektoren, in die eine Spur unterteilt ist (s. Kap. 1)

RESERVED TRACKS
Die Anzahl von Spuren, die nicht für die Speicherung von Dateien zur Verfügung steht.

Spezielle Eingaben:
Keine

STAT d: = R/O <cr>

Ordne der Platte in Laufwerk d: einen temporären Schreibschutzstatus (read-only) zu.

Beispiel

```
A>STAT B:=R/D<cr>
A>SAVE 2 B:JUNK.FIL
BDOS ERR ON B: R/O
```

In diesem Beispiel setzen wir durch STAT zuerst Laufwerk B: auf Read-only-Status. Beachten Sie, daß STAT diese Anforderung nicht bestätigt. Als nächstes haben wir versucht, eine neue Datei auf die Platte zu sichern und haben eine Meldung BDOS ERR erhalten, die anzeigt, daß CP/M die Platte als schreibgeschützt erkennt. Ein Warm- oder Kaltstart hebt den temporären R/O-Status auf.

Spezielle Eingabe

Jeder nachfolgende Warm- oder Kaltstart hebt den temporären R/O-Status auf.

PIP – Kopieren von Informationen

PIP: Peripheral Interchange Program (= Periphäres Austausch-Programm), kopiert Information. Die Dateinamen, die Sie kopieren wollen, können mehrdeutig sein, und eine Anzahl wahlfreier Parameter sind verfügbar. PIP kann auch Information von und auf Einheiten kopieren.

PIP kann auf zwei Arten aufgerufen werden. Wenn Sie nur eine Operation mit PIP ausführen wollen und es nicht nötig ist, die Diskette in A: auszutauschen, benutzen Sie PIP folgendermaßen:

```
A>PIP pipcommandline<cr>
```

Wenn Sie mehrere Operationen auszuführen haben, nutzen Sie die folgende zweite Methode, PIP aufzurufen:

```
A>PIP<cr>                    CP/M-Kommandozeile
*pipcommandline<cr>          PIP-Kommandozeile
   .
   .                         beliebige PIP-Kommandos
   .
*pipcommandline<cr>          PIP-Kommandozeile
*^C                          ^C springt nach CP/M zurück
A>
```

Gleichgültig welche Methode Sie benutzen, es besteht kein bedeutender Unterschied in der Weise, in der PIP die durch die PIP-Kommandozeile geforderte Operation ausführt. Ein geringer Unterschied wird später unter Parameter „Q“ in diesem Abschnitt erklärt. Eine andere kleine Differenz besteht in PIP's Antwort auf Fehler. Wenn die erste Aufrufsmethode angewandt wird, bewirken Fehler einen Rücksprung nach CP/M. Bei der zweiten Methode bewirken Fehler einen Rücksprung zur PIP-Anforderung. Die PIP-

Kommandozeile (in unseren Beispielen „pipcommandline") hat gewöhnlich die folgende Form:

Ziel = Ursprung [Parameter]

wobei „Ziel" der Name der neuen, „Ursprung" der Name der alten Datei ist und „Parameter" die wahlfreien Parameter sind, die Sie für den Prozeß des Kopierens auswählen.

Wenn bereits eine Datei mit dem Namen Ihrer Zieldatei existiert, wird diese nach korrekter Vollendung der Kopie gelöscht. Die vorherigen Inhalte der Zieldatei sind verloren.

PIP-Kommandozeilen und spezielle Parameter spezifizieren die Übertragung zwischen zwei Dateien oder zwei Dateigruppen. In jedem Fall bleibt die Originaldatei intakt; wir können eine vollständige Platte oder auch nur einen Teil einer Datei duplizieren, ohne das Original zu ändern. Wir können spezifizieren, daß gewisse Kennzeichen der neuen Datei ohne Änderung des Originals modifiziert werden. Es folgen einige Anwendungen von PIP (Dateien):

- Kopieren einer Datei auf eine andere Platte
- Kreieren einer identischen Datei unter einem anderen Namen
- Kopieren mehrerer Dateien auf eine andere Platte
- Kopieren aller Dateien auf eine andere Platte
- Kreieren einer Datei aus der Verkettung anderer
- Kopieren eines Teils einer Datei
- Kopieren von einer Systemdatei
- Kopieren auf eine R/O-Datei
- Inhaltsangabe der kopierten Datei während des Vorgangs.

PIP kopiert ebenfalls Daten von einer Datei auf eine Einheit, von einer Einheit auf eine Datei oder zwischen Einheiten; Sie geben einfach einen Einheitennamen anstelle eines Dateinamens an, entweder im Zielfeld, im Ursprungsfeld oder in beiden.

Wenn sie auf bzw. von einer Plattendatei anstelle einer Einheit lesen bzw. schreiben, stellen Sie sicher, daß Sie den Prozeß des Kopierens beenden können. Einige Einheiten besitzen kein Kennzeichen für das Dateiende (^Z), wie CP/M es an das Ende der Plattendatei setzt. Es folgen einige Anwendungen von PIP (Einheiten):

- Senden von Dateiinhalten an eine Einheit, wie z. B. einen Drucker
- Übergabe der Daten von einer Einheit wie Lochkartenleser oder Modem an eine Datei
- Senden von Daten an eine andere Einheit

- Ausdruck eines Dateiinhalts mit Formatierung für ein spezielles Drucker- oder Papierformat
- Anzeige ausgewählter Daten, die auf einer Eingabeeinheit ankommen, auf der Konsole
- Konvertierung von Groß- in Kleinbuchstaben oder umgekehrt
- Sichern aller Daten, die auf einer Eingabeeinheit ankommen, in eine Datei

PIP hat fünf „spezielle" Einheitennamen zusätzlich zu denen, die in dem Abschnitt über STAT behandelt wurden.

NUL:
Eine Ursprungs-Einheit, die 40 Nullen („do-nothing"-characters, z. B. 00 hex), zum angegebenen Ziel sendet.

EOF:
Eine Ursprungseinheit, die einen Dateiendemerker (^Z oder 1A hex) zum Ziel sendet.

OUT:
Eine vom Benutzer eingeführte Zieleinheit. PIP muß modifiziert werden, um diese Einheit einzuschließen.

INP:
Eine vom Benutzer eingeführte Ursprungsdatei. PIP muß modifiziert werden, um diese Einheit einzuschließen.

PRN:
Eine spezielle Form der Einheit LST:, die Tabellen ausweitet, Zeilen numeriert und die Kopie paginiert (genauso wie LST: [NPT8]).

Wie bei den vorher besprochenen Einheiten werden auch diese Einheitennamen als Ursprünge und Ziele in der Kommandozeile angewandt.

Prüfen Sie die folgenden Seiten. Jeder Parameter einer PIP-Kommandozeile wird mit einer Erklärung der Computeranzeige dargestellt. Erforschen Sie mögliche Kombinationen von PIP-Kommandozeilen, -Parametern und -Möglichkeiten sowie die resultierenden Variationen der aufgelisteten Zwecke.

PIP <cr>

Lade PIP in den Kernspeicher.

Beispiel:

A>PIP<cr>	Kommandozeile
*	PIP-Anforderung

Wenn die PIP-Anforderung (*) erscheint, können Sie eine korrekte PIP-Kommandozeile eingeben oder CARRIAGE RETURN bzw. ^C drücken, um nach CP/M zurückzukehren.

Spezielle Eingaben:

Ein CARRIAGE RETURN oder ^C löscht PIP und verzweigt zurück zu CP/M

PIP d:new.typ=d:old.typ[p]<cr>

Kopiere die Datei „old.typ" vom angegebenen Laufwerk auf die Datei „new.typ" auf dem hier angegebenen Laufwerk unter Kontrolle der Parameter ([p]).

Beispiele:

```
B>PIP LETTER2.DOC=LETTER1.DOC<cr>
B>
```

Eine simple Anwendung des Kommandos PIP, um eine Datei unter einem neuen Dateinamen zu kopieren.

```
B>PIP A:LETTER2.DOC=B:LETTER1.DOC<cr>
B>
```

PIP kopiert von Platte B: auf Platte A:.

```
B>PIP LETTER2.DOC=LETTER1.DOC[V]<cr>
B>
```

Dieses Beispiel zeigt den Gebrauch eines Parameters, in diesem Falle das „V" (für Verify [= Verifizieren]).

```
A>PIP B:=A:DOCUMENT.LET[V]<cr>
A>

A>PIP B:DOCUMENT.LET=A:[V]<cr>
A>
```

Zwei Kurzmethoden, einen Dateinamen zu spezifizieren, wenn er auf beiden Laufwerken benutzt werden soll. Eine der beiden Seiten des Gleichheitszeichens kann auf den Laufwerks-Identifikator reduziert werden.

```
A>PIP B:=A:*.*[V]<cr>
COPYING-
DOC.ONE
DOC.TWO
LET.ONE
A>
```

Ein Beispiel, das die eben gezeigte Methode (B:=) mit mehrdeutigen Datei-Referenzen (A:*.*) benutzt. In so einem Fall listet PIP die Dateinamen, die Ihrer Anforderung entsprechen, während es sie kopiert.

Spezielle Eingaben:

Keine

PIPd:new.typ = d:old1.fil[p],d:old2.fil[p] <cr>

Kreiere auf dem angegebenen Laufwerk eine neue Datei, die sich aus den beiden Dateien old1.typ und old2.typ zusammensetzt. Mit anderen Worten, ein Komma in der PIP-Kommandozeile zeigt eine Verkettung an.

Beispiele

```
A>PIP B:DOC = A:JUNE[V],A:JULY<cr>
A>
```

Dieses Beispiel würde in einer neuen Datei namens DOC die Daten aus JUNE und JULY in der angegebenen Reihenfolge kombinieren.

Spezielle Eingaben:

Keine

PIP dev: = d:filename.typ[p] <cr>

Sende die Dateiinhalte auf dem spezifischen Laufwerk an eine Einheit.

Beispiel

```
B>PIP LST: = B:LAFFERTY.RT<cr>
B>
```

Eine Kopie der Datei LAFFERTY.RT wird an die LST:-Einheit übermittelt.

ANMERKUNG: Sie können nicht mit Hilfe von mehrdeutigen Dateireferenzen eine ganze Reihe von Dateien an die LST:-Einheit senden.

Akzeptable Zieleinheiten sind

CON:	PUN:	LST		Logisch
TTY:	PTP:	LPT:	CRT:	Physisch
UP1:	UP2:	UL1:	UL2:	
OUT:	PRN:			Speziell

Spezielle Eingabe

Sie können ein PIP für Einheiten während des Kopiervorgangs stoppen, indem Sie irgendeine Taste drücken.

PIP d:filename.typ = dev:[p] <cr>

Kopiere die Eingabe von einer Einheit auf den angegebenen Dateinamen.

Beispiel

```
B>PIP B:DOCUMENT.MAY = CON: <cr>
I AM NOW TYPING FROM CONSOLE. ^Z
B>
```

Dieses Beispiel erstellt eine Datei aus der auf der Konsoleinheit eingegebenen Information (bis zum und einschließlich des abschließenden ^Z).

Akzeptable Ursprungseinheiten sind

CON:	RDR			Logisch
TTY:	PTR:	CRT:	UR1:	Physisch
UR2:	UC1:			
NUL:	EOF:	INP:		Speziell

Spezielle Eingabe

Beide, CARRIAGE RETURN (^M) (= Zeilenende) und LINE FEED (^J) (= Zeilenvorschub) müssen am Ende jeder Zeile angegeben werden. ^Z beendet die Übertragung und zeigt das Dateiende an.

PIP destinationdev: = sourcedev:[p] < cr >
Kopiert Daten auf eine andere Einheit.

Beispiel

```
A>PIP CON:=PTR:[U]<cr>
THIS IS A PAPER TAPE READING DATA
INTO THE SYSTEM, WHICH DISPLAYS IT
ON THE DESTINATION DEVICE, THE CONSOLE.
A>
```

In diesem Beispiel sollen Daten vom Lochstreifenleser auf die Konsolanzeige kopiert werden. Außerdem spezifizierten wir den wahlfreien Parameter, um alle Information in Großbuchstaben zu konvertieren ([U]).

PIP-Parameter

Dieser Abschnitt beschreibt jeden der wahlfreien Parameter von PIP. Sie werden wahrscheinlich meistens nur einen oder zwei dieser Parameter benutzen und einige werden Sie wohl nie gebrauchen. Sie können zwei oder mehr Parameter in eckigen Klammern kombinieren (z. B. [BEU]).

[B]

Blockübertragungsparameter. Der Parameter [B] bezeichnet eine Übertragung im „block mode". Im Block-Modus überträgt PIP Information von der Quelle in einen Puffer, bis das ASCII-Zeichen DC3 (53 hex oder ^S, manchmal XOFF oder „reader off" genannt) erscheint. Nachdem dieses Zeichen empfangen wurde, nimmt PIP die bis dahin übermittelte Information und sendet sie an die spezifizierte Zieleinheit. Danach fragt PIP die Quelle nach mehr Daten. Wir benutzen [B] für die Quelleinheiten, die fortlaufend Daten übertragen, im anderen Fall würden wir den Pufferbereich von PIP (wo die Information vor der Weitergabe an die spezifizierte Datei oder Einheit gesammelt wird) auffüllen, wenn der Puffer nicht periodisch gesäubert würde. Für normale Datei-zu-Datei-Operationen ist [B] nicht notwendig.

Normalerweise wird der Parameter [B] benutzt, um einen Lochstreifen in eine Datei zu lesen. Was auch immer die Quelleinheit sein mag, sie muß das

Zeichen DC3 oft genug senden, um zu verhindern, daß der Puffer gefüllt wird und PIP ihn dann leeren soll.

Der Parameter [B] wird im System CP/M-86 nicht benutzt.

[D#]

Lösche (delete) alle Zeichen nach der #. Spalte. Der Parameter [D#] weist PIP an, alle nach einer bestimmten Spaltennummer empfangenen Zeichen zu löschen. Diese Nummer muß direkt hinter dem „D" angegeben werden, z. B. [D20]. Diese Art von Übertragung wird nur für zeilenorientierte Daten anwendbar sein (Daten, die periodisch CARRIAGE RETURN-Zeichen enthalten). Nach der Entdeckung eines CARRIAGE RETURN wird PIP die Zeichen bis zu der angegebenen Spaltennummer zählen und verarbeiten und alle übrigen Zeichen bis zum nächsten CARRIAGE RETURN ignorieren.

(**Anmerkung:** Bei dieser Operation wird ein Zeilenvorschub genau wie ein CARRIAGE RETURN-Zeichen interpretiert).

Der Parameter [D#] wird in erster Linie benutzt, um eine weitzeilige Ausgabe an eine Einheit zu senden, die nur enge Zeilen kennt (wie z. B. Drucker oder CRT-Terminal). Für Datei-zu-Datei-Übertragungen wird diese Angabe kaum benutzt.

Der Wert „#" ist eine dezimale Ganzzahl im Bereich zwischen 1 und 255. Eingabezeilen von größerer Länge als 255 Zeichen werden durch die [D#]-Option nicht automatisch abgeschnitten.

[E]

Wiederhole (= echo) die Kopie während der Ausführung auf der Konsole. Die an die Zieleinheit ausgegebenen Zeichen werden während des Kopierprozesses auf der Konsolanzeige wiederholt. Dies ist eine einfache Methode, um genau zu überprüfen, was kopiert worden ist.

[F]

Siebe (= filter) Formularvorschübe aus der Originaldatei aus. Die [F]-Option „filtert" Formularvorschubzeichen (0C hex) aus dem Datenfluß; PIP ignoriert das ASCII-Zeichen für Formularvorschub und kopiert ohne es. Formularvorschübe werden oft in Dateien eingebettet, um die gedruckte Ausgabe korrekt zu paginieren. Dies werden Sie womöglich ändern wollen um Papier zu sparen, um eine Datei auf dem Bildschirm auszugeben oder um einen Drucker zu benutzen, der diese Zeichen falsch interpretiert.

Das ASCII-Formularvorschub-Zeichen heißt „FF" und entspricht hexadezimal 0C und CONTROL-L.

Das Formularvorschub-Zeichen kontrolliert das Layout oder die Positionierung von Information in Druck- oder Anzeigeeinheiten. So würde ein Drucker auf dieses Zeichen nicht durch Drucken reagieren, sondern durch einen Papiervorschub zum Beginn des nächsten Formulars oder der nächsten Seite.

Benutzen Sie [F], wenn Ihr Drucker auf Formularvorschub-Zeichen nicht

korrekt reagiert oder wenn Sie eine Datei auf eine andere Einheit als einen Drucker übertragen (s. a. den [P#]-Parameter).

[G#]
Veranlasse (= direct) PIP, Dateien von anderen Benutzerbereichen zu kopieren.

Die Option [G#] erlaubt PIP, Dateien von einem anderen Benutzerbereich zu dem derzeit laufenden zu kopieren. Eine Dezimalzahl (zwischen 0 und 15) repräsentiert den Quellbenutzerbereich und sollte direkt dem „G" in der Kommandozeile folgen. Diese Option ist hilfreich, um einen Benutzerbereich mit Dateien zu füllen, die dort vielleicht später gebraucht werden. Der Parameter [G#] ist in den Versionen 1.3 und 1.4 von CP/M-80 nicht verfügbar.

[H]
Prüfe eine Datenübertragung auf korrektes Intel-hex-Format.

Der Parameter [H] spezifiziert, daß die zu übertragenden Daten im speziellen Intel-hex-Format sein müssen anstatt in normalem ASCII- oder binären Formaten. Das Intel-Format wird normalerweise für Lochstreifenstanzer oder -leser benutzt. Die meisten Anwender brauchen diese Option nicht.

Wenn PIP Fehler im Hex-Format findet, wird eine Anforderung mit einer Anfrage angezeigt, welche Aktion zur Korrektur unternommen werden soll (s. a. den Parameter [I]).

[I]
Ignoriere alle Null-Sätze bei Übertragungen im Intel-hex-Format.

Die Option [I] bezieht sich ebenfalls auf Sätze im Intel-hex-Format (s. [H]). Der Parameter [I] weist das Programm PIP an, alle Daten zu ignorieren, die in einem Null-Satz („00:" im Intel-Format) erscheinen. Auch diese Option wird selten von Endanwendern benötigt, weil sie sich in erster Linie auf Lochstreifenleser und -stanzer bezieht.

Wenn die [I]-Option spezifiziert wird, wird die Option [H] von PIP automatisch gesetzt. Die PIP-Kommandozeile

```
PUN:=PROGRAM.HEX[I]
```

ist also identisch mit

```
PUN:=PROGRAM.HEX[HI]
```

[L]
Konvertiere Groß- in Kleinbuchstaben.

Die Option [L] erlaubt während des Kopierprozesses die Buchstaben „A" bis „Z" in ihre äquivalenten Kleinbuchstaben umzuwandeln.

Seien Sie sehr vorsichtig in der Anwendung dieser Option. Sie sollte nie bei Dateien benutzt werden, die Computer-Instruktionen enthalten. Wenn Sie sie bei einer Programmdatei anwenden, könnten Sie wichtige Instruktionen ändern, die PIP nur als Großbuchstaben erkennt.

[N]
Füge während der Datenübertragung jeder Zeile eine Zeilennummer hinzu.

Die Option [N] fügt jeder übertragenen Datenzeile eine Zeilennummer hinzu. Jedesmal, wenn PIP ein CARRIAGE RETURN (oder LINE FEED) erkennt, wird die Zeilennummer erhöht. [N] ist von besonderer Bedeutung, wenn eine Datei auf den Drucker übertragen wird. Wird die Option [N2] angegeben, werden führende Nullen hinzugefügt und die Nummer als Sechs-Zeichen-Feld, gefolgt von drei Leerzeichen, gedruckt. Das ist der Unterschied in der Übertragung von Dateien zwischen diesen beiden Optionen.

Original	**Datei mit [N]**	**Datei mit [N2]**	
Nun ist die	1: Nun ist die	000001	Nun ist die
Zeit für alle	2: Zeit für alle	000002	Zeit für alle
guten Menschen,	3: guten Menschen,	000003	guten Menschen,
ihrer Partei	4: ihrer Partei	000004	ihrer Partei
zu Hilfe zu	5: zu Hilfe zu	000005	zu Hilfe zu
kommen.	6: kommen.	000006	kommen.

[O]
Übertrage Dateien mit Objekt- oder anderem Nicht-ASCII-Code.

Benutzen Sie den Parameter [O], wenn Sie von Nicht-ASCII-Daten (wie Programm, Objekt-Code- oder Binärdaten-Dateien) oder von Einheiten kopieren wollen, die andere als ASCII-Daten senden.

Bei der Kopie von Dateien des Typs „.COM" ist die Angabe von [O] nicht notwendig, weil PIP annimmt, daß „.COM"-Dateien Nicht-ASCII-Dateien (keine Textdateien) sind.

Bei der Benutzung von [O] behandelt PIP CONTROL-Z (1A hex) wie jedes andere Zeichen, das es von einer Quelldatei oder -einheit empfängt; im anderen Fall würde PIP ^Z als Signal für das Ende der Übertragung von dieser Quelle interpretieren.

Die folgenden Abschnitte enthalten detailliertere Information über ASCII-Dateien, Nicht-ASCII-Dateien und Dateiendemerker. Lesen Sie weiter, wenn Sie an diesen Details interessiert sind.

PIP nimmt an, daß ein ^Z von einer ASCII-Quelle (Einheit oder Datei) als Dateiendemerker oder Datenterminator empfangen wurde, der signalisiert, daß alle Daten übertragen sind.

PIP sendet ein ^Z-Zeichen an ein ASCII-Ziel (Datei oder Einheit), um anzuzeigen, daß alle Daten gesendet wurden.

Die Differenz in der Behandlung von Nicht-ASCII-Daten ist notwendig, weil CP/M unterschiedliche Methoden für die Markierung und Entdeckung des Endes von ASCII- und Nicht-ASCII-Dateien benutzt.

CP/M markiert das Ende einer ASCII-Datei durch ein ^Z-Zeichen nach dem letzten Datenzeichen. Wenn eine Datei ein exaktes Mehrfaches von 128

Zeichen enthält, wenn also ein zusätzliches ^Z 127 Zeichen Speicherplatz verbrauchen würde, führt CP/M dies nicht aus. Der Gebrauch von Zeichen ^Z als Dateiendemerker ist möglich, weil ^Z in ASCII-Dateien selten als Daten-Zeichen benutzt wird.

In einer Nicht-ASCII-Datei hingegen kann ^Z genausogut wie jedes andere Zeichen auftreten. Deshalb kann es hier nicht als Dateiendemerker benutzt werden. CP/M benutzt eine unterschiedliche Methode, um das Ende einer solchen Datei zu markieren. Es nimmt an, daß es das Dateiende erreicht hat, wenn es den letzten Satz (Basiseinheit für Speicherplatz auf Platten), der dieser Datei zugewiesen war, gelesen hat. Der Eingang für jede Datei im Platten-Inhaltsverzeichnis enthält eine Liste der Plattensätze, die dieser Datei zugeordnet wurden. Diese Methode verläßt sich mehr auf den Umfang der Datei als ihren Inhalt, um das Dateiende zu lokalisieren.

[P#]
Bewirke einen Formularvorschub nach der #ten Zeile.

Benutzen Sie den Parameter [P#], wenn LST: die Empfangseinheit ist und die laufende LST:-Einheit (normalerweise ein Drucker) die Quelldaten nicht in folgendem Format ausgibt:

1. Eine leere Spanne am Anfang
2. Eine leere Spanne am Ende
3. Beide Spannen zusammen sechs Zeilen
4. Der Raum zwischen Anfang und Ende mit Text ausgefüllt.

Abweichungen von diesen Bedingungen ergeben sich gewöhnlich, wenn die laufende LST:-Einheit

1. das Formularvorschub-Zeichen nicht kennt (s. [F]),
2. nicht automatisch Anfangs- und Endespannen einfügt oder
3. in unterschiedlicher Seitenlänge als von den Quelldaten angenommen ausdruckt.

Die [P#]-Option teilt PIP mit, daß Sie einen Formularvorschub nach der jeweiligen Anzahl von Datenzeilen wünschen. Tragen Sie die gewünschte Zeilenanzahl anstelle des Nummernzeichens (#) im Parameter [P#] der PIP-Kommandozeile ein. Das Zeichen „#" repräsentiert eine Dezimalzahl im Bereich zwischen 1 und 255. Wenn Sie keine Nummer oder die Nummer 1 angeben, nimmt PIP an, daß Sie einen Seitenvorschub nach jeweils 60 Zeilen wünschen. Wenn [F] ebenfalls im Parameterfeld angegeben ist, werden alle während der Übertragung entdeckten Formularvorschübe entfernt, und PIP wird neue Formularvorschübe gemäß [P#] einfügen. Mit anderen Worten, [F] und [P#] zusammen werden die Seitenbegrenzung durch Formularvorschub-Zeichen in den Quelldaten überlesen.

Ein Drucker, der das Formularvorschub-Zeichen erkennt, reagiert darauf,

indem er sofort das Papier auf den Beginn der nächsten Seite vorschiebt, gleichgültig, ob die laufende Seite voll ist oder nicht. Drucker, die es nicht kennen, werden falsch reagieren.

Vergleichen Sie auch den Abschnitt über die Einheit PIP PRN:.

[Qstring^Z]

Kopiere einen Teil der Datei bis zu der gelisteten Zeichenkette.

Die Benutzung des Parameters Q erlaubt Ihnen, PIP eine Reihe von Zeichen anzugeben, nach denen es den Kopierprozeß abbricht, wenn es sie findet. Suchen Sie diese Zeichenkette sorgfältig aus; sie muß eindeutig sein, damit sichergestellt ist, daß Sie genau den gewünschten Teil der Datei kopieren. „Q" bezeichnet den Anfang der Kette und ^Z das Ende: [Qstring^Z].

Die dem „Q" folgende Zeichenkette wird in Großbuchstaben umgesetzt, wenn die Kommandozeile nach der CP/M-Anforderung (A>) getippt wird. Wenn das Kommando nach der PIP-Anforderung (*) eingegeben wird, wird nicht automatisch konvertiert. Um Kleinbuchstaben in die Kette einzuschließen, muß Ihre Kommandozeile der PIP-Anforderung folgen (s. a. den Parameter [S]).

[R]

Erlaube PIP, eine Systemdatei zu kopieren.

Nur wenn wir ein „R" an die Kommandozeile anhängen, können wir Systemdateien kopieren. Wenn das Attribut „Systemdatei" vorhanden ist, wird es kopiert. Dieser Parameter ist für die Version 1.3 und 1.4 des CP/M-80 nicht verfügbar.

[Sstring^Z)

Kopiere den mit der Zeichenkette beginnenden Teil der Datei.

Wie die Option [Q] spezifiziert die [S]-Option, daß Sie mit PIP nur den Teil einer Datei kopieren wollen, der an der direkt auf das „S" folgenden Zeichenkette im Parameter beginnt. ^Z beendet diese Kette im Parameter. Beide Optionen [S] und [Q] können in derselben Kommandozeile auftreten.

Vergleichen Sie die automatische Konvertierung von Klein- zu Großbuchstaben unter der Option [Q].

[T#]

Setze Tabulatorstops nach jeder #ten Spalte.

Der Parameter [T#] weist PIP an, den Text nach jedem TAB-Zeichen (09 hex oder CONTROL-I), das es entdeckt, um die nötige Anzahl von Leerzeichen zu erweitern, um den nächsten Tabulatorstop zu erreichen. Dies ist nützlich, da die Programme einiger Herausgeber TABs nicht als Leerzeichen, sondern als TAB-Zeichen speichern, und nicht alle Drucker oder Terminals in der Lage sind, TAB-Zeichen zu erkennen. Benutzen Sie [T#], um die Standardtabellierung von acht Zeichen (oder welche Zahl auch immer Standard für Ihre Konsole oder ihren Drucker ist) in eine von Ihnen definierte Anzahl zu ändern.

[U]
Übersetze Klein- in Großbuchstaben.

Die Option [U] erlaubt, während des Kopierprozesses die Buchstaben „a" bis „z" in ihre äquivalenten Großbuchstaben umzuwandeln.

[V]
Verifiziere die Richtigkeit der Kopie, indem du den Puffer im Kernspeicher mit der neu erstellten Datei oder Dateiengruppe vergleichst.

Das Empfangsgerät muß eine Plattendatei sein, anderenfalls wird [V] ignoriert. Nachdem die Übertragung vollendet ist, wird die Zieldatei erneut gelesen und mit den Daten im Kernspeicherpuffer von PIP verglichen. Das bedeutet, daß der Parameter [V] Schreib-, aber nicht Lesefehler in einer Datei entdecken wird.

Wenn mehrere Quellen verknüpft werden, muß [V] dem ersten Namen folgen; nur ein [V] ist notwendig.

> **Anmerkung:** Gary Kildall, der Urheber von CP/M, benutzt den [V]-Parameter nie und Mitglieder der Crew von Digital Research weisen daraufhin, daß sie in vier Jahren noch keinen Verify-Fehler gesehen haben, der nicht von der Meldung „BDOS ERR ON d: BAD SECTOR" begleitet war. Es ist deshalb zweifelhaft, ob der Parameter [V] sich für die Auffindung sonst ungesehener Fehler als effektiv erweist.

[W]
Erlaube PIP, in eine Datei mit dem R/O-Attribut zu kopieren.

Da [W] Ihnen erlaubt, auf eine R/O-Datei zu schreiben, prüfen Sie, ob dieses Attribut nicht mehr nötig ist, bevor Sie fortfahren.

Wenn Sie es versuchen, ohne das [W]-Attribut zu benutzen, werden Sie die Meldung erhalten:

DESTINATION FILE IS R/O, DELETE (Y/N)?

Der Parameter [W] ist in den Versionen 1.3 und 1.4 des CP/M-80 nicht verfügbar.

[Z]
Setze während der Übertragung das Prüf-Bit auf Null.

Gebrauchen Sie den [Z]-Parameter, um das ungenutzte achte Bit eines ASCII-Zeichens von einer Einheit auf Null zu setzen.

Jedes ASCII-Zeichen benutzt sieben der acht Bits, die vom Computer verarbeitet werden. Das achte Bit wird manchmal Parity-Bit (Prüf-Bit) genannt. Um rätselhaften Problemen mit ASCII-Daten vorzubeugen, ist es weise, das ungenutzte Bit auf Null zu setzen. Der Gebrauch des [Z]-Parameters mit von WordStar erstellten Dateien resultiert darin, daß alle „soft spaces" und „soft carriage returns" (Leerzeichen und <cr>s, die von WordStar zur Textformatierung eingefügt wurden) in reale Leerzeichen und Rückläufe umgewandelt werden. Je nach Gebrauch kann [Z] bei einer von WordStar erstellten Datei nützliche oder schädliche Folgen haben.

Spezielle PIP-Einheiten

EOF:
Sende einen Dateiende (end of file)-Merker an die Empfangseinheit.

Wenn eine Plattendatei des Typs ASCII kopiert wird, wird der EOF-Merker automatisch an die Empfangseinheit gesendet. In einigen Spezialfällen kann EOF: benutzt werden, um die Übertragung zu beenden. Gerade wenn Dateien von einem Computer zum anderen gesandt werden, ist es oft notwendig, die Übertragung durch das EOF:-Einheitszeichen zu beenden.

NUL:
Sende 40 Null-Zeichen an die Empfangseinheit.

NUL: wurde ursprünglich gebraucht, um eine gestanzte Ausgabe (Output) auf einem Lochstreifen mit einem Leerbereich zu beenden; es sah ein Kopf- sowie ein Endetikett für die aktuelle Information vor, die geeignet waren, den Streifen zu steuern. NUL: produziert einen Leerstreifen von 4 Zoll Länge.

Das ASCII-Null-Zeichen heißt „Null" und ist äquivalent zu 00 hex, sprich CONTROL-@.

PRN:
Sende Daten mit speziellen Instruktionen an die LST:-Einheit.

Die Einheit PRN: spezifiziert zusätzliche Instruktionen für die Einheit LST:. Sie kann Instruktionen enthalten, die Tabulatoren ausweiten, Zeilen numerieren oder die Kopie paginieren.

Tabulatorstops werden an jeder achten Spalte angenommen. Immer wenn ein TAB-Zeichen entdeckt wird, fügt PIP stattdessen Leerzeichen ein. Alle Zeilen werden von 1 angefangen numeriert, und Seitenvorschübe werden alle 60 Zeilen auf dem Drucker vorgenommen (wobei bei normalem 11 Zoll-Papier 3 Zeilen am Beginn und 3 am Ende eingeschossen werden). PRN: wird zur Programmentwicklung und für die Auflistung der Inhalte einer ASCII-Datei benutzt, die Zeilennummern liefern erkennbare Bezugspunkte innerhalb der Datei.
PRN: ist äquivalent zu [NPT8].

INP: und OUT:
Spezielle Einheitslaufwerke für PIP.

Die Einheit INP: wird zur Wiedererlangung von einer speziellen vom Benutzer erstellten PIP-Eingabequelle benutzt. OUT: sendet Information an eine spezielle vom Benutzer erstellte Ausgabedatei.

Sie können bestimmte Input- und Outputroutinen in Maschinensprache hinzufügen. Die hexadezimalen Speicherstellen 103, 104 und 105 sind für einen Sprungbefehl zu Ihrer speziellen Inputroutine reserviert. PIP ruft Stelle 103 hex auf, um ein Zeichen einzugeben und erwartet als Antwort das Zeichen in der Stelle 109 hex. Die hexadezimalen Speicherplätze 106, 107 und 108 sind für den Sprung in Ihre spezielle Outputroutine reserviert. PIP lädt das Regi-

ster C mit dem zu übermittelnden Zeichen und ruft dann Stelle 106 hex auf. Zusätzlich ist der Bereich von 10A bis 1FF hex frei für Ihre Routinen. Es ist allerdings selten, daß von dieser Möglichkeit Gebrauch gemacht wird.

ED – Kontext-Editor

ED ist ein Programm mit einer Anzahl eingebauter Kommandos, um Textdaten auszugeben.

Sie mögen sich gefragt haben, wie Sie etwas Neues auf eine Platte bekommen. Die Antwort ist, Sie benutzen einen Editor. CP/M schließt ihn, als Datei ED.COM gesichert, ein.

Der Editor ist ein Programm, das Zeichen von der Tastatur aufnimmt und sie in eine Plattendatei ausgibt. Da Sie Eingabefehler machen oder Änderungen wünschen könnten, schließt der Editor auch eine Anzahl eingebauter Kommandos ein, die Information anzeigen, modifizieren, löschen und hinzufügen.

Der CP/M-Editor (von Digital Research Context Editor genannt) ist sowohl zeichen- als auch zeilenorientiert; die Kommandos operieren mit Text entweder zeichen- oder zeilenweise zur Zeit. Eine Zeile ist ein Block von Zeichen, der mit einem CARRIAGE RETURN endet.

Um die Datei filename.typ auszugeben, tippen Sie:

```
A>ED filename.typ<cr>
```

Einmal in den Kernspeicher geladen, erstellt das Editor-Programm eine neue Datei mit dem angegebenen Dateinamen und dem Typ „.$$$“ und wartet auf Ihr Kommando. Sie können nun die eingebauten Kommandos benutzen, um

- irgendeinen Teil der existierenden Datei zu zerstören oder zu ändern o d e r
- neue Information von der Tastatur oder von einer anderen Datei in diese Datei einzufügen.

Nehmen wir z. B. an, die Datei DOCUMENT.MAY soll editiert werden. Um ED aufzurufen, tippen Sie

```
A>ED DOCUMENT.MAY<cr>
```

ED erstellt die temporäre Datei DOCUMENT.$$$ auf der Platte. Diese Datei enthält keine Information, sondern lediglich einen Eingang (Entry) im Inhaltsverzeichnis.

Um Änderungen auf der Originaldatei DOCUMENT.MAY vornehmen zu können, müssen Sie eine Möglichkeit haben, den Text zu sehen und zu manipulieren. Die Kommandos von ED erlauben Ihnen, Text von der Originaldatei in den Kernspeicher zu übertragen, ihn sichtbar zu machen, die

erforderlichen Änderungen vorzunehmen und ihn dann zurück in die Datei zu speichern. Sie können eine Datei nur editieren, während sie sich im Kernspeicher befindet. Der Kernspeicherbereich, den ED für den Text freihält, wird Edit-Puffer genannt.

Wenn Sie in der Editierung fortschreiten, werden Sie immer mehr Text von der Originaldatei in den Edit-Puffer übertragen. Dieser kann jedoch nur eine begrenzte Datenmenge aufnehmen. Wenn er voll wird, müssen Sie ihn in die temporäre Datei DOCUMENT.$$$ übertragen.

Wenn Sie die Editierung abschließen, kann sich ein Teil des Textes in der Originaldatei DOCUMENT.MAY, ein Teil im Edit-Puffer und ein Teil in der temporären Datei DOCUMENT.$$$ befinden. Um das Edit abzuschließen, überträgt ED zuerst die Inhalte des Edit-Puffers, danach den weiteren Text von der Originaldatei in die temporäre DOCUMENT.$$$. Schließlich wird die Original-Datei DOCUMENT.MAY in DOCUMENT.BAK und DOCUMENT.$$$ in DOCUMENT.MAY umbenannt.

Da die Bewegung durch den vollständigen Text erfordern kann, daß Sie Textteile in den und aus dem Edit-Puffer verschieben, bewegen Sie sich grundsätzlich vorwärts durch den Text (das Kommando H erlaubt Ihnen, vom Kopf der Datei an erneut zu beginnen). Das wird später ausführlicher erklärt werden. Studieren Sie die ED-Kommandos auf den folgenden Seiten, um zu lernen, wie man die beschriebenen Aufgaben ausführt.

ED-Kommandos

Um mit ED zu arbeiten rufen Sie es auf und geben auf seine Anforderung (*) hin das korrekte Kommando und ein <cr> ein, wie folgt:

```
A>ED d:filename.typ<cr>
*edcommand<cr>
```

Wir werden in dieser Sektion verschiedentlich Abkürzungen benutzen. Wenn entweder ein Plus- oder ein Minus-Zeichen getippt werden kann, werden Sie das Symbol „~“ finden; wenn Sie weder „+“ noch „–“ tippen, nimmt ED an, daß Sie „+“ meinen. Wenn eine Zahl getippt werden kann, um weitere Information an das Kommando zu geben, werden Sie das Symbol „n“ sehen. Dieses „n“ kann jede dezimale Ganzzahl zwischen 0 und 65535 sein. Verwenden Sie keinen Punkt zwischen den Ziffern. Wenn Sie die Zahl auslassen, nimmt ED den Wert 1 an. Eine Null (0) hat bei einigen Kommandos eine spezielle Bedeutung. Wenn Sie ein Nummernzeichen (#) tippen, setzt ED 65535.

Für ED ist eine Textzeile eine Sequenz von 0 oder mehr Zeichen, gefolgt von einem CARRIAGE RETURN- und einem LINE FEED-Zeichen. Wir stellen dieses Zeichenpaar durch das Symbol CRLF dar. Wenn Sie für ED eine Textzeile eingeben und dann CARRIAGE RETURN, fügt ED das Zeichen LINE FEED hinzu, so daß die Zeile mit CRLF endet.

Wir können ED-Kommandos in vier Funktionen untergliedern: Text übertragen, im Edit-Puffer damit arbeiten, ihn suchen und ändern, Kommandos kombinieren.

Textübertragung

Sie können Text zeilen- oder blockweise übertragen

- von der Originaldatei in den Edit-Puffer
- vom Edit-Puffer in die temporäre Datei
- von der Originaldatei in die temporäre Datei
- vom Edit-Puffer in eine dritte Datei
- von einer dritten Datei in den Edit-Puffer.

Arbeit im Edit-Puffer

Ein imaginärer „character pointer" (CP) (= Zeichenvektor) lokalisiert den Text im Edit-Puffer. Dies sind die Arbeitsweisen:

- Einfügen von Text in den Edit-Puffer
- Manipulation von Text im Edit-Puffer bezogen auf den CP
- Dirigieren der Bewegung des CP.

Textsuche und -änderung

Mit dieser ED-Funktion können Sie im Edit-Puffer arbeiten, um

- eine bestimmte Reihe von Zeichen zu suchen
- eine Reihe von Zeichen zu ersetzen
- den Platz zwischen Zeichenketten auszutauschen
- während der Suche den Text automatisch von der Originaldatei über den Edit-Puffer zur temporären Datei zu übertragen.

Kombination von Kommandos

Eine Kette von ED-Kommandos kann in einer Kommandozeile, die mit <cr> endet, eingegeben werden, um sequentielle Edit-Funktionen zu variieren und eine Kommandokette mehrmals zu wiederholen.

Kommandos zur Textübertragung

#A

Hänge (= append) (kopiere) eine Anzahl von Zeilen von der Originaldatei an den Edit-Puffer an.

Um eine einzelne Zeile zu übertragen, geben Sie nur ein „A" ein; 1 ist der Standardwert für Zeilenübertragungen. Um die maximale Zahl von Zeilen in

den Edit-Puffer zu übertragen, geben Sie „# A" ein. Das Kommando „# A" überträgt Zeilen, bis die Originaldatei erschöpft oder der Edit-Puffer voll ist. Um einen Teil des Textes in den Edit-Puffer zu übertragen, geben Sie „0A" ein; dies überträgt Text von der Originaldatei, bis der Edit-Puffer mindestens halb voll ist. Einmal angehängt, werden Zeilen in der Originaldatei von nachfolgenden Kommandos, die aus der Originaldatei lesen, ignoriert.

nW

Schreibe eine Anzahl von Zeilen aus dem Edit-Puffer in eine temporäre Datei.

W schreibt eine einzelne Zeile

#W schreibt den ganzen Puffer

0W schreibt, bis der Edit-Puffer wenigstens halb leer ist

„W" beginnt immer mit der ersten Zeile im Edit-Puffer und schreibt in die temporäre Datei hinter die letzte Zeile. Nach dem Schreiben werden die Zeilen im Edit-Puffer gelöscht.

E

Ende der Edit-Prozedur.

Der Text ist übertragen, und die Dateien werden folgendermaßen umbenannt:

1. Der übrige Text im Edit-Puffer wird, wie im Kommando W, in die temporäre Datei übertragen.
2. Aller übrige Text in der Originaldatei wird an die temporäre Datei angehängt.
3. Der Typ der Originaldatei wird „.BAK".
4. Der Typ der temporären Datei wird zum ehemaligen Typen der Originaldatei.
5. Die Block-Übertragungs-Datei X$$$$$$$.LIB wird, falls vorhanden, gelöscht.
6. Die CP/M-Anforderung erscheint.

Benutzen Sie das Kommando E als das normale Ende einer Edit-Prozedur. Es muß das einzige Kommando auf einer Zeile sein.

H

Gehe zum Beginn der editierten Datei (Gehe zum Kopf). Der Text ist übertragen, die Dateien wurden folgendermaßen umbenannt:

1. Aller übrige Text im Edit-Puffer wird, wie im Kommando W, in die temporäre Datei übertragen.
2. Aller übrige Text in der Originaldatei wird in die temporäre Datei übertragen.
3. Der Typ der Originaldatei wird „.BAK".

4. Der Typ der temporären Datei wird zum ehemaligen Typen der Originaldatei.
5. Eine neue, leere temporäre Datei wird kreiert.
6. Sie können jetzt die neue Originaldatei editieren.

Das Kommando H hat zwei Anwendungen:

1. Da sich die Kommandos A, N und W nur nach vorne, nicht rückwärts durch die ursprünglichen und temporären Dateien bewegen können, benutzen Sie das Kommando H, um die Editierung soweit zu sichern und kehren für eine weitere Editierung zum Beginn der Datei zurück.
2. Benutzen Sie das Kommando H während der Editierung kritischer Dateien alle paar Minuten, um Ihre Arbeit auf der Diskette zu sichern. Text im Edit-Puffer ist generell verloren, wenn ein Bedienungsfehler oder eine Gerätestörung vorkommt, aber in der temporären Datei gesicherter Text ist leicht wiederzuerlangen.

„H" muß das einzige Kommando auf einer Zeile sein. Der häufige Gebrauch von „H" ist besonders bei Einführung von sehr viel Text oder bei sehr vielen Änderungen wichtig. Bedenken Sie, daß „H" eine neue Backup-Datei erstellt. Dies bedeutet, daß nach zweimaligem Gebrauch des Komandos H die Originaldatei, wie sie vor dem Beginn der Edit-Prozedur war, verloren ist. Deshalb benutzen Sie PIP, um Ihre eigene zweite Kopie der Originaldatei zu machen (nennen Sie sie nicht „.BAK", nennen Sie sie „.OLD" – oder sichern Sie sie auf ein anderes Laufwerk), wenn Sie eine mehrfache Anwendung von „H" erwarten.

O

Lösche die editierte Datei. Der Text wird wie folgt übertragen:

1. Die Inhalte des Edit-Puffers und der temporären Datei werden zerstört.
2. Sie gehen an den Anfang der Originaldatei.

ED fragt, bevor es fortfährt, „O-(Y/N)?", da bei der Ausführung des Kommandos O alle Textänderungen gelöscht werden. Drücken Sie „Y", um zur Originaldatei zurückzukehren, oder „N", um die Editierung fortzusetzen.

„O" muß das einzige Kommando auf der Zeile sein.

Q

Höre mit der Editierung auf; ändere nichts.

ED fragt, bevor es fortfährt, „Q-(Y/N)?", denn alle Textänderungen werden bei Ausführung des Kommandos Q gelöscht. „Y" steht für Beendigung, „N" für Fortführung der Edit-Prozesses.

„Q" muß das einzige Kommando auf der Zeile sein.

R <cr>

Liest eine durch das Kommando X erstellte Datei in den Edit-Puffer.

„R" fügt den gesamten Inhalt der Übertragungs-Datei X$$$$$$$.LIB

direkt hinter den Zeichen-Vektor (CP) in den Edit-Puffer ein. Die Übertragungsdatei wird nicht verändert; Sie können sie während einer Editierungsprozedur lesen, sooft sie wollen. Die Übertragungsdatei wird automatisch gelöscht, wenn Sie mit einem Kommando E, Q oder C aus ED aussteigen.

Rfilename <cr>

Liest die Bibliotheks-Datei filename.LIB in den Edit-Puffer.

Dieses Kommando fügt den gesamten Inhalt der spezifizierten Datei direkt hinter dem Zeichen-Vektor (CP) in den Edit-Puffer ein. Die Bibliotheksdatei wird nicht berührt.

nX

Schreibt (Xfer = Übertragungs-) Zeilen vom Edit-Puffer in eine temporäre Datei namens X$$$$$$$.LIB.

Sie können diese Zeilen später durch das Kommando R<cr> in den Edit-Puffer zurückübertragen. Durch die Kommando-Kombination X, R<cr> können Sie Information blockweise übertragen. Die Datei X$$$$$$$.LIB wird gelöscht, wenn das Edit mit E, Q oder C abgeschlossen wird. Das Kommando X kopiert „n" Zeilen nach dem Zeichenvektor im Edit-Puffer in die Übertragungsdatei. Diese Zeilen werden im Edit-Puffer nicht zerstört, sie werden an jeden beliebigen vorherigen Inhalt der Übertragungsdatei angehängt.

Kommandos zur Arbeit im Edit-Puffer

~B

Setzt den Zeichen-Vektor auf den Anfang (tippen Sie ein „+" oder nichts vor dem „B") oder auf das Ende (tippen Sie ein „–" vor dem „B") des Edit-Puffers.

Beachten Sie, daß das Vorzeichen des Kommandos B die gegenüber allen anderen Kommandos umgekehrte Richtung anzeigt.

~nC

Versetze den Zeichen-Vektor um plus oder minus „n" Stellen.

Die CRLF-Zeichenkombination, die in ED eine Zeile beendet, belegt zwei Stellen. Um den Zeichen-Vektor fünf Stellen weiter zu setzen, tippen Sie „+5C".

~nD

Löscht n Zeichen direkt vor (–) oder nach (+) dem Zeichen-Vektor.

```
100: NOW IS THIME FOR ACTION          vorher
                ^

100:*-4D<cr>                          Kommando

100: NOW IS THE TIME FOR ACTION       nachher
                ^
```

(^ steht für den Zeichen-Vektor.)

I <cr>

Einfügungs-Modus. Ed nimmt alle getippen Zeichen an und fügt sie hinter CP in den Edit-Puffer. Mit gewissen Ausnahmen gelangt jedes getippte Zeichen in den Puffer, bis Sie CONTROL-Z drücken. ^Z beendet den Einfügungs-Modus und kehrt zur ED-Anforderung zurück. Wenn Sie „I" als Großbuchstaben eingeben, wird automatisch aller eingegebene Text in Großbuchstaben übersetzt.

Es gibt einige Zeichen, die im Insert-Modus anders funktionieren, als Sie vielleicht erwarten:

^H oder BACKSPACE vernichtet bei den gegenwärtigen Versionen des CP/M-80 und CP/M-86 das zuletzt getippte Zeichen

^L fügt ein CRLF ein

^M oder RETURN fügt ein CRLF ein

^R zeigt nochmals die laufende Zeile an

^U vernichtet die laufende Zeile

^X vernichtet die laufende Zeile

RUBOUT oder DEL vernichtet das zuletzt getippte Zeichen und zeigt es an.

Die „laufende Zeile" besteht aus den nach dem letzten <cr> im Insert-Modus getippen Zeichen.

Benutzen Sie den Insert-Modus, um eine neue Datei einzugeben oder einer existierenden Datei Zeilen hinzuzufügen. Vergessen Sie nicht, gelegentlich den Insert-Modus zu verlassen und mit H zu sichern.

Istring^Z

Füge eine Zeichenkette (string) ein.

Die Zeichenkette, String, wird hinter dem Zeichen-Vektor in den Edit-Puffer eingefügt. Der String darf nicht länger als der Rest der Kommandozeile sein.

Istring <cr>

Füge eine Zeile ein.

Dieses Kommando fügt die Zeichenfolge hinter dem Zeichen-Vektor in den Edit-Puffer ein. Darauf folgt die Kombination CRLF. Das CARRIAGE RETURN/LINE FEED bewirkt hier anderes als beim vorigen Kommando. Der String wird als separate Zeile eingefügt.

~nK

Eliminiert die Zeichen aus dem Edit-Puffer, die Sie in der Endversion nicht haben wollen.

Durch das Kommando nK bezeichnete Zeilen werden nicht in die Temporärdatei übertragen, verbleiben jedoch in der Originaldatei, die ja später die Backup-Datei wird. Sie können aus dem Edit-Puffer jede beliebige Anzahl von Zeilen in beiden Richtungen entfernen. „+K" oder „K" löscht alle Zeilen

hinter CP inklusive dem CRLF der laufenden Zeile. „–K" löscht außer dem letzten CRLF alle Zeichen der Zeile vor dem Zeichenvektor. Sollten nach Ausführung des Kommandos noch Zeichen in der Zeile vorhanden sein, prüfen Sie, ob der Zeichenvektor an der richtigen Stelle stand.

~nL

Bewegt den Zeichenvektor im Puffer um „n" Zeilen vorwärts oder rückwärts.

Der Zeichenvektor wird auf den Beginn der laufenden oder nächsten Zeile gesetzt, wenn er nicht darauf steht. Nachfolgende Bewegungen umfassen ganze Zeilen; der Zeichenvektor wird um „n" Zeilen vorwärts oder rückwärts bewegt. 0L bewegt den Zeichenvektor zum Beginn der laufenden Zeile.

~nP

Bewegt den Zeichenvektor um „n" Seiten und zeigt anschließend die nächste an.

„P" ist eine gebräuchliche Methode, um Text im Edit-Puffer durchzublättern. Eine Seite ist eine festgelegte Zeilenanzahl, die in den Versionen von CP/M variiert, aber normalerweise einem Bildschirminhalt entspricht. „+nP" schreitet auf dem Edit-Puffer vorwärts und „–nP" rückwärts. „P" bewegt den CP um eine Seite vorwärts oder rückwärts und zeigt die auf den Zeichenvektor folgende Seite an. „0P" listet die Seite, ohne den CP zu bewegen.

~nT

Zeilenanzeige

Um die letzten drei Zeilen vor dem Zeichenvektor zu listen, würden Sie „–3T" angeben. Für die nachfolgenden drei Zeilen würden Sie „+3T" oder nur „3T" tippen. Wenn der Zeichenvektor am Anfang einer Zeile steht, können Sie sie mit „T" anzeigen. Steht er innerhalb einer Zeile, tippen Sie „0T", um den Teil davor und „T", um den Teil danach zu sehen; mit „0TT" sehen Sie die ganze Zeile.

Das Kommando T bewegt den Zeichen-Vektor nicht; es listet nur Textzeilen aus seiner Umgebung.

~U

Übersetzt Klein- in Großbuchstaben.

„+U" beginnt, „–U" beendet die Konvertierung.

Es wird entweder von der Tastatur oder von der Original-Datei übersetzt.

Gewöhnlich übersetzt ED keine Zeichen. Nach „U" oder „+U" übersetzt ED Kleinbuchstaben, die von der Tastatur oder der Original-Datei in den Edit-Puffer gelangen, in Großbuchstaben. Diese Übersetzung wird durch „–U" oder den Abschluß der Edit-Prozedur beendet.

Anmerkung: Sie können mit PIP[L] eine Datei nur aus Kleinbuchstaben erstellen.

~V

„V" oder „+V" zeigen Zeilennummern an.

„–V" unterdrückt diese Anzeige. Dadurch kann man Text bezeichnen und den Zeichenvektor versetzen. („V" steht für „verify line numbers".)

0V

Eine Spezialfunktion des Kommandos V zur Anzeige des übrigen Speicherplatzes.

```
*ØV⟨cr⟩
337Ø6/33719
*
```

Mit der ersten Zahl meldet ED den verfügbaren Speicherplatz im Edit-Puffer, mit der zweiten dessen maximale Ausdehnung. In diesem Beispiel stellen wir nach der Subtraktion des zweiten vom ersten Wert fest, daß im Puffer noch 13 Stellen übrig sind.

n:

CP wird auf Zeilennummer „n" gesetzt.

Wenn Zeilennummern ausgegeben werden, (siehe ~V), können Sie mit diesem Kommando den Speichervektor direkt zum Anfang einer bestimmten Zeile setzen. Z. B. wird ED durch 75:0P eine Textseite ab Zeile 75 listen.

:m

Bearbeite den Bereich vom Zeichenvektor bis zur Zeile „m".

Dies ist eigentlich kein eigenständiges Kommando, sondern ein Praefix.

```
55: *75T⟨cr⟩
```

wird ED veranlassen, vom Zeichenvektor (in diesem Fall auf Zeile 55) an bis zu Zeile 75 zu listen (T).

Beachten Sie, daß Sie dieses Kommando zusammen mit dem vorigen benutzen können, um zeilenweise eine Operation mit einem spezifizierten Textbereich durchzuführen. Wenn Sie z. B. die Zeilen 20 bis 30 vernichten wollen, gleichgültig wo der Zeichenvektor steht, tippen Sie die Kommandozeile

```
87: *20::3ØK⟨cr⟩
```

~n

Bewege dich gemäß Vorzeichen vorwärts oder zurück und gib eine Zeile aus.

Dies ist eine abgekürzte Form des Kommandos ~nLT. Das einfachste Beispiel wäre ein <cr>:

```
11: *⟨cr⟩
12: LINE OF TEXT
12: *
```

„LT" würde das gleiche bewirken.

Kommandos zur Textsuche und -änderung

In diesem Abschnitt über Textsuche und -änderung gilt folgende Regel: Sie müssen Kommandos in Kleinbuchstaben für die Suche nach Kleinbuchstaben benutzen. Durch ein Kommando in Großbuchstaben wird ED nur nach einer Zeichenkette in Großbuchstaben suchen.

nFstring^Z

Finde einen bestimmten eindeutigen String.

Wenn die Suche erfolgreich verlaufen ist (die Zeichenkette wurde n mal gefunden), wird der Zeichenvektor direkt hinter die n-te Folge gesetzt. Nehmen wir an, Sie würden den ersten Refrain „Row, Row, Row Your Boat" mit dem Kommando 3fROW <cr> durchsuchen, so würde der CP wie im folgenden Beispiel stehen. Wäre die Suche allerdings erfolglos, so würde der Zeichenvektor nicht bewegt.

```
*Row, Row, Row Your Boat          vorher
 ^
*3fRow<cr>                        Kommando
*Row, Row, Row Your Boat          nachher
             ^
```

Die Suche beginnt an der Stelle, auf die der Zeichenvektor zeigt. Das Kommando F sucht auch nur im Edit-Puffer; es kann weder in der Original- noch in der bereits auf Diskette gesicherten Datei suchen. Dafür benötigen Sie das anschließend beschriebene Kommando N.

nNstring^Z

Suche im Edit-Puffer und auf der Diskette nach der Zeichenfolge.

Dieses Kommando lädt automatisch den nächsten Teil der Originaldatei in den Edit-Puffer und schreibt von dort, wenn nötig, zeilenweise in die Temporär-Datei. Während „F" nur im Edit-Puffer nach der Zeichenkette sucht, bearbeitet das Kommando N (sogenanntes Autoscan) das gesamte Dokument vom Zeichenvektor an, selbst wenn Teile desselben sich noch in der Originaldatei befinden. Positionieren Sie CP also korrekt.

Benutzen Sie ^L, um in der Zeichenkette ein CRLF zu repräsentieren.

nSfindstring^Zreplacestring^Z

Tausche Information im Edit-Puffer.

Das Substitute-Kommando findet einen String und ersetzt ihn durch einen anderen. Wie gezeigt, würde das Kommando nach „findstring" suchen und es mit „replacestring" ersetzen. Die Suche beginnt beim Zeichenvektor und endet mit dem letzten Zeichen im Edit-Puffer. Der Austausch wird n mal durchgeführt.

Wie in allen Substitutions-Kommandos stellen Sie sicher, daß die Zeichenfolgen, die Sie ersetzen wollen, eindeutig sind.

Benutzen Sie ^L zur Darstellung eines CRLF in der Zeichenkette.

nJfindstring^Zinsertstring^Zendstring^Z

Stelle zwei oder mehr eindeutige Zeichenfolgen einmal oder mehrfach gegeneinander.

Das Kommando kombiniert die Operationen der Einfügung und der Vernichtung.

Es arbeitet folgendermaßen: Zuerst finde „findstring". Direkt danach füge „insertstring" ein. Dann vernichte alle Zeichen am Ende von „insertstring" bis zum Anfang von „endstring". Dies führe n mal aus. Wenn die Operation erfolgreich verlaufen ist, wird der Zeichenvektor auf das Ende des letzten „insertstring" gesetzt.

Z. B. könnten die folgenden Zeilen in einem Dokument erscheinen:

```
*WHEN IN ROME DO AS THE ROMANS DO,
*AND BE ROMANTIC
```

Der Gebrauch des Juxtaposition-Kommandos „JROM ^ZDON'T^Z AS^Z" hätte das folgende Ergebnis:

```
*WHEN IN ROME DON'T AS THE ROMANS DO,
*AND BE ROMANTIC
```

Seien Sie vorsichtig in der mehrfachen Anwendung dieses Kommandos: Sie könnten manchmal Informationen anders ändern, als Sie es wollten.

Kombination von Kommandos

Man kann mit ED Kommandos zusammenfassen, um Zeit zu sparen, und die Zeile dann mit einem CARRIAGE RETURN beenden, z. B.:

```
1: *ØA<cr>
1: *B<cr>
1: *T<cr>
1: LINE 1
1: *
```

läßt sich wie folgt eingeben:

```
1: *ØABT<cr>
1: LINE 1
1: *
```

Hier sind ein paar einfache Regeln, wenn Sie mehrere Kommandos auf eine Zeile schreiben:

1. Die Kommandos E, H, O und Q müssen allein in einer Kommandozeile stehen. Dies wurde so eingerichtet, um die verheerenden Folgen von Tippfehlern bei diesen Kommandos zu vermeiden.
2. Bei Kommandos mit Zeichenketten benutzen Sie zur Beendigung ^Z statt <cr>. Diese Kommandos sind F, I, J,. N und S. Benutzen Sie <cr> lediglich am Ende der Kommandozeile.

Für häufig benutzte Kommandos können Sie eine Kommandofolge erstellen. Das Format ist

*nMcommand1command2command3 <cr>

Nehmen wir als Beispiel

*MFROM⟨Z-3DIRAM⟨ZOTT⟨cr⟩

Die Kommandozeile bewirkt die wiederholte Durchführung der folgenden Schritte:

1. finde ROM (der Zeichenvektor wird hinter das M gesetzt)
2. Vernichte die drei vorherigen Zeichen (ROM)
3. Füge RAM ein
4. Zeige jede geänderte Zeile an (0TT).

Dieses Makrokommando wird jedes auftauchende ROM in RAM verwandeln. Würden wir es beim Beispiel im Kommando J verwenden, so wäre die resultierende Zeile: WHEN IN RAME DO AS THE RAMANS DO, AND BE RAMANTIC . . . Wenn „n" nicht angegeben oder gleich 0 oder 1 ist, wird die Kommando-Sequenz durchgeführt, bis eine Fehlerbedingung auftritt. Das Berühren irgendeiner Taste bricht die Ausführung des Makro-Kommandos ab.

DUMP – Anzeige von Dateiinhalten

DUMP präsentiert Dateiinhalte in hexadezimaler Form.

DUMP arbeitet wie das bereits besprochene Kommando TYPE. Es listet die Datei nicht im ASCII-, sondern im hexadezimalen Format. Assembler-Programmierer benutzen DUMP, um die Inhalte von Nicht-ASCII-Daten (wie Programme oder Binärdaten) zu prüfen.

DUMP d:filename.typ <cr>
Zeigt die Hex-Repräsentation jedes in der Datei filename.typ auf Laufwerk d: gespeicherten Bytes an. Es können mehrdeutige Dateinamen angegeben werden.

```
A⟩DUMP B:PROGRAM.COM⟨cr⟩
ØØ ØØ 3A Ø7 ØØ FE CB DA AC Ø3 21 ØØ ØØ 39 22 25 Ø7 31
ØØ 1Ø ØØ CB 3E 11 D3 FD 21 27 Ø7 7D D3 FD 7C D3 FD CD
ØØ 2Ø 3B Ø2 11 13 Ø4 CD 28 Ø2 CD 38 Ø2 11 55 Ø4 CD 28
A⟩
```

Die ersten vier Ziffern am Beginn jeder Zeile stellen die relative Adresse des ersten Bytes der Zeile dar. Die hexadezimalen Ziffernpaare repräsentieren jeweils ein Byte der gespeicherten Datei.

Spezielle Eingabe

^S unterbricht DUMP zeitweilig, jede andere Taste endgültig.

Stapelverarbeitungs-Dienstprogramme

Ihr Computer ist weit eleganter als die Mammutgeräte der 60er Jahre. Nahezu alle Computermodelle haben in erster Linie im Batch-Modus gearbeitet. Ein Batch (= Stapel) ist eine Gruppe von Dingen; im Falle des Computers ist es eine Folge von Kommandos oder Daten.

Die heute gebräuchlichen Mikrocomputer sind interaktive Maschinen. Sie geben etwas ein, der Computer antwortet, Sie geben noch etwas ein, der Computer antwortet wieder u.s.w. Interaktiv-Prozesse sind kleinen Buchhaltungsprogrammen, Textverarbeitung und anderen Aufgaben, für die Mikrocomputer verwendet werden, angemessen.

Jedenfalls ist es manchmal sinnvoll, eine Gruppe von Kommandos so anzugeben, daß sie ohne Ihre Gegenwart nacheinander ausgeführt werden. Diese Funktionen werden in CP/M durch die Kommandos SUBMIT und XSUB wahrgenommen.

SUBMIT – Kommandozeilen Automatik

SUBMIT dirigiert den sequentiellen Einstieg in die Ausführung einer Anzahl von CP/M-Kommandos ohne zusätzlichen Benutzereingriff. Um SUBMIT zu gebrauchen, müssen Sie zuerst mit dem CP/M- oder einem anderen Editor eine Datei des Typs „.SUB" erstellen. Diese Datei sollte die Liste der CP/M-Kommandos in der Reihenfolge enthalten, in der sie ausgeführt werden sollen, und zwar jeweils eins pro Zeile.

SUBMIT kann nach jeder Programmausführung mit den Programmen BACKUP oder DISKCOPY verbunden werden. Zuerst erstellen Sie eine SUBMIT-Datei namens ORREPENT.SUB mit nur zwei Kommandos.

```
RUN YOURPROG.RAM⟨cr⟩
BACKUP⟨cr⟩
```

(Das RUN YOURPROG.RAM kann jedes gültige CP/M-Kommando sein, und für BACKUP sollten Sie den Namen Ihres Kopierprogrammes einsetzen).

Danach lassen Sie YOURPROG.RAM durch Eingabe von SUBMIT ORREPENT<cr> laufen; wenn Sie versuchen, zu CP/M zurückzukehren, wird SUBMIT Sie zunächst zur Durchführung des BACKUP-Programmes, also zur Duplizierung Ihrer Diskette, auffordern.

Programmentwickler benutzen bei der Erstellung häufiger einen Compiler als einen Interpreter. Dies erfordert allerdings einen zusätzlichen Prozeß: die aktuelle Kompilierung. In kleineren Buchhaltungspaketen können manchmal fünf, zehn oder mehr Programm-Module miteinander verbunden sein, und sie

alle müssen kompiliert werden. Dieser Prozeß kann eine Stunde oder länger dauern; SUBMIT führt diese Funktion aus und befreit den Programmierer davon, jede Kompilierung einzeln anzuweisen. SUBMIT-Sequenzen können durch Anschluß einer SUBMIT-Kommandozeile als letzte in einer „.SUB"-Datei gekettet werden.

SUBMIT filename<cr>

Erstellt auf dem derzeitigen Standardlaufwerk eine Datei namens $$$.SUB, die die in der Datei „filename" enthaltenen Kommandos aufnimmt und führt diese von $$$.SUB anstatt von der Tastatur aus.

> **Anmerkung:** CP/M erwartet die Datei $$$.SUB in jedem Fall auf Laufwerk A:; wenn Sie also gerade mit einem anderen Standardlaufwerk arbeiten, wird SUBMIT nicht funktionieren. Digital Research stellt für dieses Problem eine Korrektur zur Verfügung.

Angenommen, Sie haben eine Datei namens NUCLEAR.SUB mit folgendem Inhalt erstellt:

```
STAT  *.BAS<cr>
ERA   *.BAS<cr>
DIR   *.BAS<cr>
```

so wird der Dialog mit der Konsole bei Ausführung etwa so aussehen:

```
A>SUBMIT NUCLEAR<cr>
A>STAT *.BAS

RECS BYTS EX D:FILENAME.TYP

2   4K    1  A:PORGY.BAS
4   8K    1  A:PORKY.BAS
8   17K   2  A:PORTLY.BAS
BYTES REMAINING ON A: 151K

A>ERA  *.BAS
A>DIR  *.BAS
NO FILE

A>
```

Die dem SUBMIT folgenden Kommandos sind nicht unterstrichen, denn sie werden von CP/M angezeigt, nicht von Ihnen getippt. Sie tippen nur SUBMIT NUCLEAR<cr>.

Spezielle Eingabe

Kommandos aus SUBMIT -Dateien können direkt vor der nächsten Anforderung durch das Drücken einer beliebigen Taste in der Ausführung unterbrochen werden.

SUBMIT filename A B C <cr>

Erstellt eine Datei $$$.SUB mit den in der Datei „filename" aufgelisteten Kommandos und führt diese von $$$.SUB gesteuert aus. Diese Form des SUBMIT-Kommandos unterscheidet sich von der vorher beschriebenen dadurch, daß Sie unvollständige Kommandozeilen in der Datei „filename.SUB" ergänzen können. SUBMIT fügt die ursprünglich fehlende Information durch die Parameter „A", „B", „C" usw. ein. Diese Parameter können Dateinamen oder andere benötigte Informationen sein. In der Ausgangsdatei sollten sie mit den Symbolen $1, $2, $3 usw. angegeben werden, um dann bei der Ausführung ersetzt werden zu können. Bis zu neun Parameter sind erlaubt.

Wenn Sie beispielsweise eine Datei auf Laufwerk „A" editieren und sie nach „B" kopieren wollen, sollten Sie nach jeder Edit-Prozedur den verfügbaren Plattenspeicher prüfen. Ohne SUBMIT würden Sie im Laufe Ihrer Arbeit folgende Kommandozeilen eingeben (der zusätzliche Dialog wurde zugunsten der Klarheit ausgelassen):

```
A>ED MYFILE.DOC<cr>
A>PIP B:=A:MYFILE.DOC[V]<cr>
A>STAT B:MYFILE.*<cr>
```

Wenn Sie also diese Methode bei jeder editierten Datei und nicht nur bei MYFILE.DOC anwenden wollen, können Sie dies durch SUBMIT tun. Zuerst erstellen Sie eine Datei „WORKON.SUB" mit den unten aufgeführten unvollständigen Kommandos:

```
ED $1.$2<cr>
PIP B:=A:$1.$2[v]<cr>
STAT B:$1.*<cr>
```

Zur Benutzung dieser „.SUB"-Datei tippen Sie nun:

```
A>SUBMIT WORKON MYFILE DOC<cr>
```

SUBMIT wird aus WORKON.SUB $$$.SUB erstellen, wobei es für den ersten Parameter $1 MYFILE, für den zweiten Parameter $2 DOC einsetzen wird. SUBMIT initiiert dann einen Warmstart und CP/M sucht $$$.SUB. Danach führt es die dort aufgeführten Kommandos aus.

XSUB – Automatische Benutzer-Eingabe
Eine Untermenge von SUBMIT

Es ist möglich, mehr als nur Kommandos in eine „.SUB"-Datei einzugliedern. Sie kann tatsächlich auf Fragen antworten, die ein Programm stellen könnte, oder andere Variable für den SUBMIT-Aufruf enthalten. Dies geschieht durch das Kommando XSUB. Es ist für die Versionen 1.3 und 1.4 des CP/M-80 und auch für das CP/M-86 nicht verfügbar.

In der SUBMIT-Datei muß XSUB vor dem Programmnamen und der Programmantwort stehen; es kann als Kommando nicht auf eine CP/M-Anforderung hin getippt werden, sondern nur in einer „SUB"-Datei für SUBMIT stehen.

Wenn ein SUBMIT das Kommando XSUB einschließt, wird ein spezieller Set von Instruktionen in den Kernspeicherbereich für CP/M geladen. Wenn dann ein Programm Konsoleninformation anfordert, wird sie aus der SUBMIT-Datei gegeben.

Wie bereits besprochen, können Sie $1, $2 usw. zur Angabe letzter Informationen an SUBMIT benutzen, das diese, wenn nötig, an XSUB weitergibt. Diese Symbole werden bei Ausführung durch die im aktuellen Kommando SUBMIT getippen Parameter ersetzt.

XSUB wird am häufigsten für Programm- und System-Entwicklungen gebraucht. Auch wird sich gelegentlich herausstellen, daß gekaufte Programme ein XSUB für einen Teil ihrer Eingabe benutzen. Ebenfalls findet es bei automatischen Backup-Systemen, die wir bereits besprochen haben, Anwendung. Wenn z. B. die Datei &BEFREE.SUB die folgenden Zeilen enthält:

```
RUN YOURPROG.RAM<cr>
XSUB<cr>
BACKUP<cr>
A<cr>
B<cr>
```

wird folgendes passieren, wenn Sie diese Datei mit SUBMIT verarbeiten:

```
A>SUBMIT &BEFREE<cr>
A>RUN YOURPROG.RAM<cr>
    .
    .   Programm läuft bis zum Ende
    .
A>XSUB<cr>
(XSUB ACTIVE)                    Meldung von CP/M
A>BACKUP<cr>
DRIVE FOR SOURCE DISKETTE? A<cr>

DRIVE FOR DESTINATION DISKETTE? B<cr>

PUT BLANK DISKETTE IN B AND PRESS RETURN WHEN READY<cr>
```

In diesem Beispiel geschieht alles automatisch durch den Computer von dem Zeitpunkt an, an dem Sie das Original-SUBMIT-Kommando eingegeben haben, bis Sie aufgefordert wurden, eine leere Diskette in Laufwerk B: einzulegen und RETURN zu drücken. XSUB übermittelt A<cr> und B<cr> als Antwort auf die Frage des Backup-Programmes.

Die überlagernde Auswirkung beim Gebrauch von SUBMIT zusammen mit XSUB ist die, daß der Datenverarbeitungs-Prozeß bis zu dem Punkt

standardisiert werden kann, daß Aktionen genau in der angegebenen Reihenfolge wiederholt werden können, ohne daß in diesem Zeitraum ein Benutzer anwesend sein muß. Dies ist für manche Computer-Aufgaben sinnvoll, bei anderen wiederum nicht, besonders wenn unvorhersehbare oder wechselnde Eingaben gemacht werden müssen.

Fehlermeldungen

Wir haben die Beschreibung der Fehlermeldungen bis zum Ende dieses Kapitels hinausgezögert, da viele genereller Natur und für verschiedene Programme gleich sind. Tabelle 3-1 listet sie auf.

Jedes dieser CP/M-Programme hat aber auch spezifische Fehlermeldungen, die in Tabelle 3-2 mit einigen Anmerkungen zur Behebung kurz aufgeführt werden.

Generell gibt das System Fehlermeldungen aus, weil es von Ihnen eine Korrektur erwartet oder weil CP/M den Lauf aufgrund des Fehlers abbrechen mußte.

Die meisten Fehlermeldungen von CP/M sind zwar kurz, aber klar genug, um das Problem aufzuzeigen. Einige Programme, die Sie von anderen Firmen als Digital Research erhalten können, benutzen gelegentlich Nummern wie „Error # 1", um die Art der Fehlermeldung anzuzeigen. Das geschieht, um Speicherplatz zu sparen, aber Sie müssen dann im Handbuch nachschlagen.

Wenn Sie eine Fehlermeldung nicht verstehen, versuchen Sie, sich dem Grund mit Hilfe der Logik zu nähern. Wenn Sie dennoch nicht herausfinden, was als nächstes zu tun ist, fragen Sie Ihren Computer-Händler. Sollte er Ihnen auch nicht weiterhelfen können, wenden Sie sich an den Hersteller der Software, und erklären Sie genau, was Sie taten und wie der Computer reagiert hat.

Tabelle 3–1. Generelle Fehlermeldungen

Fehlermeldung	Was sie bedeutet und was zu tun ist
command?	Wenn das getippte Kommando von CP/M wiederholt und mit einem Fragezeichen versehen wird, konnte CP/M es weder als eingebautes Kommando erkennen noch eine Datei dieses Namens mit dem Typ „.COM" (CP/M-80) bzw. „.CMD" (CP/M-86) finden. Prüfen Sie Ihre Eingabe sorgfältig. Wenn Sie wegen des Kommandos im Zweifel sind, benutzen Sie DIR*.COM bzw. DIR*.CMD, um das Inhaltsverzeichnis der Kommandodateien auf der Diskette einzusehen.
BDOS ERR ON d:	BDOS steht für „Basic Disk Operating System" (= Basis-Plattenbetriebssystem), ERR für „error" (= Fehler). Diese Meldung erscheint, wenn CP/M eine Arbeit auf einer Diskette auszuführen versucht und nicht vollenden kann. In erster Linie existieren drei Arten von BDOS-Fehlern.
BAD SECTOR	Die Meldung „BDOS ERR ON d:. A BAD SECTOR" zeigt an, daß Sie entweder eine unformatierte Diskette benutzen, daß auf dieser Diskette ein Sektor unlesbar ist oder daß Sie möglicherweise die Diskette umgekehrt ins Laufwerk gelegt haben.
SELECT	Die Meldung „BDOS ERR ON d:. SELECT" bedeutet, daß CP/M das angegebene Laufwerk nicht finden konnte. Entweder haben Sie ein nicht vorhandenes Laufwerk angegeben, es war nicht eingeschaltet oder Sie haben die Tür des Laufwerks nicht geschlossen.
R/O	Die Meldung „BDOS ERR ON d:. R/O" sagt, daß Sie entweder Disketten ausgewechselt haben, ohne es CP/M mitzuteilen, oder daß die Diskette schreibgeschützt ist. Wenn Sie RETURN drücken, versucht CP/M, den Vorgang zu wiederholen; wenn jedoch der Fehler nicht behoben wurde, werden Sie wahrscheinlich eine neue Fehlermeldung erhalten. Durch ^C bewirken Sie einen Neustart (reboot); das ist die normale Methode, auf einen BDOS-Fehler zu reagieren.

Tabelle 3–2. CP/M-Programm-Fehlermeldungen

Programm	Fehlermeldung	Kommentar
STAT	** ABORTED **	Das STAT-Kommando wurde abgebrochen.
	Bad Delimiter	Falsche Namensabgrenzung; prüfen Sie, wo der Punkt steht.
	Invalid Assignment	Sie können diese Einheit nicht in der angegebenen Weise zuordnen.
	Invalid File Indicator	Sie können mit den Dateinamen nicht in der angegebenen Weise arbeiten.
	TOO MANY FILES	STAT hat eine Obergrenze in Bezug darauf, wieviele Dateien es sortieren kann; versuchen Sie, Wildcard-Referenzen zu benutzen.
	Invalid Disk Assignment	Sie können die Platte so nicht zuordnen.
	Wrong CP/M Version	STAT ist abhängig von der CP/M-Version. Sie können deshalb nicht ein STAT gemäß 1.4 für ein CP/M 2.2 verwenden.
PIP	DISK READ ERROR	Lesefehler.
	DISK WRITE ERROR	Schreibfehler.
	VERIFY ERROR	Dr. Kildall hat diese Meldung nie ohne einen gleichzeitigen BDOS-Error erhalten.
	NOT A CHARACTER SINK	Sie können hier keine Zeichen senden.
	READER STOPPED	Der Leser ist ausgeschaltet.
	NOT A CHARACTER SOURCE	Sie können hier keine Zeichen empfangen.
	ABORTED	Arbeitsabbruch.
	BAD PARAMETER	Der/die [ ] Parameter ist/sind unkorrekt.
	INVALID USER NUMBER	Falsche Benutzernummer.
	RECORD TOO LONG	Satz ist zu lang.
	INVALID DIGIT	Hex-Format paßt nicht.
	END OF FILE, CTL-Z?	PIP nimmt an, das Dateiende gefunden zu haben, wünscht aber eine Bestätigung durch CONTROL-Z.
	CHECKSUM ERROR	Die Prüfnummer im Intel-Format paßte nicht zum gelesenen Satz.
	CORRECT ERROR	Bitte um Korrektur.
	INVALID FORMAT	Format ungültig

Tabelle 3–2. CP/M-Programm-Fehlermeldungen (Fortsetzung)

Programm	Fehlermeldung	Kommentar
	NO DIRECTORY SPACE	Kein Platz im Inhaltsverzeichnis, es kann nicht weitergemacht werden.
	NO FILE	PIP konnte die angegebene Datei nicht finden.
	START NOT FOUND	Die Anfangs-Zeichenkette konnte bis Dateiende nicht gefunden werden.
	QUIT NOT FOUND	Die End-Zeichenkette konnte bis Dateiende nicht gefunden werden.
	CANNOT CLOSE FILE	Prüfen Sie die Diskette.
	DESTINATION IS R/O	Die Empfangsdatei ist schreibgeschützt. Drücken Sie „Y“, um sie zu vernichten und stattdessen die neue Datei zu schreiben.
	NOT DELETED	Versichern Sie sich, daß Sie als Antwort auf die vorherige Meldung „N“ eingegeben haben.
	NOT FOUND	Wie NO FILE
	REQUIRES CP/M 2.0	Die PIP-Version muß zur CP/M-Version passen.
	UNRECOGNIZED DESTINATION	Die angegebene Zieleinheit existiert nicht.
	CANNOT WRITE	Kann nicht schreiben.
	INVALID PIP FORMAT	Prüfen Sie die Zeichensetzung.
	CANNOT READ	Kann nicht lesen.
	INVALID SEPARATOR	Prüfen Sie die Zeichensetzung.
SUBMIT	Error on line	Fehlerhafte Kommandozeile in einer SUB-Datei.
	No SUB file present	SUBMIT konnte die „SUB-Datei“ nicht finden.
	Disk Write Error	Damit SUBMIT korrekt arbeiten kann, darf die Diskette nicht schreibgeschützt sein und sie muß genug Platz für $$$.SUB übrig haben.
	Command Buffer Overflow	Kommando zu lang.
	Command Too Long	Kommando enthält mehr als 127 Zeichen.
	Parameter Error	$1, $2 usw. sind nicht korrekt spezifziert oder passen nicht zu den Anforderungen der „SUB“-Datei.

Tabelle 3–2. CP/M-Programm-Fehlermeldungen (Fortsetzung)

Programm	Fehlermeldung	Kommentar
	Invalid Control Character	Sie haben in Ihrer Datei ein Kontrollzeichen, mit dem SUBMIT nichts anfangen kann.
	Directory Full	Inhaltsverzeichnis voll.
	Cannot Close, R/O?	SUBMIT-Datei kann nicht abgeschlossen werden, ist sie read-only?
XSUB	XSUB Already Present	Sie haben ein unnötiges XSUB-Kommando in der SUB-Datei.
	Requires CP/M 2.0 or Later	XSUB ist für CP/M 1.3 oder 1.4 nicht zulässig.
ED	Break „X" at „C" Das „C" bezieht sich auf das Kommando, das die Fehlermeldung verursachte.	„X" bezieht sich auf eines der sechs Symbole: # Suche den Fehler ? Kommando unbekannt 0 Datei nicht gefunden > Puffer voll E Kommandoausführung abgebrochen F Platte voll.

Kapitel 4
Assembler-Dienstprogramme

Obwohl CP/M so einfach ist, wird der Computerneuling doch eine Menge zu lernen finden – sicher mehr, als die meisten Anwender fordern. Die Information in diesem Kapitel hat für Anwender, die in erster Linie Standard-Software benutzen, nur begrenzten Gebrauchswert. Wenn Sie nicht selbst Assembler-Programme entwickeln oder modifizieren, können Sie dieses Kapitel überschlagen.

Dieses Kapitel behandelt die Progammierhilfen für die CP/M-Assemblersprache: ASM, DDT, LOAD, ASM-86, DDT-86 und GENCMD. ASM und ASM-86 sind Assembler. Sie konvertieren ein in Assemblersprache geschriebenes Quellprogramm in ein Objektprogramm hexadezimalen Formats. Dieses Produkt ist ein Zwischenschritt zu einem wirklichen Programm in Maschinensprache, mit der der Computer arbeitet. Die Endstufe, die Programmdatei in Maschinensprache, wird von LOAD und GENCMD aus der Datei im hexadezimalen Format erzeugt. DDT und DDT-86 dienen der Testhilfe für Maschinenprogramme. Das im vorherigen Kapitel beschriebene Programm DUMP ist eine andere nützliche Assembler-Programmierhilfe. DUMP gibt Dateieninhalte hexadezimal aus.

Assemblersprache

Die Assemblersprache liegt zwei Stufen vor der Ausführung durch den Computer. Sie erinnern sich, daß der Computer die Binärziffern 0 und 1 als Basisdatenelemente benutzt. Diese **Bits** können Daten oder Instruktionen sein, abhängig vom Kontext, in dem die CPU sie empfängt.

Da es umständlich ist, in Zahlenbegriffen wie 10001010 oder 10010010 zu denken, benutzen wir die hexadezimale Notation, um 8 Bits auf einmal darzustellen. Wir nennen diese Gruppen von je 8 Bits Bytes. Solche hexadezimalen Entsprechungen wären z. B. 0C3, 45 oder 0D.

Da auch die hexadezimalen Entsprechungen schwer lesbar sind, vergeben wir Namen für Maschineninstruktionen und die Daten, die sie repräsentieren. Diese Namen sind die Komponenten der **Assemblersprache**. Die Namen, die CPU-Instruktionen repräsentieren, werden auch **Mnemonics** genannt. Für andere Programmelemente werden andere Namen benutzt; sie werden alle in den Sektionen über ASM und ASM-86 beschrieben.

Generell wird jedes Programm in Maschinensprache, das für die CPU 8080 geschrieben wurde, auch auf einer CPU 8085 oder Z80 korrekt laufen. Dieses Buch geht im Bezug auf die Assemblersprache des CP/M-80 lediglich auf die von Intel entworfenen symbolischen 8080-Instruktionen ein, denn diese repräsentieren die für alle Benutzer von CP/M-80 verfügbaren Assemblerinstruktionen. Die Assembler-Mnemonics für die CPU 8085 sind nahezu iden-

tisch, die für die CPU Z80 jedoch völlig verschieden, obwohl sie zumeist in die gleichen Maschineninstruktionen umgesetzt werden. ASM und DDT verstehen nur die 8080-Mnemonics, die Assembler- und die Maschinensprache.

CP/M-86 wurde für die CPUs 8086 und 8088 von Intel geschrieben. Der Unterschied zwischen diesen beiden „Central Processing Units" (= zentrale Rechnereinheiten) besteht in erster Linie in ihrer Beziehung zu anderen Systemkomponenten (die 8088 hat nur einen 8-Bit-, die 8086 einen 16-Bit-Datenbus [Übertragungskanal]). Unabhängig von der CPU sind jedoch die mnemonischen Symbole sowie die Assembler- und Maschinensprache für CP/M-86 konsistent.

Dieses Kapitel geht davon aus, daß Sie in Assembler programmieren können. Falls Sie es lernen wollen, werden Sie in der Bibliographie in Anhang F ein geeignetes Buch finden.

ASM und ASM-86: Assembliere ein Programm

Beide CP/M-Assembler, ASM und ASM-86, konvertieren Assembler-Quellprogramme in den durch den Computer ausführbaren Objektcode. ASM generiert ihn für die 8080, ASM-86 für die 8086. Jeder Assembler erstellt auf Wunsch auch eine Liste mit jeder Zeile des Quellprogrammes sowie dem korrespondierenden Objektcode. ASM-86 generiert zusätzlich eine „cross-reference table", eine Symboltabelle, die dem Programmierer ein schnelles Auffinden von Routinen in Quellprogrammen erlaubt.

Der Assembler-Quellcode wird über einen Editor in eine Plattendatei getippt. Es kann das ED von CP/M oder auch verfeinertere Editoren wie WordStar, PMATE oder VEDIT benutzt werden.

Kommandozeilen des ASM und ASM-86

Wenn Sie einmal Ihre Quelldatei erhalten oder erstellt haben, rufen Sie ASM oder ASM-86 folgendermaßen auf:

A>ASM filename.opt<cr>	CP/M-80
A>ASM86 filename $option(s)<cr>	CP/M-86

Der Dateiname muß eine gültige CP/M-Datei des Typs „.ASM" bzw. „.A86" sein. Die beiden Assembler erfordern unteschiedliche Typen, weil beide Betriebssysteme auf derselben Platte gespeichert sein können.

Das **opt** spezifiziert eine von drei Möglichkeiten und bezeichnet nicht etwa einen „.OPT"-Dateitypen. Die drei Buchstaben nach dem Punkt repräsentieren:

- das Laufwerk, das die **Quell**datei (Original) „filename.ASM" enthält,
- das Laufwerk, das die **hex**-Datei (assembliertes Programm) „filename.HEX" empfangen soll,

– die Einheit, die die **Druck**datei (Programmliste mit Fehlermeldungen) „filename.PRN". empfangen soll.

Sie können die standardmäßigen CP/M-Laufwerks-Spezifikatoren „A" bis „P" als Optionsbuchstaben angeben, vorausgesetzt, diese Laufwerke sind bei Ihrem Computer vorhanden. Die Optionen „X" und „Z" haben besondere Bedeutungen, sie bezeichnen keine Laufwerke.

Z kann benutzt werden, um die „.HEX"- oder „.PRN-Dateioptionen auszulassen.

X kann benutzt werden, um die „.PRN"-Datei auf Konsole anstatt auf Platte zu listen.

Durch die Eingabe

```
A>ASM BSM.AAZ<cr>
```

weisen Sie den CP/M-80-Assembler an, eine Datei namens BSM.ASM auf Laufwerk „A": in eine hex-Datei namens BSM.HEX, ebenfalls auf Laufwerk „A:", umzusetzen und die Generierung einer Druckdatei zu unterdrücken. Das assemblierte Programm hat immer den Dateitypen „.HEX", die Druckdatei den Typen „.PRN". Sollen sich alle Dateien auf dem derzeitig aktiven Laufwerk befinden, brauchen Sie nach dem Dateinamen weder Punkt noch Option anzugeben. In diesem Fall hätten Sie nach Assemblierung eines Programms BSM-ASM die folgenden Dateien:

BSM.ASM	Quelldatei
BSM.HEX	hex-Datei
BSM.PRN	List-Datei

ASM-86 arbeitet etwas anders. Hier wird ein Punkt mit drei Buchstaben als Dateityp interpretiert. Optionen werden durch ein Leer- mit anschließendem Dollarzeichen spezifiziert.

A~ bezeichnet die Position der Originaldatei

H* bezeichnet die Position des hex-Output

P* bezeichnet die Position des Print-Output

S* bezeichnet die Position der Symboltabelle

F@ spezifiert das Format der hex-Datei.

Jede Option muß von den anderen durch ein Leerzeichen getrennt sein und alle sind wahlfrei. Werden sie nicht angegeben, wird das derzeitig aktive Laufwerk bzw. der Normal-Set angenommen.

Die Zeichen „~", „*" und „@" nach dem obigen Buchstaben stehen jeweils für einen zweiten Buchstaben, nämlich:

~ kann jeder gültige Laufwerksspezifikator sein (A bis P)

* kann jeder gültige Laufwerksspezifikator (A bis P) sein o d e r
X für die Konsoleinheit
Y für die List-Einheit
Z für die Unterdrückung der Ausgabe

@ kann „I" für Intel- oder „D" für Digital Research-Format sein.

Die Optionen hinter dem Dollar-Zeichen können in jeder beliebigen Reihenfolge eingegeben werden. Das Folgende ist eine gültige ASM-86-Kommandozeile:

```
A>ASM 86 FILE-19.86 $FI HB PY SB<cr>
```

Durch dieses Kommando würde FILE-19.86 auf Laufwerk „A:" (Standard) im Intel-hex-Format („FI") nach Laufwerk „B:" („HB") assembliert, die Liste gedruckt („PY") und die Symbol-Tabelle (cross reference) ebenfalls auf Laufwerk („B:") („SB") ausgegeben werden. Danach würden folgende Dateien existieren:

FILE-19.A86	Original
B:FILE-19.H86	assembliertes Programm
B:FILE-19.SYM	Symboltabelle

Die Datei FILE-19.LST würde nicht generiert, sondern sofort auf die Druckeinheit ausgegeben. Beachten Sie, daß die hex-Datei in CP/M-86 den Typen „.H86" anstelle von „.HEX" und die Druckdatei „.LST" anstelle von „.PRN" hat. Auch ist die Standard-Quelldatei vom Typ „A86" anstelle von „.ASM".

Von ASM und ASM-86 benutzte Dateien

Was sollen nun die vielen Dateien? Die „.ASM"- bzw. „.A86"-Dateien enthalten das Assembler-Ursprungsprogramm und werden mit Hilfe von ED oder einem anderen Text-Editor erstellt.

Der Assembler erstellt eine zum Ursprungsprogramm analoge hexadezimale Datei des Typen „.HEX". Ihr Format wird in CP/M-80 als „Intel Hex Format" bezeichnet, denn Intel hat es als erste Firma zur Speicherung hexadezimaler Zeichen auf Lochstreifen verwandt, und das Plattenformat ist äquivalent. Hier ein Beispiel einer kurzen „.HEX"-Datei:

```
:Ø2Ø1ØØØØ3EØØBF
:ØØØØØØØØØØ
```

Hier ist das 8080-Assembler-Programm, das diese Information kreierte:

```
ORG   Ø1ØØh    ;program starts at address Ø1ØØ hex
MVI   A,Ø      ;move a ØØ hex into Register A
END            ;end of program
```

Die List-Datei „SAMPLE.PRN" kombiniert beide Informations-Sets, das Original und das Ergebnis:

```
Ø1ØØ            ORG  Ø1ØØh    ;program starts at address Ø1ØØ hex
Ø1Ø1  3E ØØ     MVI  A,Ø      ;move a ØØ hex into Register A
Ø1Ø2            END           ;end of program
```

Diese Art der Auflistung ist besonders dann nützlich, wenn Sie viele *Labels* und *Symbole* verwenden.

Die „.HEX"-Datei wird also zur Erstellung eines ausführbaren Programms, die „.PRN"-Datei als Testhilfe (Fehlerbereinigung) sowie zur Dokumentation benötigt.

CP/M-86 hat eine andere Namensgebung. Dieses sind äquivalente Dateien:

CP/M-80	CP/M-86	
filename.ASM	filename.A86	Datei mit Ursprungscode
filename.HEX	filename.H86	Datei mit Ursprungscode (hex-Format)
filename.PRN	filename.LST	Datei mit Programmauflistung
	filename.SYM	Datei mit Listung der Symboltabelle

Eine typische Auflistung einer Symboltabelle für ein kurzes 8086-Assembler-Programm könnte wie folgt aussehen:

```
ØØØØ VARIABLES
ØØ8Ø NAMEN ØØ9Ø STAT

ØØØØ NUMBERS
Ø123 SHORTSTACK
ØØØØ LABELS
ØØ73 LAST   ØØ42 MIDDLE   ØØØØ FIRST
```

VARIABLES listet in alphabetischer Reihenfolge jedes vom Assembler bezeichnete Daten-Statement +, wie z. B. „NAMEN DB 0. NUMBERS" jede Gleichsetzung und jede errechnete Anzahl, LABELS jedes Statement-Label. Die Zahlen zu Beginn jeder Zeile bezeichnen die Zeilennummer.

Format von Quellprogrammen

Der Assembler erwartet das Originalprogramm in einer stark spezifizierten Form. Es ist eine Folge von ASCII-kodierten (Text) Anweisungen bzw. Zeilen. Jede Zeile endet mit einem CRLF (CARRIAGE RETURN + LINE FEED).

Statements in Assembler-Sprache

Jedes Assembler-Statement ist aus einem bis zu fünf **Feldern** zusammengesetzt. Ein Feld ist eine Gruppe von Zeichen; und die Felder sind voneiander durch Leer- oder TAB-Zeichen getrennt. Durch Einfügung von TAB-Zeichen

wird ein Quellprogramm lesbarer, da die Felder hier bei den TAB-STOPS beginnen. Diese werden allgemein für die Spalten 1, 9, 17, 25, 33 usw. gesetzt. Im Gebrauch der Felder differieren ASM und ASM-86 nur geringfügig. Das generelle Format eines Assembler-Statements ist

line #	label	mnemonic	operand(s)	;comment	CP/M-80
label:	prefix	mnemonic	operand(s)	;comment	CP/M-86

Zeilenummern (line #)

Die Zeilennummer ist eine wahlfrei anzugebende Ganzzahl zu Beginn einer Assembler-Zeile. Einige Editoren fügen diese Zeilennummern automatisch ein, ED jedoch nicht und ASM ignoriert sie sowieso.

Label (= Marken)

Ein Label ist ein Identifikator, der eine Adresse oder einen Wert repräsentiert. Es kann für ASM bis sechzehn Stellen und für ASM-86 bis zur tatsächlichen Zeilengrenze lang sein. Das erste Zeichen muß ein Buchstabe, die anderen können Buchstaben oder Ziffern sein. Kleinbuchstaben werden wie Großbuchstaben behandelt. Bei ASM kann nach einem Label ein Doppelpunkt folgen, wenn es in einem Label-Feld steht, bei ASM-86 muß er folgen. ASM ignoriert das Zeichen „$", ASM-86 die Zeichen „@" und „—". Das Label muß bei allen Statements angegeben werden, auf die sich Operandenfelder wie z. B. bei EQU oder SET-Direktiven beziehen; ansonsten kann es wahlfrei benutzt werden. Generell darf eine bestimmte Marke nur im Label-Feld eines einzigen Statements erscheinen, in Operanden-Feldern dagegen nahezu beliebig oft.

Es gibt gewisse Schlüsselworte, die in Label-Feldern nicht benutzt werden dürfen, weil sie für ASM bzw. ASM-86 im voraus festgelegte Bedeutungen haben. Es sind:

- Alle 8080- und 8086-Instruktionssymbole
- Alle Direktiven-Namen von ASM und ASM-86
- Alle Register-Namen von 8080 und 8086
- Spezielle Schlüsselworte des ASM-86: BYTE, WORD und DWORD.

Praefix

Verschiedene Instruktionen der 8086, insbesondere die der REP-Familie sowie LOCK, führen eine spezielle Praefix-Aktion in Bezug auf den Rest der Instruktion bzw. des Assembler-Statements durch. Um eine korrekte Arbeitsweise zu gewährleisten, müssen solche Praefixe vor dem mnemonischen Symbol stehen.

Mnemonics

Das mnemonische Feld ist das einzige, das in jedem Statement unbedingt angegeben werden muß. Es enthält entweder eine 8080/8086-Instruktion oder eine Assembler-Direktive. Die meisten im Anhang F aufgeführten Assembler-Bücher enthalten eine Liste der Instruktions-Mnemonics für 8080 und 8086. Sie sind das Herz der Assembler-Sprache; und deshalb ist es sinnvoll, sich mit ihnen vertraut zu machen, bevor man Assembler programmiert.

Operanden

Viele Assembler-Instruktionen erfordern einen oder mehrere Operanden, manche keine. Assembler-Direktiven erfordern sie generell. Operanden können Konstante, Labels oder Ausdrücke sein. Konstante und Ausdrücke werden nachfolgend beschrieben.

Kommentare

Ein Kommentar-Feld beginnt mit einem Semikolon. Es ist wahlfrei und wird vom Assembler ignoriert. Dennoch sollten Sie in Ihre Programme möglichst gründliche Kommentare einfügen, denn sie sind die optimale Dokumentation.

Konstante und Ausdrücke

Ein Operandenfeld einer Assembler-Instruktion oder -Direktive kann ein Label, eine Konstante oder einen Ausdruck beinhalten. Labels wurden bereits, Konstante und Ausdrücke sowie die Regeln zu ihrer Bildung werden nachfolgend beschrieben.

Konstante

Eine **numerische Konstante** ist eine feste Zahl bezogen auf eine von vier Zahlenbasen: binär, oktal, dezimal oder hexadezimal.

Eine **binäre Konstante** ist eine Sequenz der Ziffern 0 und 1 gefolgt vom Buchstaben „B" zur Signifizierung der Basis.

Eine **oktale Konstante** ist eine Folge der Ziffern 0 bis 7 gefolgt von „Q" oder „O".

Normalerweise wird eine **dezimale Konstante** angenommen, der ein „D" folgen kann.

Eine **hexadezimale Konstante** ist eine Sequenz der Ziffern 0 bis 9 und der Buchstaben „A" bis „F", die die Zahlen 10 bis 15 repräsentieren, gefolgt vom Buchstaben „H". Sie muß mit einer Dezimalziffer beginnen; diese Anforderung können Sie erfüllen, indem Sie jeweils eine 0 voransetzen.

Wenn Sie glauben, daß eine Zahl dadurch lesbarer wird, können Sie Dol-

lar-Zeichen einfügen, die dann von ASM ignoriert werden. Die folgenden drei Konstante sind also gleich:

11010101 = 1101$0101 = 11$010$101

Eine Zeichenkonstante wird in Hochkommata (') eingeschlossen. Sie darf für ASM 64 und für ASM-86 255 Zeichen lang sein. Es sind nur druckbare Zeichen erlaubt. Kleinbuchstaben werden nicht in Großbuchstaben konvertiert. Sie können auch ein Apostroph einschließen, indem Sie zwei hintereinander tippen. ASM und ASM-86 bearbeiten den Wert einer Zeichenkette, indem sie dem 7-Bit-ASCII-Code eine führende 0 hinzufügen.

Ausdrücke

Für viele Instruktionen können Sie auch Ausdrücke als Operanden einsetzen. Ein **Ausdruck** ist eine Kombination von Konstanten, Marken, arithmetischen Operatoren, logischen Operatoren und Klammern. Während einer Assemblierung wird jeder Ausdruck berechnet und auf einen einzigen Wert reduziert.

Arithmetische Operatoren

Wie man sieht, kann der Assembler einfache Arithmetik durchführen, um den Wert eines Ausdrucks zu errechnen. Folgende Operatoren können benutzt werden, um Label und Konstante zu Ausdrücken zu verbinden:

A+B
ist die Summe von „A" und „B"

A−B
ist die Differenz zwischen „A" und „B"

+B
ist „B"

−B
ist 0 minus B

A*B
ist das Resultat von „A" mal „B" (ohne Vorzeichen)

A/B
ist der Quotient von „A" geteilt durch „B" (ohne Vorzeichen)

A MOD B
ist der Divisionsrest

A SHL B
ist „A", um „B" Bit-Positonen nach links geshiftet (= verschoben); die führenden Bits werden vernachlässigt und die Freistellen rechts mit Nullen aufgefüllt

A SHR B
Rechts-Shift; funktioniert umgekehrt wie „A SHL B".

Der Assembler führt diese Operationen mit 16-Bit-Werten aus, der Maximalwert ist also 2^{16}.

Logische Operatoren

Der Assembler kennt Boolersche Algebra mit folgenden Operatoren:

NOT B
das Bit-Komplement von „B"

A AND B
logische Bit-Verknüpfung von „A" und (AND) „B"

A OR B
logische Bit-Verknüpfung von „A" oder (OR) „B"

A XOR B
logische Bit-Verknüpfung von „A" und „B" durch Exklusiv-Oder (Die Bedingung ist erfüllt (Resultat = 1 (wahr)), wenn eins oder das andere 1 ist, aber nicht beide.)

Logische Operationen werden mit 16-bit-Werten (Worten) durchgeführt.

Andere Operatoren

Ein anderer häufig benutzter Operator ist das Dollarzeichen ($), dem bei Gebrauch der laufende Wert des Zuordnungszählers zugewiesen wird.

Außerdem erkennt ASM-86 folgende zusätzliche Operationen:

SEG A
ist der Segment-Wert von „A"

OFFSET A
ist die Distanzadresse von „A"

TYPE A
1, 2 oder 4, abhängig davon, ob „A" vom TYPE BYTE, WORD oder DWORD ist

LENGTH A
Längenattribut von A in Bytes

LAST A
ist LENGTH A-1 außer bei LENGTH = 0, wobei LAST A ebenfalls 0 ist

A PTR B
ist die virtuelle Variable des Typs „A" mit den Attributen von „B"

Verhältnis-Operatoren (nur bei 8086)

ASM-86 erlaubt auch folgende Verhältnis-Operatoren:

A EQ B
ist wahr (Ergebnis = 1), wenn „A" gleich „B" ist

A LT B
ist wahr, wenn „A" kleiner als „B" ist

A LE B
ist wahr, wenn „A" kleiner oder gleich „B" ist

A GT B
ist wahr, wenn „A" größer als „B" ist

A GE B
ist wahr, wenn „A" größer oder gleich „B" ist

A NE B
ist wahr, wenn „A" ungleich „B" ist.

Rangordnung von Operatoren

Bei der Bewertung von Ausdrücken mit verschiedenen Operatoren geht der Assembler nicht einfach von links nach rechts vor, sondern behandelt bestimmte Operatoren vorrangig. Diese Hierarchie heißt **Rangordnung** und wird unten aufgezeigt. Die in der ersten Zeile aufgeführten Operatoren werden grundsätzlich zuerst berücksichtigt, wenn sie in einem Ausdruck auftreten, die der letzten Zeile zuletzt. Innerhalb einer gleichen Rangstufe werden Operationen von links nach rechts durchgeführt, wenn sie zusammen in einem Ausdruck auftreten. Die Rangfolge sieht so aus:

$ (höchste Stufe)
die Operation in den innersten Klammern
(8086: SEG, OFFSET, PTR, TYPE, LENGTH, LAST)
* / MOD SHL SHR
− +
(8086: EQ, LT, LE, GT, GE, NE)
NOT
AND
OR XOR (niedrigste Priorität)

Klammern werden benutzt, um die Hierarchie zu ordnen oder den Ausdruck lesbarer zu machen.

Assembler-Direktiven

Sie können in einem Assemblerprogramm eine Anzahl von Spezialinstruktionen verwenden, die nicht in Objektcode umgewandelt werden. Diese sogenannten Direktiven kontrollieren den Prozeß der Assemblierung und beeinflussen insofern den resultierenden Maschinencode nur mittelbar. Solche Assembler-Direktiven werden in nahezu der gleichen Form in das Quellprogramm eingefügt wie normale Statements.

DB, DW und DS – Assembler-Direktiven

Benutzen Sie diese drei Direktiven, um Kernspeicherbereiche zu initialisieren.

DB Definiere Byte.
Initialisiert einen Bereich Byte-weise

DW Definiere Wort.
Initialisiert einen Bereich in Einheiten von jeweils zwei Bytes

DS (8080) Definiere Speicher.
Reserviert einen Bereich eines bestimmten Formats

DD (8086) Definiere Doppelwort.
Initialisiert einen Bereich in Einheiten von jeweils vier Bytes

RS (8086) Definiere Speicherbereich.
Reserviert einen Bereich eines bestimmten Formats

RB (8086) Definiere Speicherbereich.
Ähnlich wie RS

RW (8086) Definiere Wortspeicher.
Reserviert einen Bereich eines bestimmten Formats

Operanden-Ausdrücke im Statement DB werden errechnet und als 8-Bit-Werte aufeinanderfolgend im Kernspeicher gespeichert. Ausdrücke im Statement DW werden errechnet und als 16-Bit-Werte in aufeinanderfolgenden Speicherstellenpaaren gespeichert; innerhalb eines Paares geschieht dieser Vorgang von rechts nach links. Die Ausdrücke in den Direktiven DS, RS, RB und RW werden errechnet und danach die Anzahl der Kernspeicherstellen in 16-Bit-Einheiten reserviert. Diese Bereiche werden nicht verändert.

In diesen Speicherdirektiven kann wahlweise ein Label angegeben werden. In diesem Fall ordnet ASM und ASM-86 ihm den Wert der Adresse des ersten definierten oder reservierten Bytes zu. Bei den Speicherdirektiven DB, DW und DD kann eine beliebige Anzahl von Ausdrücken, jeweils durch Kommata getrennt, angegeben werden, bei DS, RS, RB und RW jedoch nur einer.

ORG, END, EQU, CSEG, DSEG, SSEG und ESEG – Assembler-Direktiven

Eine Anzahl von Direktiven für die Bereichszuordnung stehen für beide, ASM und ASM-86, zur Verfügung.

ORG

Diese Direktive gibt dem Assembler die Kernspeicheradresse für die nachfolgende Sequenz von Statements an. Das Format von ORG ist

```
Label       ORG   Ausdruck     ;Kommentar
```

Der Assembler nimmt den Wert des Ausdrucks als Kernspeicheradresse der

nächsten Instruktion oder Definitionsdirektive an, diese Adresse ist für den 8080-Code fest, für den 8086-Code relativ. Wahlweise können Kommentare eingefügt werden. Es kann mehr als ein ORG in einem Programm gesetzt werden.

END

Die Direktive END teilt dem Assembler mit, daß kein Quell-Statement mehr folgt. Die 8080-Form der Direktive ist

```
Label     END   Ausdruck    ;Kommentar
```

Diese Direktive ist wahlfrei.

Dem wahlfreien Label wird der Wert des Zuordnungszählers im Assembler an diesem Programmpunkt zugeordnet. Der wahlfrei anzugebende Ausdruck wird errechnet und als Startadresse des Programms in der Intel-HEX-Datei angegeben. Wenn kein Ausdruck genannt ist, wird „0000" als Start-Adresse angenommen.

Die 8086-Form der Direktive ist

```
          END              ;Kommentar
```

EQU

Die Direktive EQU (equate = setze gleich) ordnet einem Label Werte oder Ausdrücke zu. Das Format ist:

```
Label     EQU   Ausdruck    ;Kommentar   oder
Label     EQU   Register    ;Kommentar   oder
Label     EQU   Mnemonic    ;Kommentar
```

Der Ausdruck kann jede gültige Zahl, Adresse, Konstante oder eben ein Ausdruck sein, für den 8086-Assembler kann stattdessen auch ein Register oder ein mnemonisches Symbol angegeben werden. Label und Ausdruck sind in EQU erforderlich. Sie können vorher definierte Marken im Ausdruck verwenden.

Eine 8080-Variante von EQU ist die Direktive SET, die im Gegensatz zu EQU erlaubt, einem Label erneut Werte zuzuordnen. Ansonsten ist das Format der Direktive SET gleich.

CSEG, DSEG, ESEG, SSEG

CSEG (code segment), DSEG (data segment), ESEG (extra segment) und SSEG (stack segment = Puffer-Segment) können als 8086-Direktiven spezifiziert werden. Alle 8086-Quellanweisungen müssen einem dieser Segmente zugeordnet werden, damit die CPU sich in korrekter Weise darauf beziehen kann. Instruktions-Anweisungen erfüllen nur in CSEG ihren Zweck. Direktiven-Statements sind in jedem Segment gültig. Statements zur Datenspeicherung sind in den Segmenten DSEG und CSEG gültig.

Diese Formen von Direktiven sind integrierter Bestandteil des 8086-Assemblers. Um sie vernünftig anwenden zu können, sollten Sie zuerst die Arbeits-

weise der 8086-CPU verstehen. Dafür ist ein Nachschlagwerk über die 8086 geeignet.

IF und ENDIF – Assembler-Direktiven

Definieren Sie eine Assembler-Sektion nur dann, wenn die IF-Bedingung wahr ist. Die Form dieser Direktive ist

```
Label     IF      Ausdruck                    ;Kommentar
                    .
                    .
                    .
                  nachfolgende Statements
                    .
                    .
                    .
Label     ENDIF                               ;Kommentar
```

Der Assembler berechnet den Ausdruck im IF-Statement. Ist der Wert Null, werden die Statements zwischen IF und ENDIF ignoriert, anderenfalls umgewandelt.

Der Gebrauch von IF wird als **konditionierte Assemblierung** bezeichnet; sie findet nur statt, wenn (IF) bestimmte Bedingungen erfüllt sind.

IF-Statements können bei der Assembler-Programmierung in unterschiedlichster Weise angewandt werden; hier ist einer der häufigsten Fälle:

```
HAVE$TERMINAL     EQU     ØFFFFh              ;value = true
NO$TERMINAL       EQU     NOT HAVE$TERMINAL   ;value = false
;
                  IF      HAVE$TERMINAL
                            .
                            .
                            .
                          Assembler-Statement,
                          sofern ein Terminal angeschlossen ist
                            .
                            .
                            .
                  ENDIF
                  IF      NO$TERMINAL
                            .
                            .
                            .
```

```
                    Assembler-Statement,
                    sofern kein Terminal angeschlossen ist
                    .
                    .
                    .
ENDIF
```

In diesem 8080-Beispiel verändern Sie das Programm nur durch die Gleichsetzung von HAVE$TERMINAL, je nachdem, ob ein Terminal vorhanden ist oder nicht. Beachten Sie, daß dem Label HAVE$TERMINAL lauter binäre Einsen zugeordnet sind, dem Label NO$TERMINAL das Komplement, lauter Nullen.

Spezielle 8086-Assembler-Direktiven

INCLUDE

Die Direktive INCLUDE ermöglicht einem Programm, ein anderes Assembler-Programm völlig einzuschließen, ohne gegenwärtig dessen Text zu kennen. Die Form ist

INCLUDE filename.typ

Eine Datei, die während einer Assemblierung eingeschlossen wird, darf selbst keine INCLUDE-Direktive enthalten; geschachtelte INCLUDEs sind nicht erlaubt.

TITLE

Die List-Datei druckt die durch TITLE definierte Kopfzeile zu Beginn einer jeden Seite.

TITLE ‚Zeichenkette'

PAGESIZE

Normalerweise werden 66 Zeilen pro Seite gedruckt. Dies kann durch PAGESIZE variiert werden.

PAGESIZE Ausdruck

PAGEWIDTH

Normalerweise druckt eine List-Datei 120 Zeichen pro Zeile oder 79 beim Display-Terminal. Dies kann durch PAGEWIDTH reduziert werden.

PAGEWIDTH Ausdruck

EJECT

EJECT bewirkt einen Seitenvorschub.

EJECT

SIMFORM

Es wird erwartet, daß die Formularvorschubzeichen von der List-Einheit erkannt werden. Ist sie dazu nicht in der Lage, so kann durch SIMFORM die

Stellung des Formularvorschubzeichens aus der Anzahl der Zeilenvorschübe errechnet werden.

SIMFORM

NOLIST

Diese Direktive blockiert die Ausgabe einer List-Datei für die gesamte folgende Assemblierung.

NOLIST

LIST

Diese Direktive hebt die Blockade wieder auf.

Assembler-Fortschrittsmeldungen

Der Assembler gibt bei Start und Ende Meldungen aus. Da die Operationen von ASM und ASM-86 etwas unterschiedlich sind, wollen wir beide getrennt behandeln.

ASM

Nach Eingabe einer ASM-Kommandozeile wird eine Meldung auftauchen wie

```
CP/M ASSEMBLER – VER 2.0
```

Darauf könnten Fehlermeldungen folgen, wenn Anlaß bestand.

Wenn ASM seinen Job beendet hat, gibt es eine dreizeilige Meldung aus:

```
xxxx
yyyH USE FACTOR
END OF ASSEMBLY
```

Diese Meldung kann folgendermaßen interpretiert werden: „xxxx" ist die hexadezimale Adresse der ersten freien Speicherstelle nach dem Programm. „yyyH" verblüfft einigermaßen: es zeigt den Quotienten des benutzten gegenüber dem zur Verfügung stehenden Speicherplatz für eine Symboltabelle an. „yyy" liegt zwischen hex 000 und 0FF. Geteilt durch hex 0FF ergibt sich der prozentuale Anteil, der von der Symboltabelle tatsächlich belegt wird. Wenn also z. B. „yyy" 080 ist, wurde ungefähr eine Hälfte (80/0FF hex = 128/255 dezimal) des Speicherplatzes für die Symboltabelle benutzt. Die Meldung END OF ASSEMBLY bedeutet, daß die Assemblierung beendet wurde; sie bedeutet nicht unbedingt, daß dies erfolgreich geschah.

ASM-86

Analog zur 8080 beginnt ASM-86 die Assemblierung mit einer Meldung:

```
CP/M 8086 ASSEMBLER VER 1.1
```

Während der Assemblierung erscheint:

```
END OF PHASE 1
END OF PHASE 2
```

Diese Meldungen bezeichnen Zeitstufen in der Durchführung von ASM-86. Nach Vollendung erscheint eine Meldung:

```
END OF ASSEMBLY. NUMBER OF ERRORS: Ø
```

Selbst wenn die Fehleranzahl mit 0 angegeben wird, heißt das nicht, daß das Programm **erfolgreich** assembliert wurde, sondern nur, daß keine Kodierfehler entdeckt wurden. ASM-86 kann durch das Drücken jeder Taste unterbrochen werden.

Assembler-Fehlermeldungen

Der Assembler zeigt zwei Arten von Fehlermeldungen an. Die eine führt zum Abbruch der Arbeit (typisch dafür sind Probleme mit Platten oder Dateien), die andere übergeht lediglich ein Ursprungs-Statement, das sich so nicht übersetzen läßt.

Abbruchsbedingungen

NO SOURCE FILE PRESENT oder
NO FILE

Der Assembler konnte den angegebenen Quellcode mit dem Typ „.ASM" bzw. „.A86" nicht finden.

NO DIRECTORY SPACE oder
DIRECTORY FULL

CP/M kann im Inhaltsverzeichnis keine Dateinamen mehr aufnehmen. Löschen Sie überflüssige Dateien auf der Diskette oder nehmen Sie eine neue.

SOURCE FILE NAME ERROR

Formalfehler im Namen der Quelldatei; auch können „*" und „?" nicht benutzt werden.

SOURCE FILE READ ERROR oder
DISK READ ERROR

Datei oder Platte kaputt

OUTPUT FILE WRITE ERROR oder
CANNOT CLOSE oder
CANNOT CLOSE FILES

Die Diskette ist entweder schreibgeschützt oder voll.

SYMBOL TABLE OVERFLOW
Sie benutzen mehr Labels und Symbole in Ihrem Programm als der Assembler verkraften kann.

PARAMETER ERROR
Die dem Dateinamen in der Kommandozeile folgenden Optionen sind nicht korrekt.

Fehlermeldungen im Quell-Programm

Ein zweites Set von Meldungen steht während der Assemblierung zur Verfügung. Wenn keine erscheint, hatte der Assembler nichts zu beanstanden, (das heißt aber nicht, daß das Programm korrekt laufen wird!). Wenn ASM oder ASM-86 etwas nicht verstehen, zeigen sie es in folgendem Format an.

```
a bbbb cccc  label  mnemonic  operand  ;comment
```

„a" ist ein Buchstabe, der den Fehlertyp repräsentiert. „bbbb" ist die hexadezimale Adresse des fehlerhaften Statements und „cccc" die hexadezimale Repräsentation des vom Assembler erzeugten Maschinen-Codes. Dieser wird bei einem Fehler auf Null gesetzt.

ASM zeigt folgende Fehlertypen an:

D Data error (= Datenfehler).
Der Wert eines Ausdrucks paßt nicht zum angegebenen Datenbereich.

E Expression error (= Ausdrucksfehler).
Der Ausdruck ist falsch formuliert oder vielleicht auch zu komplex.

L Label error (= Markenfehler).
Sie haben ein Label falsch verwendet, es vielleicht für mehrere Statements vergeben.

N Feature not implemented (= für diese Auslegungsvariation nicht gültig).
Digital Research hat auch einen Assembler, unter dem Namen MAC bekannt, der mit Direktiven arbeitet, mit denen ASM nichts anfangen kann. In so einem Fall gibt ASM die Fehlermeldung „N" aus.

O Overflow error (= Ausdrucks-Überlauf).
Ihr Ausdruck ist zu kompliziert für den Assembler. Unterteilen Sie ihn oder reduzieren Sie die Anzahl seiner Operatoren.

P Phase error (= Phasenfehler).
Ein Label ändert während einer Assemblierung seinen Wert. Wenn Sie diesen neu zuordnen müssen, benutzen Sie SET. Vielleicht haben Sie auch ein Label doppelt angegeben.

R Register error (= Registerfehler).
Sie haben ein ungültiges Register angegeben. „POP B" wäre ein gültiges Assembler-Statement, nicht aber „POP A". Dieses würde eine „R"-Fehlermeldung hervorrufen.

S Syntax error (= Syntaxfehler).
Dies ist eine sehr häufig vorkommende Meldung, die durch ein ungültiges mnemonisches Symbol oder einen typographischen Fehler ausgelöst werden kann.

U Undefined symbol (= undefiniertes Symbol).
Sie haben in einem Ausdruck ein Label benutzt, ohne diesem einen Wert zuzuordnen. Wenn z. B. ONE im Programm nicht definiert ist, wird die Zeile etwa folgendermaßen gelistet werden:

```
U 0102 06 00        MVI     B,ONE
```

V Value error (= falscher Wert).
Der Operand (oder Ausdruck) wurde nicht korrekt angegeben. Dies geschieht gewöhnlich, wenn Sie fehlerhaft eintippen, ein Komma vergessen oder der Assembler einen bestimmten Buchstaben erwartet.

ASM-86 benutzt für seine Meldungen ein Nummernsystem, das ähnlich wie das von ASM funktioniert, aber meistens vollständiger und offensichtlicher in der Bedeutung ist. ASM-86 präsentiert die folgenden Fehlermeldungen:

0 Ungültiges erstes Item.
Der erste Teil der Anweisungszeile ist weder ein gültiges Label noch ein Mnemonic.

1 Fehlende Pseudo-Instruktion.

2 Ungültige Pseudo-Instruktion.

3 Mehrfach definierte Variable.

4 Mehrfach definiertes Label.

5 Undefinierte Instruktion.

6 Fehler am Zeilenende – ignoriert.
Tritt gewöhnlich auf, wenn in eine Datei Kontrollzeichen eingebettet sind.

7 Operand(en) nicht zur Instruktion passend.

8 Ungültige Operanden für eine Instruktion.

9 Instruktion fehlt.

10 Undefiniertes Element in einem Ausdruck.

11 Ungültiger Pseudo-Operand.

12 Geschachteltes IF ungültig – IF ignoriert.
Sie haben ein IF-Statement in einen Abschnitt des Codes eingebettet, der bereits als Teil einer IF-Direktive erkannt wurde.

13 Ungültiger Operand für IF – IF ignoriert.

14 Kein zu einem ENDIF gehörendes IF.

15 Auf ein Symbol wurde sich fälschlich im voraus bezogen – übergangen.

16 Mehrfach definiertes Symbol – wurde als undefiniert behandelt.

17 Instruktion außerhalb eines Codesegments.

18 Syntaxfehler bei einem Dateinamen.

19 Geschachteltes INCLUDE nicht erlaubt.

20 Ungültiges Element in einem Ausdruck.

21 Fehlende TYPE-Information für Operand(en).
BYTE, WORD oder DWORD muß vorher zugeordnet werden.

22 Label außerhalb Reichweite.

23 In einem Operanden fehlt eine Segmentinformation.

24 Kodierfehler im Makroaufbau.

DDT – Dynamic Debugging Tool
Dynamische Testhilfe

Mit DDT werden Programme in Maschinensprache getestet und ihre Fehler entfernt. CP/M-80 arbeitet mit DDT, CP/M-86 mit DDT-86. Sie können DDT wie folgt benutzen:

- Um ein assembliertes Programm in den Kernspeicher zu laden.
- Um simple Änderungen an einem Maschinenprogramm vorzunehmen.
- Um Fehler in Maschinenprogrammen lokalisieren zu helfen.
- Um Ihre Software zu korrigieren oder aufzufrischen.
- Um spezielle Betriebs-Routinen zu installieren (Programme, die eine periphere Einheit wie z. B. einen Drucker steuern).
- Um in CP/M-80 Version 2.2. oder CP/M-86 Plattenparameter zu ändern.
- Um Kernspeicherinhalte zu prüfen und zu modifizieren.
- Um zeilenweise Assembler-Codes einzugeben.
- Um eine Programmsektion zu reassemblieren (z. B. Maschinen- in Assembler-Instruktionen zurückzuversetzen).
- Um Inhalte der internen CPU-Register zu prüfen und zu modifizieren.
- Um Unterbrechungspunkte zu setzen – Stellen, an denen das Programm gestoppt werden soll, um zu sehen, was bis dahin passiert ist.
- Um die Ausführung eines Programmes schrittweise aufzuzeichnen – nachzuverfolgen, was im Kernspeicher und den internen CPU-Registern geschieht.

Die Eingabe von

DDT⟨cr⟩ CP/M-80

oder

DDT86⟨cr⟩ CP/M-86

lädt das Dynamische Testhilfeprogramm in den Kernspeicher Ihres Computers, wo es auf weitere Instruktionen wartet.

DDT d:filename.typ⟨cr⟩ CP/M-80

oder

DDT86 d:filename⟨cr⟩ CP/M-86

lädt DDT und außerdem die bezeichnete Datei zur Prüfung, Modifikation oder Ausweitung in den Kernspeicher. Der Typ muß „.COM" oder „.HEX" für CP/M-80 sein; DDT-86 erwartet einen Datei-Typ „.CMD".

DDT bzw. DDT-86 geben als Anforderung (Zeichen, daß ein Kommando erwartet wird) einen Bindestrich aus.

DDT-Kommandos

Einmal in den Kernspeicher geladen (u. U. mit der wahlfrei anzugebenden Datei) ist DDT bereit, Kommandos anzunehmen. Ein kurzer Überblick darüber wird in Tabelle 4-1 gegeben.

Die meisten DDT-Kommandos brauchen für ihre Anwendung zusätzliche Information. Diese besteht häufig aus einer Kernspeicheradresse oder einem hexadezimalen Wert. DDT und DDT-86 erwarten leicht unterschiedliche Formate für die Angabe einer Kernspeicherstelle, denn CP/M-80 arbeitet nur mit 16-Bit-, CP/M-86 dagegen mit 20-Bit-Adressen.

CP/M-80
0000 bis FFFF sind gültige Adressen

CP/M-86
0000:0000 bis F000:FFFF sind gültige Adressen. Die Zahl vor dem Doppelpunkt ist die 16-Bit-Segmentnummer; die Zahl hinter dem Doppelpunkt ist die 16-Bit-Adresse innerhalb des spezifizierten Segments. Wenn Sie nicht sicher sind, was ein Kernspeichersegment ist, schlagen Sie es in dem Abschnitt über CP/M-86 in Kapitel 7 nach oder sehen Sie in einem technischen Handbuch über das Design der 8086-CPU nach. Sie brauchen die Segmentnummer nicht anzugeben – sie ist nur nötig, um die 16-Bit-Adresse zu spezifizieren. Außerdem können Sie den Namen eines der vier 8086-Segmentregister als die Zahl vor dem Doppelpunkt einsetzen und DDT-86 wird die richtige Adressverschiebung ermitteln.

Tabelle 4–1. DDT-Kommandos und -Funktionen

Getippter Buchstabe		Funktion
CP/M-80 (DDT)	**CP/M-86 (DDT-86)**	
A	A	Assemble instructions (Assembliere Instruktionen)
D	D	Display memory (Zeige Kernspeicherinhalt an)
	E	Load program (Lade Programm)
F	F	Fill memory (Fülle Kernspeicher)
G	G	Go – execute program (Führe Programm aus)
H	H	Hexadecimal arithmetics (Hex-Arithmetik)
I	I	Set up FCB (file control block) (Richte Datenkontrollblock ein)
L	L	List instructions (Liste Instruktionen)
M	M	Move memory (Übertrage Kernspeicher)
R	R	Read disk file (Lies Plattendatei)
S	S	Set memory to value (Setze Kernspeicherinhalt auf Wert)
T	T	Trace program (Zeichne Programmausführung schrittweise auf)
U	U	Execute partially (Führe teilweise aus)
	V	Show disk file read (Zeige gelesene Plattendatei)
	W	Write contents to disk (Schreibe Inhalte auf Platte)
X	X	Examine/modify registers (Prüfe und modifiziere Register)

Eingabe von Assembler-Instruktionen

A# <cr>

Gib von der spezifizierten Adresse an Assembler-Instruktionen ein. Zwischen dem Kommandobuchstaben (A) und der spezifizierten Adresse darf kein Leerzeichen liegen. Nach dem CARRIAGE RETURN zeigt DDT die Kernspeicheradresse an und wartet auf die Eingabe eines für 8080 bzw. 8086 gültigen mnemonischen Symbols oder eines Operanden. Mnemonic und Operand sollten durch ein Leerzeichen getrennt sein und jede Instruktion wird

durch <cr> beendet. Nach Eingabe einer Instruktion wird die als nächstes verfügbare Adresse angezeigt und Sie können in gleicher Weise fortfahren, Assembler-Instruktionen einzugeben. Um diesen Vorgang abzuschließen, tippen Sie einen Punkt gefolgt von einem CARRIAGE RETURN oder auch nur ein <cr>.

```
-AØ1ØØ<cr>
Ø1ØØ MOV A,C<cr>
Ø1Ø1 .<cr>
```

Wenn DDT oder DDT-86 Ihre Instruktion nicht verstehen, setzen sie ein Fragezeichen und zeigen die letzte aufzufüllende Adresse an.

Zeige Kernspeicherinhalt an

D <cr>

Es existieren drei Formen des Kommandos „Display memory". Die erste ist einfach der Buchstabe „D" gefolgt von einem CARRIAGE RETURN. Er weist DDT an, den Kernspeicherinhalt vom laufenden Zuordnungszähler an zu listen. Bei den meisten Systemen werden jeweils insgesamt 192 Bytes (12 Reihen à 16 Bytes) angezeigt. Einige Systeme, insbesondere solche mit einer Bildschirmzeilenbreite von weniger als 80 Zeichen, werden möglicherweise weniger Zeichen ausgeben, aber es besteht kein funktioneller Unterschied zu dem „Display memory"-Kommando.

Der Kernspeicher wird mit der ersten Adresse für eine Zeile zur linken, gefolgt von den 16 hexadezimalen Byte-Repräsentationen, die im Kernspeicher an dieser Adresse beginnen, und den entsprechenden 16 ASCII-Zeichen ganz rechts (hier stehen Punkte für nicht-druckbare Zeichen) gelistet.

```
-D<cr>
Ø1ØØ 3A Ø7 ØØ FE C8 DA AC Ø3 21 ØØ ØØ 39 22 25 Ø7 31 :.......!..9"%.1
Ø11Ø ØØ C8 3E 11 D3 FD 21 27 Ø7 7D D3 FD 7C D3 FD CD ..>...!'.}..|...
Ø12Ø 3B Ø2 11 13 Ø4 CD 28 Ø2 CD 3B Ø2 11 55 Ø4 CD 28 ;.....(.:...U..(
Ø13Ø Ø2 1E Ø7 ØE Ø2 CD Ø5 ØØ ØE Ø1 CD Ø5 ØØ FE 1B CA ................
Ø14Ø A1 Ø3 FE ØD C2 28 Ø1 11 91 Ø4 CD 28 Ø2 3E 4D 32 .....(.....(.>M2
Ø15Ø 2A Ø7 21 1D Ø7 36 ØB 7E D6 ØD D3 FF 3E Ø7 32 1E *.!..6....~.....>.2.
Ø16Ø Ø7 21 2A Ø7 35 3E Ø1 32 2B Ø7 11 AE F9 CD AB Ø2 .!*.5>.2+........
Ø17Ø 3E Ø4 32 2Ø Ø7 CD 65 Ø2 21 27 Ø7 36 21 CD 2E Ø2 >.2 ..e.!'.6!....
Ø18Ø FA D6 Ø2 CD C5 Ø2 C2 6A Ø1 21 1E Ø7 35 C2 61 Ø1 .........j.!..5.a
Ø19Ø 3E Ø2 32 24 Ø7 CD 65 Ø2 3A 2A Ø7 C6 Ø7 32 2A Ø7 >.2$..e.:*...2*.
Ø1AØ 36 Ø7 21 2A Ø7 35 3E Ø2 32 21 Ø7 CD 65 Ø2 21 27 6.!*.5..2!..e.!'
Ø1BØ 07 36 32 CD 2E Ø2 FA E9 Ø2 3E Ø1 32 28 Ø7 11 AE .62.......>.2+...
```

Jedes nachfolgende „D" gibt die nächsten 192 Kernspeicher-Bytes aus. Die Startadresse für die erste Anwendung hängt, wenn angegeben, von der mit DDT geladenen Datei ab. Übliche Startwerte sind 0100 hex und 0000 hex.

D# <cr>

Die zweite Form des Kommandos „Display memory" erlaubt die Spezifizierung einer Startadresse. Wenn sie nicht durch 16 teilbar ist (letzte Ziffer 0), wird die erste Zeile des Displays keine 16 Stellen anzeigen. Wenn wir z. B. D501 tippen würden, würde die erste Zeile der Anzeige nur 15 Bytes enthalten. Jedes folgende „D" ohne Adresse setzt die Anzeige nach dem letzten „D#" fort.

```
-D5ØØ<cr>
Ø5ØØ 52 52 4F 52 2Ø 2Ø 2Ø 44 52 3E 42 24 52 45 41 44 RROR    DR=B$READ
Ø51Ø 2Ø 45 52 52 4F 52 2Ø 2Ø 2Ø 2Ø 44 52 3D 42 24 56 ERROR      DR=B$V
Ø52Ø 45 52 49 46 59 2Ø 45 52 52 4F 52 2Ø 2Ø 42 4C 4F ERIFY ERROR  BLO
Ø53Ø 43 4B 3D 2D 2D 24 2Ø 2Ø 54 52 4B 3D 2D 2D 2Ø 2Ø CK=--$  TRK=--
Ø54Ø 53 43 54 52 3D 2D 2D 24 2Ø 2Ø 53 54 53 57 44 3D SCTR=--$  STSWD=
Ø55Ø 2D 2D 2D 2D 2D 2D 2D 2D 2Ø 2Ø 24 AE Ø5 C1 Ø5 D5 --------  $.....
Ø56Ø Ø5 E5 Ø5 FS Ø5 Ø8 Ø6 1A Ø6 9B Ø5 2F Ø6 3F Ø6 58 ............/.?.×
Ø57Ø Ø6 9B Ø5 9B Ø5 9B Ø5 9B Ø5 9B Ø5 71 Ø6 83 Ø6 95 ..........q....
Ø58Ø Ø6 A7 Ø6 BE Ø6 D4 Ø6 EB Ø6 FF Ø6 9B Ø5 9B Ø5 9B ...............
Ø59Ø Ø5 9B Ø5 9B Ø5 9B Ø5 9B Ø5 9B Ø5 4E 4F 2Ø 45 52 ..........NO ER
Ø5AØ 52 4F 52 2Ø 4D 53 47 2Ø 46 4F 55 4E 44 24 44 52 ROR MSG FOUNG$DR
Ø5BØ 49 56 45 2Ø 4E 4F 54 2Ø 4F 5Ø 45 52 41 42 4C 45 IVE NOT OPERABLE
```

D#,# <cr>

Eine dritte Form des Displays bezieht das Tippen zweier Zahlen ein: die Start- sowie die Endadresse des anzuzeigenden Kernspeicherinhalts. Auch hier werden bei Start oder Ende nicht exakt 16 Bytes gelistet, wenn die angegebene Adresse nicht durch 16 teilbar ist. Jedes folgende „D" ohne Adresse wird die Anzeige nach dem letzten „D#,#" fortsetzen.

DDT-86 erlaubt Ihnen, Information in 16-Bit-Werten, normalerweise Kernspeicherworte genannt, auszugeben, nicht wie normal in 8-Bit-Werten. Dazu fügen Sie ein „W" zwischen dem „D" und den Adressparametern ein.

DW<cr>	Zeigt 96 Kernspeicherworte an
DW#<cr>	Zeigt 96 Kernspeicherworte an, beginnend bei „#"
DW#,#<cr>	Zeigt 96 Kernspeicherworte an, beginnend bei der ersten und endend bei der zweiten Nummer.

Fülle den Kernspeicher

F#,#,# <cr>

Das Kommando „Fill memory" erlaubt Ihnen, einen Kernspeicherblock mit einfachen, konstanten Byte-Werten zu füllen. Ein normaler Gebrauch dieser Instruktion wäre, einen Teil des Kernspeichers auf Nullzeichen (00 hex) zu setzen, bevor man diesen Bereich erneut verändert.

Nach dem „F" müssen drei Nummern angegeben werden: die Startadresse des ersten Blocks, die Endadresse des zweiten Blocks und das Füllzeichen.

```
-F0100,01F0,00A0<cr>
-
```

Dieses Kommando würde jede Kernspeicherstelle von 0100 hex bis 01F0 hex auf A0 hex setzen. Wie bei vielen anderen Kommandos auch, können Sie DDT-86 durch Eingabe des Buchstabens „W" direkt hinter dem „F" veranlassen, 16-Bit-Worte anstelle von Bytes zu verarbeiten (vergessen Sie nicht, im dritten Operanden den Wert eines Wortes anstelle eines Bytes anzugeben, sonst können Sie unerwartete Resultate erhalten).

Sie können dieses Kommando als einen groben Kernspeichertest benutzen, um offensichtliche Kernspeicherfehler zu isolieren. Versuchen Sie, Kernspeicher mit „00", „55", „AA" oder „FF" aufzufüllen und anschließend Kommandos einzugeben. Nach jedem Kommando benutzen Sie „D", um sicherzustellen, daß der Kernspeicher tatsächlich die Zeichen empfangen hat, die Sie gesendet haben. Sollte dies einmal nicht der Fall sein, so haben Sie ein Kernspeicherproblem, und Sie sollten die Maschine Ihrem Händler zur Reparatur übergeben. Füllen Sie jedoch besser keine Kernspeicherbereiche, die von CP/M oder DDT genutzt werden (generell die ersten 256 Bytes und die letzten 6K bis zu 10K).

Go – führe das Programm im angegebenen Bereich aus

G <cr>

Das Kommando „G" weist DDT an, mit der Ausführung von Kernspeicherinstruktionen an der im Programmschrittzähler der CPU enthaltenen Adresse zu beginnen. DDT-86 beginnt die Ausführung an der aus den Registern CS und IP abgeleiteten Adresse.

Sie werden das Kommando „G" bei einem Programmtest selten ohne spezifizierte Endadresse benutzen, denn bei dieser Form des Kommandos hat DDT keine Kontrolle über seine Ausführung. Sie können also nicht stoppen und nach DDT zurückkehren, es sei denn, Sie hätten irgendwo in den ausgeführten Code eine korrekte Instruktion „RST" eingefügt.

G# <cr>

Um die Durchführung an einer anderen als der im laufenden Programmzähler enthaltenen Adresse (bzw. die Register CS und IP für DDT-86) zu beginnen, geben Sie diese direkt nach dem „G" an. Durch diese Form des Kommandos wird die spezifizierte hexadezimale Zahl in den Befehlszähler (oder die Register CS und IP) überstellt, und die CPU beginnt dort mit der Ausführung. Wie auch beim „G"-Kommando gewinnt DDT die Kontrolle nicht zurück.

G#,# <cr>

Um die auszuführenden Instruktionen einzugrenzen, geben Sie nach dem

„G" die die Start- und Endadresse durch ein Komma getrennt ein. Dieses Kommando setzt durch Speicherung eines RST 7 zuerst einen Breakpoint auf die zweite Adresse. Danach beginnt die Ausführung bei der ersten Adresse. Wenn die CPU diesen Breakpoint passiert, wird die Programmausführung gestoppt und DDT fordert ein neues Kommando an.

Sie können durch die Form G#,#,# <cr> noch einen zweiten Breakpoint angeben. Wenn einer der beiden erreicht wird, fordert DDT ein neues Kommando.

G,# <cr>

Sie können bei „G" auch die Start-Adresse auslassen, DDT wird dann den laufenden Befehlszähler (bzw. die Registerwerte von CS und IP) annehmen. Diese Form ist besonders nach einem vorherigen Unterbrechungspunkt nützlich.

Hex-Arithmetik

H#,# <cr>

Das Hexadezimal-Arithmetikkommando berechnet Summe und Differenz der beiden angegebenen Hex-Zahlen. Sie werden zuerst in ihre 16-Bit-Äquivalente konvertiert (sofern Sie nicht mehr als 4 Hex-Ziffern eingeben). Danach wird als erste Zahl die Summe, als zweite die Differenz angezeigt.

Richte Datei-Kontrollblock ein

Ifilename <cr>

Das Kommando „I" identifiziert die Datei, die dann mit „R" in den Kernspeicher geladen werden soll. Der erste dem „I" folgende Dateiname wird in den Standard-FCB (file control block), normalerweise auf Adresse 005C hex, übertragen. Wenn mehr als ein Dateiname (durch Leerzeichen getrennt) angegeben ist, wird der zweite in den zweiten Teil des FCB überstellt. Die Länge des dem „I" folgenden Kommandos wird, gefolgt von dem Kommando sowie einem abschließenden Nullzeichen, auf Adresse 0080 hex placiert.

Kurz: Das Kommando „I" bereitet den Dateikontrollblock vor, so daß Sie mit „R" eine Datei von Platte in den Kernspeicher lesen können. Das Kommando „I" richtet den FCB immer für den Zugriff auf das derzeitig aktive Laufwerk ein. Sie können bei diesem Kommando keine Laufwerkskennzeichnung wie z. B. „B:" einschließen – es würde zu einer Fehlermeldung führen. Sie könnten höchstens den FCB mit „S" modifizieren, so daß das Byte an der Adresse 005C den korrespondierenden Wert des gewünschten Laufwerks anstatt 00 enthält. Wählen Sie einen der Werte aus Tabelle 4–2.

Wenn in DDT-86 eine Datei mit „E" geladen wurde, kopiert DDT-86 die Information des FCB aus der Basis-Page des geladenen Programmes.

Da „I" den Kommandopuffer an 0080 hex genauso wie eine Benutzereingabe belegt, kann es zur Simulation folgender Situation benutzt werden: Auf

eine CP/M-Anforderung wurde ein Kommando eingegeben und danach die Ausführung des Programms mit Hilfe von DDT schrittweise kontrolliert.

Liste den Kernspeicherinhalt

L<cr>

Mit „L" reassemblieren Sie einen Teil des Kernspeichers. „L" listet von der zuletzt angezeigten Adresse (oder 0100 hex, wenn noch nichts gelistet wurde) an die Kernspeicherinhalte in Assembler-Sprache.

```
-L<cr>
Ø151 MOV D,A
Ø152 MVI E,ØØ
Ø154 PUSH D
Ø155 LXI H,Ø2ØØ
Ø158 MOV A,B
Ø159 ORA C
Ø15A JZ Ø165
Ø15D DCX B
Ø15E MOV A,M
Ø15F STAX D
Ø16Ø INX D
-
```

Tabelle 4–2. Werte für Plattenlaufwerks-Spezifikationen im Standard-FCB

Inhalte auf Adresse 005C hex	Ausgewähltes Plattenlaufwerk
00	Derzeitig aktives Laufwerk
01	Laufwerk B
02	Laufwerk C
03	Laufwerk D
04	Laufwerk E
06	Laufwerk F
07	Laufwerk G
08	Laufwerk H
09	Laufwerk I
0A	Laufwerk J
0B	Laufwerk K
0C	Laufwerk L
0D	Laufwerk M
0E	Laufwerk N
0F	Laufwerk O
10	Laufwerk P

Wenn DDT nicht weiß, wie es einen hexadezimalen Wert aus der 8080-Assembler-Sprache wiedergeben soll, wird die Meldung „?? = # #" angezeigt, wobei „# #" das hexadezimale Byte ist.

Jeder nachfolgende Gebrauch von „L" reassembliert die nächsten elf Instruktionen. Operanden, die numerische Werte repräsentieren, werden stets hexadezimal wiedergegeben, Labels und Symbole überhaupt nicht (der ZSID Symbolic Debugger von Digital Research zeigt Labels an, wenn er sie kennt).

L#,# <cr>

Wie bei vielen anderen DDT-Kommandos können Sie für die Instruktion „L" eine Start- und eine Endadresse angeben. Sie können jede der beiden Nummern auslassen, aber das Komma muß da sein, wenn Sie nur die Endadresse angeben.

Übertragen von Kernspeicherinhalt

M#,#,# <cr>

Das Kommando „M" (move memory) benötigt für seine Funktion drei Zahlen. Die erste bezeichnet den ersten zu übertragenden Speicherplatz, die zweite den letzten und die dritte die Adresse des ersten Bytes der neuen Position des Kernspeicherblocks.

Einige Überlegungen in bezug auf das Übertragungskommando sind anzustellen. So sollte normalerweise die Zieladresse nicht innerhalb des zu übertragenden Kernspeicherblocks liegen.

Das Übertragungskommando verschiebt kein Programm, denn ein Programm enthält normalerweise Referenzen zu anderen Adressen innerhalb seiner selbst. Dies wird jedoch vom Move-Kommando nicht beachtet; jedes Byte im Originalblock wird so wie es ist an seine neue Position übertragen. Daher würde also ein Programm außerhalb seines normalen Adressbereiches wahrscheinlich nicht arbeiten.

Wenn nach Eingabe eines „M"-Kommandos ein Fragezeichen erscheint, ist mindestens eine der spezifizierten Zahlen keine gültige hexadezimale Adresse. Auch wenn die Adresse der zuletzt zu übertragenden Speicherstelle niedriger als die Startadresse ist, bewegt sich nichts.

Lies Datei

DDT
R# <cr>

Um eine Datei von einer Platte zu lesen, benutzen Sie zuerst das Kommando „I", um sie zu identifizieren, danach das Kommando „R".

Die wahlfrei anzugebende Zahl ist ein Verschiebefaktor oder Bias (= Distanzadresse), wie die Bezeichnung von Digital Research lautet. Wird sie ausgelassen, wird als Verschiebefaktor 0 angenommen. Er wird zur normalen Ladeadresse addiert und die Datei beginnend bei der resultierenden Adresse geladen.

Jede Adresse, die FFFF hex übersteigt, bewirkt eine Fortführung ab 0000 hex. So würde eine Datei, die bei einem Verschiebefaktor von Null die

Lade-Adresse 8000 hex hätte, bei einem Verschiebefaktor von 8100 hex (R8100 <cr>) auf 0100 hex geladen. Auch darf keine Datei so geladen werden, daß ein Teil von ihr im Kernspeicherbereich zwischen 0000 und 0100 hex liegt oder daß sie DDT überlagert.

Sie sollten keine Datei so laden, daß sie irgendeinen Bereich von CP/M-80 in Anspruch nimmt. Die verbotenen Bereiche sind 0000 bis 0100 hex und der CP/M-80 Anteil unterhalb FFFF hex. Die Startadresse dieses Bereiches kann durch Eingabe des Kommandos L5,7<cr> gefunden werden; die nach dem Mnemonic JMP angezeigte Adresse verweist auf das niedrigste von CP/M und DDT benutzte Byte im oberen Bereich.

Wenn Sie eine „.COM"-Datei mit „I" identifizieren, beginnt der Ladevorgang bei 0100 plus Offset. Bei einem anderen Dateitypen wie z. B. „.HEX", erwartet das Kommando „R" eine Datei im Intel-hex-Format, und es wird den Verschiebefaktor zu den in der hex-Format-Datei enthaltenen Adressen addieren, um die Ladeadressen zu ermitteln.

DDT-86
R filename<cr>

DDT-86 erlaubt keinen Offset. „R" liest die spezifizierte Datei in den Kernspeicher. Die 8086-CPU erlaubt es, daß sich außer DDT-86 noch bis zu sieben Dateien gleichzeitig ohne Überlappung im Kernspeicher befinden. DDT-86 lädt also mit „R" Dateien in den jeweils ersten freien Kernspeicherbereich.

Setze Kernspeicherinhalt

S# <cr>

Mit diesem Kommando läßt sich der Kernspeicherinhalt anzeigen und wahlweise ändern (Set), und zwar von der direkt hinter „S" folgenden Adresse an. In DDT-86 können Sie nach dem „S" ein „W" angeben, um anzuzeigen, daß Worte anstatt Bytes gesetzt werden sollen.

DDT zeigt eine Adresse und den derzeitigen Inhalt an und erwartet dann von Ihnen die Eingabe des Inhaltes eines Bytes (oder Wortes) oder einfach ein CARRIAGE RETURN, um den Inhalt zu belassen.

```
-SØ1ØØ<cr>
Ø1ØØ C3  3D<cr>      C3 wird 3D
Ø1Ø1 28  4F<cr>      28 wird 4F
Ø1Ø2 38  <cr>        Bleibt
Ø1Ø3 C#  <cr>        Ende der Eingabe
```

Ein Fragezeichen erscheint, wenn DDT oder DDT-86 Ihre Eingabe nicht verstehen kann oder eine ungültige hexadezimale Ziffer oder Adresse getippt wurde.

Zeichne Programmausführung schrittweise auf

DDT, DDT-86

T # <cr>

Um eine Programmausführung selektiv aufzuzeichnen, benutzen Sie das „Trace"-Kommando. „Trace" zeigt die CPU-Register in ihrem Zustand direkt vor der Ausführung der nächsten Instruktion des Programmes an. Die dem Kommando „T" folgende Nummer gibt an, wieviele Instruktionen so ausgeführt werden sollen. Das Drücken irgendeiner Taste bewirkt den Abbruch und die Rückkehr zum DDT.

Ein schrittweise aufgezeichnetes Programm läuft ungefähr 500mal langsamer als normal, weil DDT jede Programminstruktion simulieren muß. Das „Trace"-Kommando macht Unterbrechungen möglich. Das kann zum Problem werden, wenn Ihr Programm keine Unterbrechungen zuläßt.

Für CP/M-80 sieht eine „Trace"-Anzeige etwa folgendermaßen aus:

```
CØZØMØEØIØ  A=ØØ  B=ØFBS  D=ØØØØ  H=ØØØØ  S=Ø1ØØ  P=Ø13D LXI SP,Ø2ØØ
```

C0 bedeutet: das Carry-Kennzeichen (Übertragungs-Kennzeichen) ist auf Null gesetzt
Z0 bedeutet: das Null-Kennzeichen (zero-flag) ist auf Null gesetzt
M0 bedeutet: der Wert des Minus-Kennzeichens ist Null
E0 bedeutet: das Even-parity-flag (= Kennzeichen für gerade Anzahl von Bits) ist Null
I0 bedeutet: das Intermediate-carry-flag (= Kennzeichen für Zwischenübertragung) ist Null
A=00 bedeutet: der Inhalt des Registers A ist 00
B=0FB6 bedeutet: der Inhalt des Registerpaares BC ist 0FB6
D=0000 bedeutet: der Inhalt des Registerpaares DE ist 0000
H=0000 bedeutet: der Inhalt des Registerpaares HL ist 0000
S=0100 bedeutet: der Pufferbereichvektor ist 0100
P=013D bedeutet: der Befehlszähler steht auf 013D
LXI SP ,0200 ist die nächste auszuführende Instruktion.

Eine „Trace"-Anzeige für CP/M-86 würde etwa wie folgt aussehen:

```
              AX    BX    CX    DX    SP    BP    SI    DI    IP
---SZAPC  ØØØ3  Ø1ØØ  ØØØØ  ØØØØ  119E  ØØØØ  ØØØØ  ØØØØ  ØØ1C INC   SI
```

Die linke Spalte bezeichnet die Flags, die gesetzt sind. Die nächsten neun Spalten bezeichnen die 8086-Register. Die letzte Spalte bezeichnet die nächste auszuführende Instruktion.

DDT-86 erlaubt während der „Trace"-Operation auch die Ausgabe der 8086-Segment-Register. Dazu spezifizieren Sie ein „S" nach dem „T" (TS # <cr>).

Führe teilweise aus

U # <cr>

Das „Untrace"-Kommando arbeitet genau wie das „Trace"-Kommando, nur daß Sie die Zeile mit den Inhalten der CPU-Register nur einmal vor der Ausführung der nachfolgenden Programmschritte sehen. Es ist ähnlich wie mit einem „G"-Kommando einen Breakpoint zu setzen und dann mit „X" fortzufahren. „U" ist schneller als „T", aber langsamer als „G". Trotzdem ist „U" einem „G" ohne Breakpoints vorzuziehen, denn Sie können die Ausführung durch Drücken einer Taste unterbrechen. Wie beim Kommando „Trace" wird mit DDT-86 durch ein auf das „U" folgendes „S" die Anzeige der Segment-Register bewirkt.

Prüfe CPU-Status

X <cr>

Das Kommando „X" wird für die Prüfung des gegenwärtigen Zustands der CPU benutzt. Das Format der Ausgabe ist wie bei „T" und „U", es sei denn, Sie spezifizieren dafür nur ein einzelnes Register wie:

DDT:

XC	Carry flag	Übertragungs-Kennzeichen
XZ	Zero flag	Null-Kennzeichen
XM	Minus flag	Minus-Kennzeichen
XE	Even parity flag	Kennzeichen für Bit-Anzahl gerade
XI	Interdigit carry flag	Kennzeichen für Übertragungspause
XA	Accumulator	Akkumulator
XB	Register pair BC	Registerpaar BC
XD	Register pair DE	Registerpaar DE
XH	Register pair HL	Registerpaar HL
XS	Stack pointer register	Pufferregister
XP	Programm counter register	Programmzählerregister

DDT-86

XO	Overflow flag	Überlauf-Kennzeichen
XD	Direction flag	Richtungs-Kennzeichen
XI	Interrupt enable flag	Kennzeichen zur Ermöglichung von Unterbrechungen
XT	Trap flag	Kennzeichen für Zeithaftstelle

XS	Sign flag	Vorzeichen-Kennzeichen
XZ	Zero flag	Null-Kennzeichen
XZ	Auxiliary carry flag	Zusätzliches Übertragungs-Kennzeichen
XP	Parity flag	Paritäts-Kennzeichen
XC	Carry flag	Übertragungs-Kennzeichen

Zusätzlich alle Register (AX, BX, CX usw.)

Lade für Ausführung (nur DDT-86)

Efilename <cr>

Das Kommando „E" erlaubt eine 8086-Datei so in den Kernspeicher zu laden, daß ein nachfolgendes „G", „T" oder „U" mit der Ausführung beginnt. Wenn kein Datei-Typ angegeben ist, wird „.CMD" angenommen. Wie bei 8086-Programmdateien üblich, wird der Header (= Kopfetikett) benutzt, um die Segment- und die IP-Register vor der Ausführung zu verändern. DDT-86 zeigt die Start-und Endadressen eines jeden Programmteils an, während die Datei geladen wird.

Werte (nur DDT-86)

V <cr>

Das „V"-Kommando zeigt Information über die letzte mit „E" oder „R" geladene Datei an. Wurde das Kommando „E" verwendet, listet „V" die Start- und Endadresse eines jeden Teils. Bei „R" zeigt „V" nur die Start- und Endadressen des Kernspeicherblocks an, in den die Datei geladen wurde.

Schreibe Datei (nur DDT-86)

Wfilename, #, # <cr>

Das „Write"-Kommando arbeitet ähnlich wie das SAVE-Kommando von CP/M-80. Es schreibt einen Informationsblock auf eine Plattendatei. Die beiden durch ein Komma getrennten Nummern nach dem Dateinamen sind die Start- und Endadresse des Kernspeicherblocks. Sind diese Adressen nicht angegeben, werden sie dem letzten „R"-Kommandoblock entnommen.

LOAD – Erstelle ein ausführendes Programm

Das Kommando LOAD des CP/M-80 hat nur eine Funktion: es nimmt eine Datei vom Typen „.HEX" und konvertiert sie in ein ausführbares Programm des Typen „.COM".

Der CP/M-80-Assembler erstellt eine „.HEX"-Datei im Maschinencode vom Intel-hex-Format, die dann mit DDT getestet oder in eine ausführbare Datei umgewandelt werden kann. LOAD erstellt eine „.COM"-Datei, die auf 0100 hex beginnt und den ausführbaren Maschinencode enthält.

Zuerst assemblieren Sie Ihre Datei mit „ASM". Danach erstellen Sie mit LOAD die „.COM"-Datei.

```
A>LOAD B:POX<cr>
FIRST ADDRESS          0100
LAST ADDRESS           0234
BYTES READ             0135
RECORDS WRITTEN          02
A>
```

Wenn Ihre „.HEX"-Datei lang ist, kann Ihr Laufwerk für die LOAD-Funktion unverhältnismäßig lange Zeit beanspruchen. Die ausgedruckten Meldungen haben nur informativen Charakter und geben Aufschluß über den Umfang der erstellten „.COM"-Datei.

LOAD-Fehlermeldungen

LOAD präsentiert verschiedene Fehlermeldungen, wenn Ihre Datei nicht im richtigen Format ist oder sonst ein Umwandlungsproblem besteht. Das kann sein:

CANNOT OPEN SOURCE, LOAD ADDRESS xxxx
Diese Meldung wird ausgegeben, wenn LOAD den Dateinamen nicht finden kann oder keiner angegeben wurde.

INVERTED LOAD ADDRESS
Der Programmstart liegt unter 0100 hex oder das Programm enthält mindestens ein Statement mit einer niedrigeren Adresse als der der vorherigen Instruktion (die Sätze im hex-Format sind nicht notwendig aufsteigend nach Adresse sortiert). Prüfen Sie die ORG-Statements in Ihrem Assembler-Quellprogramm.

INVALID HEX DIGIT (= ungültige Hexadezimalziffer)
CHECKSUM ERROR (= Prüfsummenfehler)
Diese Meldungen erscheinen, wenn Informationen innerhalb der hex-Datei falsch sind. Das kann normalerweise nicht bei einer direkten Assemblierung geschehen, wohl aber, wenn Sie versuchen, Informationen im Intel-hex-Format zwischen Computern auszutauschen und danach wieder die Datei auszugeben.

DISK READ (= Lesefehler)
DISK WRITE (= Schreibfehler)
NO MORE DIRECTORY
SPACE (= kein Platz im Inhaltsverzeichnis)
CANNOT CLOSE FILE (= Datei kann nicht abgeschlossen werden)
Jede dieser Fehlermeldungen wird durch LOAD bei spezifischen Problemen im Zusammenhang mit dem Lesen oder Beschreiben von Platten ausgegeben. Prüfen Sie, ob die Diskette oder das Inhaltsverzeichnis voll oder auf Read-only-Status gesetzt ist.

GENCMD – Erstelle eine ausführbare Programmdatei (nur CP/M-86)

CP/M-86-Programme können in nahezu jedem Teil des Kernspeichers verankert sein, während CP/M-80 für alle transienten Kommandos (Programme) die Startadresse 0100 hex erfordert. Deshalb wird ein CP/M-86-Programmierer GENCMD anstelle von LOAD benutzt.

GENCMD geht von einem Datei-Typ „.H86" (erstellt durch ASM-86) aus und macht daraus eine „.CMD"-Datei. In diesem Sinne funktioniert GENCMD genau wie LOAD.

Außerdem enthält GENCMD eine große Anzahl von Optionen. Diese erlauben Ihnen beispielsweise die Mischung von Code und Daten des Modells 8080, die Adressierung von Code-, Daten- oder Stapelelementen oder auch die Festlegung der Kernspeichergröße für jedes Datenteil.

Das Format des Kommandos GENCMD ist

```
A>GENCMD filename options
```

wobei „ filename" der Name der „.H86"-Datei und „options" eine Liste von Möglichkeiten ist, um die Generierung der „.CMD"-Datei zu modifizieren. Die gültigen Optionen sind:

8080
legt fest, daß Code und Daten zu einem einzigen Segment von 64K zusammengemischt werden, unabhängig von den Direktiven für den Gebrauch des spezifischen Teils im Programm

CODE [#]
Spezifiziert den Beginn des Code-Teils

DATA [#]
Spezifiziert den Beginn des Datenteils

EXTRA [#]
Spezifiziert den Beginn des Extrateils

STACK [#]
Spezifiziert den Beginn des Stack-Teils

X1 [#]

X2 [#]

X3 [#]

X4 [#]
Spezifizieren den Beginn von Hilfsteilen.

Kapitel 5
Transiente Programme und CP/M

Sie werden im Zusammenhang mit CP/M viele verschiedene Typen von Programmen gebrauchen. Digital Research bietet eine Reihe von Unterstützungsprogrammen, um Ihnen bei der Entwicklung Ihrer eigenen Software zu helfen, vermarktet aber kaum Benutzer-Anwendungsprogramme. Bedenken Sie, CP/M ist nicht die vollständige Lösung, obwohl Anwendungsprogramme sich weitestgehend an diese Lösung anlehnen.

Die vier Klassen von **Lösungs**programmen, die dieses Kapitel behandelt, sind:

- Dienstprogramme
- Höhere Programmiersprachen
- Anwendungsprogramme
- Text-Prozessoren

In Kapitel 2 benutzten wir den Begriff **Haushaltung**, um Programme zu beschreiben, die Ihnen helfen, Disketten und ihre Dateien sauber zu halten. Eine andere Klasse allgemein anwendbarer Programme sind **Dienst**programme. Eine Diskette mit Dienstprogrammen enthält Routinen, die Ihre Diskettenkollektion instandhalten. Solche Dienstprogramme sind:

- Editoren
- Programme zum Kopieren von Platten
- Programme zum Formatieren von Platten
- CP/M-System-Generatoren
- Programme zur Einsichtnahme in Platten

Editoren unterscheiden sich im Grad ihrer Ausarbeitung; ein Bleistift, eine Schere, ein Band, eine Schreibmaschine oder ein Kopiergerät addieren oder modifizieren auf verschiedene Weise Informationen auf Papier. Ein Editor oder ein Text-Prozessor addiert, vernichtet oder modifiziert Informationen auf einer Platte.

Sie können auf einer Platte Information durch den Context-Editor von Digital Research, durch ED oder durch ein gutes Textverarbeitungsprogramm (z. B. WordStar) hinzufügen. In diesem Kapitel beschreiben wir in Kurzform verschiedene populäre Editoren und ihre Relation zu CP/M.

Bevor wir mit der Beschreibung der Dienst-, Sprachen- und Anwendungsprogramme im Zusammenhang mit CP/M fortfahren, ist es angemessen, eine wichtige Grundlage sicherzustellen, daß Sie nämlich die Hierarchie der Software verstehen.

Maschineninstruktionen sind eine der verschiedenen Softwareschichten zwischen dem Computer und dem Benutzer. Historisch gesehen, war Software oft darauf ausgerichtet, die Weiterentwicklung von Mikrocomputerprogrammen zu erleichtern. Infolgedessen existieren verschiedene Schichten von Software; jede Schicht beruht direkt auf einer vorherigen Entwicklungsstufe.

Maschineninstruktionen sind das Herz der Programmierung. Sie sind die Binärziffern (1 und 0), die die CPU interpretiert. Die verschiedenen Formationen von Einsen und Nullen veranlassen die CPU jeweils zur Ausführung einer eindeutigen Aufgabe. Jedes CPU-Modell hat einen bestimmten Instruktionssatz, der von ihm interpretiert werden kann. Deshalb unterscheiden sich CP/M-80 und CP/M-86. Um einen Computer auf dieser Ebene zu programmieren (instruieren), brauchen Sie eine Zugangsmethode zu den Maschineninstruktionen. CP/M bietet diese Möglichkeit nicht direkt, indirekt können Sie jedoch Instruktionen über DDT eingeben (s. Kap. 4). Zwei Konzeptionsprobleme sind jedenfalls in bezug auf Maschineninstruktionen offensichtlich.

Eine Maschineninstruktion (z. B. 1100 0011) hat wenig Ähnlichkeit mit der Operation, die der Computer daraufhin durchführt. Das trifft auf jeden Code und jede Sprache zu. Da die Einsen und Nullen lediglich Variationen von ON- und OFF- (EIN- und AUS-)Zuständen innerhalb des Computers darstellen, müssen wir sie weiter übersetzen. In menschlicher Sprache bilden Zahlenketten allein noch keine Basiskomponenten der Verständigung. Um sie im Zusammenhang zu begreifen, bedarf es einer vermittelnden Übersetzung. Computer sind ähnlich, und so ist die Übersetzung von „1100 0011" eben ein „Sprung" zu einer Adresse in einem auf dem 8080 basierenden Computer.

Instruktionen bewirken Operationen, die stark in Aufbau und Terminologie eines Computers verwurzelt sind. Es ist schwierig, einem Computer mit seinen Maschineninstruktionen zu beschreiben, was er tun soll. Angenommen, Sie möchten einen Drucker veranlassen, ein Zeichen auszugeben, können einige der notwendigen Instruktionen die folgenden sein:

Hexadezimale Repräsentation	**Binäre Maschinen-Repräsentation**
DB	11011011
03	00000011
E6	11100110
01	00000001
C2	11000010
00	00000000
00	00000000
3E	00111110
58	01011000
D3	11010011
02	00000010

Die Assembler-Sprache geht einen Schritt weiter und insofern auch weg vom Computer, indem sie ein Mnemonic (Sprachsymbol) zur Aktionsbeschreibung jeder Maschineninstruktion anwendet. Für das obige 8080-Beispiel (Senden eines Zeichens zur Ausgabe an den Drucker) erhalten wir nun:

```
IN   03
ANI  01
JZ   0000
MVI  A,58h
OUT  02
```

Wenn Sie die Bedeutung von MVI, IN, ANI, JZ und OUT kennen, ist die Instruktion nun in abgekürzter Form in verständliche Sprache übersetzt. Natürlich schildern diese fünf Programmzeilen in Assembler-Sprache lediglich die Bedeutung jeder einzelnen Instruktion. Was bewirken jedoch diese fünf Zeilen zusammen – als ein Programm?

Ein weiterer Schritt (auch weg vom Computer) ist die Interpretation höherer Programmiersprachen. Hier werden Assembler-Instruktionen in größeren Blocks kombiniert. Unser Programm wird nun zu

```
LPRINT „X“,
```

das das Zeichen „X“ an den Drucker sendet. Nun wird klar, was das Programm bewirkt. LPRINT „X“ könnte theoretisch die Ausführung von Hunderten von Maschineninstruktionen umfassen, aber durch diesen Befehl begreifen wir sofort die Funktion des Programms. Je näher wir also dem Verständnis der jeweiligen Aufgabenstellung des Computers kommen, desto weiter entfernen wir uns von den Maschineninstruktionen, die die CPU tatsächlich ausführt. Die ersten Sprachen höherer Ebenen wurden mit Hilfe der nächst höheren Methode, in diesem Falle der Assembler-Sprache, entwickelt. Und die ersten Assembler-Sprachprogramme wurden in Maschinensprache geschrieben.

Die letzte Ebene, die wir zu betrachten haben, ist das Anwendungsprogramm. Ein Anwendungsprogramm weist einen Computer an, eine spezifische Aufgabe auszuführen, z. B. eine Namens- und Adreßliste zu erstellen und zu warten. Die meisten Computeranwender verwenden Benutzerprogramme ohne Kenntnis der niederen Ebenen der Programmverknüpfung.

Nun, nachdem wir uns mit den Ebenen der Programminstruktionen befaßt haben, können wir die vier Typen von Programmen, die Sie im Zusammenhang mit CP/M verwenden können, detaillierter betrachten.

Utility-(= Dienst-)Programme

Wir wollen verschiedene nützliche Dienstprogramme in allgemeiner Fassung präsentieren. Jeder Systemberater für CP/M wird sicher ähnliche Dienstprogramme anbieten, aber nicht die hier beschriebenen. Es ist wichtig, von diesen Utilities Kenntnis zu nehmen und zu lernen, sie anzuwenden. Insbeson-

dere gilt dies für die Programme zum Kopieren und Formatieren von Platten und Programmen. Während Sie die folgenden Abschnitte lesen, vergleichen Sie die Handbücher, die mit Ihrer Version des CP/M geliefert wurden, um zu sehen, wie Sie Ihre CP/M-Dienstprogramme einsetzen können.

Format – Vorbereitung einer Diskette für den Gebrauch

Woher weiß der Computer, wie er eine Information auf eine Diskette zu schreiben hat? Wir haben bisher beschrieben, daß Disketten in Sektoren und Spuren ausgelegt sind, aber wie unterscheidet der Computer dazwischen? Wie ordnet er Sektor- oder Spurinformation zu? Ein Teil dieser Aufgabe wird von der Hardware und von CP/M erledigt, den andern Teil müssen Sie übernehmen, indem Sie die Diskette vorbereiten.

Nahezu jedes CP/M-80- oder CP/M-86-Package (= Utility-Paket) schließt ein Formatierungs- bzw. Initialisierungsprogramm ein. Unter anderen wurden folgende Namen für Initialisierungsprogramme vergeben:

FORMAT	INIT	IN
DSKFMT	FMT	CREATE
INITDSK	INITDISK	FORMT#
FORMTHD	MFORMAT	HDF

Der genaue Name Ihres Formatierungsprogrammes hängt von Ihrer CP/M-Bezugsquelle ab. In einigen wenigen Fällen wird keines vorhanden sein; statt dessen wird die begleitende Dokumentation beschreiben, wie eine Diskette zu initialisieren ist.

Jede leere Diskette muß formatiert werden, bevor sie Daten übernehmen kann. Es empfiehlt sich, dies selbst vorzunehmen, selbst wenn Ihr Diskettenverkäufer angibt, es bereits für Sie erledigt zu haben.

Angenommen, Ihr Programm zur Initialisierung von Disketten heißt FORMAT. Die Ausführung könnte etwa so aussehen:

```
A>FORMAT<cr>
DISK TO FORMAT? (A,B,C,D) B<cr>
PRESS RETURN TO BEGIN FORMATTING <cr>
DISK FORMATTED. MORE? (Y/N) N<cr>
A>
```

Die genaue Syntax der Meldungen mag unterschiedlich sein, die folgenden Schritte werden jedoch in jedem normalen Formatierungsprogramm unternommen werden:

1. Das Programm aufsetzen (Tippen des Namens, danach ein CARRIAGE RETURN)
2. Mitteilung an das Programm, in welchem Laufwerk sich die zu formatierende Diskette befindet

3. Spezifikation der Dichte, wenn Ihr System Disketten einfacher und doppelter Dichte verarbeiten kann
4. Anweisung an das Programm, mit der Formatierung zu beginnen
5. Mitteilung an das Programm, ob Sie es beenden oder weitere Disketten formatieren wollen.

Es gibt sogar Programme, die unter Umständen empfehlen, eine Diskette nicht zu formatieren, um eine irrtümliche Vernichtung unersetzbarer Information zu verhindern.

Formatierungsprogramme beachten eine Schreibschutzsperre auf der Diskette aber manchmal auch nicht. Um dieses herauszufinden, versuchen Sie es. Sollte Ihr Formatierungsprogramm den Schreibschutzmechanismus ignorieren, müssen Sie besonders vorsichtig sein.

Die Initialisierung einer Diskette ist nur die Vorbereitung für den Gebrauch. Wenn Sie daraufhin nach einem Inhaltsverzeichnis fragen, erscheinen Meldungen wie NO FILE oder NOT FOUND, weil die Diskette noch leer ist. Auch kann man mit der Diskette das System nicht starten: die Diskette enthält noch nicht das CP/M-Betriebssystem (s. SYSGEN nachfolgend).

Das Formatierungsprogramm bewegt den Magnetkopf des Plattenlaufwerks zum ersten Sektor der ersten Spur. Dann schreibt das Programm eine Pseudoinformation und fährt fort. Jeder Sektor einer Spur wird so beschrieben, danach bewegt sich der Kopf zur nächsten Spur. Die meisten Programme füllen die Diskette mit dem hexadezimalen Byte E5. Wahrscheinlich wird es Sie wenig kümmern, was geschieht oder wie es geschieht, solange die Diskette richtig formatiert ist.

Seien Sie besonders vorsichtig, wenn Sie Hartplatten formatieren. Normalerweise geschieht das nur einmal. Formatierung löscht alle Information von einer Plattenoberfläche, und da es hier sehr viele Informationen sind, kann ihre Rückgewinnung nach versehentlicher Formatierung sehr lange dauern.

Copy – Übertragung von Information zwischen Platten

Angenommen, Sie wollen eine ganze Diskette kopieren. Sie könnten das in Kapitel 3 beschriebene PIP benutzen, aber es arbeitet langsam und initialisiert keine Leerdiskette. Also nehmen Sie ein COPY-Programm.

Wie beim Initialisieren hängt auch der Name des Programmes zum Kopieren ganzer Disketten von Ihrer CP/M-Version ab. Das liegt daran, daß Digital Research mit dem CP/M-80 weder ein Initialisierungs- noch ein Kopierprogramm herausgab (in CP/M-86 von Digital Research ist ein Dienstprogramm COPYDISK eingeschlossen). Solche Programme müssen, sollen sie zweckmäßig sein, speziell für jedes Hardware-System geschrieben sein; Hersteller und Händler bieten gewöhnlich gängige Versionen dieser beiden Programme an. Zu den vielen Namen von Kopierprogrammen gehören:

COPY	DISKCOPY	COPYDISK
DSKCPY	BACKUP	COPYATOB
COPYID	SDCOPY	MCOPY
FCOPY	FASTCOPY	DUPE

Diese Programme arbeiten ähnlich wie die vorher besprochenen Formatierungsprogramme.

```
A>DISKCOPY<cr>
SOURCE DRIVE? A<cr>
DESTINATION DRIVE? B<cr>
PRESS RETURN TO BEGIN COPYING<cr>
COPY COMPLETE. MORE? (Y/N) N<cr>
A>
```

Die meisten Kopierprogramme arbeiten ähnlich, aber sie müssen nicht genau unserem Beispiel entsprechen. Gewöhnlich treten die folgenden Schritte auf:

1. Aufsetzen des Kopierprogrammes
2. Anzeige, welches Laufwerk die Originaldiskette enthält (die, VON der kopiert werden soll)
3. Anzeige, welches Laufwerk die **Empfangs**diskette enthält
4. Beginn des Kopiervorgangs durch CARRIAGE RETURN
5. Ende oder Anzeige, ob noch mehr Kopien gemacht werden sollen.

Prüfen Sie anhand Ihrer Handbücher, ob die Zieldiskette initialisiert sein muß. Nur wenige Kopierprogramme erfordern dies. Sollten Sie also bei einer Leerdiskette eine Fehlermeldung erhalten, versuchen Sie zuerst, sie zu initialisieren.

MOVCPM – Adjustierung von CP/M auf die Kernspeicherkapazität

Wenn Sie Ihre CP/M-80-Diskette zuerst erhalten, kann man damit normalerweise in einem 16K- oder 24K-Computer arbeiten. Wie Sie schnell feststellen werden, reicht dieser Speicher jedoch für die Ausführung der meisten Programme nicht aus, insbesondere wenn Sie eine höhere Programmiersprache wie BASIC oder Pascal verwenden. In diesem Falle müssen Sie CP/M-80 auf den maximal verfügbaren Kernspeicher in Ihrem System hinweisen. Wenn Sie 48 Kilobytes zur Verfügung haben und ein CP/M-80 erhalten, das nur 24K erwartet, verschwenden Sie die anderen 24K. CP/M-86 braucht kein MOVCPM, weil es normalerweise mit der Untergrenze des verfügbaren Kernspeicherbereiches auskommt und den Rest Ihrer Entscheidung überläßt.

Das Kommando MOVCPM beinhaltet eine simple Methode, die Erwartung von CP/M-80 in Bezug auf den Kernspeicherumfang zu ändern (dieser

Prozeß wird „moving" genannt). Man kann CP/M-80 nach dem "moven" sichern oder nicht.

Für die sofortige Ausführung tippen Sie

```
MOVCPM<cr>
```

es gibt den gesamten existierenden Kernspeicher-Umfang an

oder

MOVCPM # <cr>

wobei # die dezimale Anzahl von Kilobytes angibt, für die CP/M-80 ausgelegt werden soll. Dieses Kommando läßt sich anwenden, wenn Sie außer CP/M noch Spezial- oder Dienstprogramme gleichzeitig laden wollen.

Meistens werden Sie jedoch eine endgültige Lösung wünschen. Um CP/M-80 zu „moven" und dann auf Diskette zu sichern, tippen Sie

```
MOVCPM**<cr>          oder
MOVCPM # *<cr>
```

wobei das letzte „*" Sicherung bedeutet.

Die „#" in den obigen Beispielen muß

- für die Versionen 1.3 und 1.4 des CP/M-80 eine Dezimalzahl zwischen 16 und 64 sein,
- ab Version 2.0 des CP/M-80 eine Dezimalzahl zwischen 20 und 64 sein,
- kleiner oder gleich der Kernspeichergröße in Kilobytes Ihres Computers sein.

Das Sichern Ihres neuadjustierten CP/M-Betriebssystems auf Diskette erfordert einen weiteren Schritt. MOVCPM fordert ihn durch folgendes Display an:

```
READY FOR „SYSGEN" OR
„SAVE 32 CPM # #.COM"
```

Diese rätselhafte Meldung bedeutet, daß Sie, um das gerade generierte CP/M zu sichern, „SYSGEN<cr>" oder „SAVE 32 CPM # #.COM" tippen müssen („# #" ist der Umfang des aus dem MOVCPM-Kommando resultierenden neuen CP/M-80-Betriebssystems). Wenn Sie nicht gerade erfahren in der Anwendung von CP/M-80 sind, sollten Sie einfach „SYSGEN<cr>" eingeben (s. nächsten Abschnitt).

MOVCPM wird gelegentlich anders genannt, wie CPM, NEWCPM, MAKECPM, CPMGEN oder MOVECPM; in keinem Falle hat es Einfluß auf irgendwelche Dateien auf der Diskette.

Wenn MOVCPM nicht korrekt arbeitet, kann es sein, daß Ihr Computer „stirbt" (ebenfalls unzuverlässig arbeitet). Drücken Sie den RESET-Knopf, um den alten Zustand wiederherzustellen. Es bedeutet, daß dieser Teil Ihres CP/M-80 (das BIOS), der es einem speziellen Computer angleicht, in Ihrem

MOVCPM-Programm (s. Kap. 7) nicht enthalten ist oder daß Sie mehr Kernspeicher angegeben haben, als Ihr System enthält. Diese letzte Möglichkeit wird häufig übersehen; der Computer „Osborne 1" sowie die meisten Computer, die kernspeichergesteuertes Video verwenden, benutzen die ersten 4K für das Video-Display; Sie können also nur ein CP/M-80-System für 60K, nicht für 64K kreieren.

SYSGEN – Placierung des Systems CP/M-80 auf einer Diskette

SYSGEN steht für **system generation**, also die Einrichtung des CP/M-80-Betriebssystems.

Warum läßt sich das System CP/M-80 nicht genauso kopieren wie eine Datei? Weil es nicht so gespeichert ist. Die für diesen Unterschied verantwortlichen Einzelheiten werden am Ende dieses Abschnittes erklärt.

Die Generierung eines Systems geschieht durch eine der drei folgenden Arten:

1. Durch SYSGEN allein, womit ein System unverändert auf eine neue Diskette kopiert wird.
2. Durch MOVCPM und danach SYSGEN, wodurch ein neues System auf einer neuen Diskette generiert wird.
3. Durch MOVCPM mit anschließender Sicherung, danach einem erneuten Laden des Resultats zwecks Modifikation durch DDT und endgültiger Sicherung auf Diskette durch SYSGEN.

Die Modifizierung des Systems wird in Kapitel 7 beschrieben. Hier wollen wir nur den Gebrauch von SYSGEN alleine schildern und danach seinen Gebrauch im Zusammenhang mit MOVCPM. SYSGEN berührt keine Disketten-Datei.

Wenn Sie das System von einer Diskette kopieren, könnte Ihr Dialog mit dem Computer folgendermaßen aussehen:

```
A>SYSGEN<cr>
SYSGEN VER 2.2
SOURCE DRIVE NAME (OR RETURN TO SKIP) A
SOURCE ON A:. THEN TYPE RETURN <cr>
FUNCTION COMPLETE

DESTINATION DRIVE NAME (OR RETURN TO REBOOT) B
DESTINATION ON B:, THEN TYPE RETURN <cr>
FUNCTION COMPLETE
DESTINATION DRIVE NAME (OR RETURN TO REBOOT) <cr>
A>
```

Lassen Sie uns jeden einzelnen Schritt genauer betrachten. Zuerst tippen Sie SYSGEN <cr>, damit es geladen und ausgeführt wird. Dann geben Sie „A" als Originallaufwerk an. Auf die anschließende Frage nach dem Empfangs-

laufwerk geben Sie „B" als Laufwerk an und ein CARRIAGE RETURN, wenn die Platten eingelegt sind. Danach werden die Laufwerke eine Weile surren und klicken, und dann beginnt der Prozeß von vorn (DESTINATION DRIVE NAME: . . .). Er wird entweder durch ein <cr> beendet, oder ein neuer Laufwerks-Spezifikator teilt SYSGEN mit, daß Sie das gleiche System noch auf eine andere Diskette speichern möchten.

Wenn Sie das neu generierte System CP/M-80 nach Laufwerk „A" sichern wollen, könnten Sie statt „B" dieses Laufwerk angeben. Dann allerdings könnte sich nach <cr> irgend etwas als falsch erweisen. Normalerweise sollten Sie vom SYSGEN auf der Diskette ausgehen, mit der Sie das System zuerst gestartet haben.

Wenn Sie das neue mit MOVCPM generierte System sichern wollen, wird Ihr Dialog mit SYSGEN etwas anders aussehen. Nehmen wir also an, Sie tippen „MOVCPM * * <cr>" oder „MOVCPM # # * <cr>" und MOVCPM teilt Ihnen nun mit, daß es für ein SYSGEN oder ein SAVE bereit ist, verfahren Sie folgendermaßen:

```
READY FOR "SYSGEN" OR
"SAVE 32 CPM# #.COM"
A>SYSGEN<cr>
SYSGEN VER 2.2
SOURCE DRIVE NAME (OR RETURN TO SKIP) <cr>
DESTINATION ON B:, THEN TYPE RETURN <cr>
FUNCTION COMPLETE
DESTINATION DRIVE NAME (OR RETURN TO REBOOT) <cr>
A>
```

Wie das Beispiel zeigt, fragt SYSGEN weiterhin nach einer Zieldiskette, bis sie nur <cr> eingeben; Sie können also in einer Sitzung viele Disketten updaten.

SYSGEN kann auch den Namen eines CP/M-80-Betriebssystems einschließen, das als Datei gesichert wurde, z. B. SYSGEN CPM48.COM<cr>. Diese Variante des Kommandos SYSGEN lädt die Systemdatei in den Kernspeicher und fragt dann nur noch nach dem Bestimmungslaufwerk.

Warum SYSGEN notwendig ist

Die beschriebene Prozedur erfordert weitere Erklärung.

Die ersten zwei Spuren – 0 und 1 – jeder CP/M-80-Diskette enthalten keine Dateien. Stattdessen ist dieser Bereich von 6656 Bytes für „bootstrap loaders" (Kaltstart-Lader) und das CP/M-80-Betriebssystem reserviert. Diese Programme sind nicht als Dateien gespeichert; weder erscheinen sie im Inhaltsverzeichnis noch kann man auf sie wie auf eine Datei zugreifen. Jedenfalls kann es gelegentlich notwendig werden, auch auf diese Spuren zuzugreifen.

Die Spuren 0 und 1 einer Diskette werden grundsätzlich ausgelassen, egal ob sie Lader und Systemprogramme enthalten oder nicht. Man braucht sie schließlich nicht auf jeder Diskette, sondern nur, wenn Sie mit ihr einen Kalt- oder Warmstart durchführen wollen.

Die meisten Disketten-Kopierprogramme kopieren das System oder, wenn es fehlt, die gesamte Diskette. Disketten mit besonders wertvollen Anwenderprogrammen, die Sie von Software-Häusern nur selten erhalten, enthalten eigentlich immer CP/M-80 auf den Systemspuren.

Deshalb eben wird SYSGEN zur Speicherung oder Modifizierung eines Betriebssystems auf Diskette benötigt. Nur dadurch können Sie Ihr geändertes CP/M-80 sichern.

Ein letztes Wort zur System-Generierung: MOVCPM überträgt nur das reine Skelett eines Betriebssystems. Änderungen wie beispielsweise der Anschluß eines zusätzlichen Druckers oder anderer Einheiten, die das BIOS (Ein-/Ausgabe-System – die Sektion BIOS s. Kap. 7) berühren, werden durch MOVCPM nicht übertragen. Wenn also dann eine solche Einheit nicht funktioniert, brauchen Sie vielleicht Spezial-Routinen. Andererseits kopiert SYSGEN stets das gesamte Betriebssystem einschließlich BIOS, also die Spuren 0 und 1.

Generierung eines neuen CP/M-86-Systems

Bis jetzt haben wir neue Systeme nur im Zusammenhang mit CP/M-80 diskutiert. CP/M-86 operiert jedoch in unterschiedlicher Weise, deshalb muß hier auch ein neues System anders generiert werden.

CP/M-86 lädt das Betriebssystem von einer gesicherten Diskettendatei namens CPM.SYS. Dazu muß allerdings ein Kaltstart-Ladeprogramm auf den ersten beiden Spuren der Diskette vorhanden sein. Da die Aufgabe kompliziert ist, sollte sie nur von jemand durchgeführt werden, der mit dem 8086-Assembler vertraut ist; deshalb wollen wir die Schritte zur Kreierung eines neuen Ladeprogrammes nur kurz aufführen:

1. Assemblieren Sie eine spezielle Version Ihres BIOS mit dem Namen LDBIOS.A86.
2. Verketten Sie die resultierende Datei aus LDBDOS.H86 und LDCPM.H86 durch

   ```
   PIP LOADER.H86 = LDCPM.H86,LDBDOS.H86,LDBIOS.H86<cr>
   ```

3. Generieren Sie eine resultierende Kommando-Datei (Typ „.CMD")

   ```
   GENCMD LOADER 8080 CODE [A400] <cr>
   ```

   (Das angegebene Beispiel kreiert ein 8080-Kernspeicher-Modellsystem, beginnend mit der absoluten Adresse 0400 hex).
4. Laden Sie mit DDT oder DDT-86 eine Kopie der Datei LOADER.CMD

in den Kernspeicher und speichern Sie mit SYSGEN oder LDCOPY eine Kopie des CP/M-86-Laders auf die Systemspuren Ihrer Diskette.

Wenn Sie Ihr CP/M-86-System starten, wird der Lader von den Systemspuren in den Kernspeicher geladen und ausgeführt, was zum gleichen Vorgang bei der Datei CPM.SYS führt. CPM.SYS ist wie folgt organisiert:

- 128-Byte-Kopfetikett zur Identifizierung von Ladeparametern
- CCP.CMD
- BDOS.CMD
- BIOS.CMD

Eine Datei CPM.SYS zu kreieren ist ähnlich wie der gerade für den System-Lader beschriebene Prozeß:

1. Assemblieren Sie Ihre Datei BIOS (BIOS.A86).
2. Verketten Sie die resultierende Datei mit CPM.H86 zu

   ```
   PIP CPMTEMP.H86 = CPM.H86, BIOS.H86<cr>
   ```

3. Konvertieren Sie die resultierende in eine Kommando-Datei

   ```
   GENCMD CPMTEMP 8080 CODE [A40] <cr>
   ```

4. Benennen Sie die resultierende Datei für den Gebrauch neu

   ```
   REN CPM.SYS = CPMTEMP.CMD
   ```

CPM.SYS sollte die erste Datei auf einer CP/M-86-Boot-Diskette sein.

Höhere Programmier-Sprachen

Wie in diesem Kapitel bereits erwähnt, sind höhere Programmiersprachen einer der Aufbaublocks, die von Programmentwicklern benutzt werden. Sie schreiben Computerprogramme aus vielen Gründen in höheren Sprachen. Die hauptsächlichen Gründe sind:

- Höhere Sprachen sind leichter in der Anwendung, denn eine ihrer Anweisungen umfaßt in der Regel die Bedeutung mehrerer Maschineninstruktionen. Programmierung und Programmlesbarkeit wird schneller und einfacher.
- Es ist leichter, den auszuführenden Prozeß zu konzeptualisieren, wenn die Kommandos menschlicher Kommunikation angenähert sind. So ist z. B. PRINT für menschliche wie Computersprachen verständlich.
- Es wurden höhere Programmiersprachen für besondere Computer-Aufgaben entwickelt (Computer-Kontrolle von Maschinen, Schwerpunkt auf mathematischen Kalkulationen usw.).

Einschluß oder Ausschluß einer bestimmten Sprache von diesem Kapitel bedeutet weder Lob noch Tadel der Software durch den Autor oder den

Herausgeber. In der Diskussion innerhalb dieses Kapitels geht es nicht um die relativen Vorzüge bestimmter Software-Angebote.

Geschichte höherer Programmiersprachen für CP/M-80 und CP/M-86

Vor der Beschreibung der Details verschiedener höherer Programmiersprachen ist eine kurze Zusammenfassung der Entwicklungsgeschichte höherer Sprachen im Zusammenhang mit CP/M sinnvoll.

CP/M wurde ein Quasi-Standard-Betriebssystem, weil es als eines der ersten verfügbar war. Ein Betriebssystem wirkt als Steuer- (scheduler) und Vermittlungs- (arbitrator)programm für die verschiedenen Aufgaben, die ein Computer zu erfüllen hat. Mit anderen Worten: um Plattenlaufwerke zu benutzen, brauchen Sie ein Platte-Betriebssystem; dieses ist aber nur das Verbindungsstück, kein Endzweck.

Bald nachdem CP/M-80 öffentlich verkauft wurde, gab Gordon Eubanks jr. EBASIC als höhere Programmiersprache heraus, die er als Teil seiner Doktorarbeit entwickelt hatte (EBASIC wird manchmal auch BASIC-E genannt). EBASIC wurde in PL/M geschrieben, einer anderen höheren Programmiersprache. Während andere Sprachen im Zusammenhang mit CP/M-80 benutzt werden konnten, wurde EBASIC relativ früh zur Verfügung gestellt und machte guten Gebrauch von den Handhabungsinstrumenten des CP/M für logische und physische Einheiten (s. Abschnitt über E/A-Einheiten in Kapitel 3). EBASIC wurde mit Regierungsgeräteeinrichtungen geschrieben, und das sicherte dieser Software automatisch seinen Rang in der Anwendung durch Öffentlichkeitsorgane. Für Eubanks konnten keine Copyright-Privilegien auflaufen, noch konnte EBASIC für mehr als eine „vernünftige Kopier-Entschädigung" verkauft werden. So wird in der Tat EBASIC – diese Konstruktion von PL/M-Instruktionen (Quell-Code) – von der CP/M-Anwendergruppe angeboten (s. Anhang G).

EBASIC ist ein Compilertyp für BASIC. Sie erlangen Zugang zu BASIC-Instruktionen über einen Editor, sodann erstellt EBASIC daraus eine Vermittlungsdatei. Aber lassen Sie uns einen Schritt erneut betrachten, was nämlich während der Ausführung eines Programmes oder seiner Kompilierung innerhalb eines Computers geschieht.

Der Computer verarbeitet nur Maschineninstruktionen im Binär-Format. BASIC ist eine in Dartmouth entwickelte Computersprache für Anfänger; es hat Instruktionen wie:

PRINT
GOTO
LET
IF

Wie werden diese Instruktionen in Einsen und Nullen übersetzt? Im EBASIC haben Sie, nachdem Sie mit einer Reihe von BASIC-Instruktionen über

einen Editor eingegeben haben, eine Textdatei. Wir nennen diese Programmdatei **Quell-Code**. Der Quell-Code besteht aus erkennbaren Buchstaben und Ziffern. Mit Hilfe anderer CP/M-80-Programme können Sie den Text des Quell-Codes manipulieren. So können Sie z. B. mit PIP den Text über Drucker kopieren.

Auf Ihrer BASIC-Diskette haben Sie wahrscheinlich eine Datei namens EBASIC.COM (manchmal als BAS.COM abgekürzt). Mit dieser Datei erstellen Sie eine Zwischendatei aus dem Quellprogramm. Wenn Sie ein BASIC-Programm namens SOURCE.BAS haben, tippen Sie

```
EBASIC SOURCE<cr>
```

Die Konvertierung von Text zu der kompakteren Form, die der Computer benutzt, geschieht automatisch. Nach ihrer Vollendung enthält Ihre Diskette eine Datei namens SOURCE.INT. Diese neue Datei beinhaltet keinen Text; stattdessen wurde jede BASIC-Instruktion zu einer 1-Byte-Repräsentation zusammengezogen. Z. B. wird eine PRINT-Instruktion aus der Quell-Datei (SOURCE.BAS) als 1A hex in die Zwischen-Datei (intermediate code file) SORUCE.INT gespeichert. Aus diesem Prozeß der Komprimierung ergibt sich eine Reihe von Vorteilen (er wird oft **Kompilierung** genannt, obwohl ein Compiler im eigentlichen Sinne wirkliche Maschineninstruktionen generiert, nicht Repräsentationen von Hochspracheninstruktionen). Die entscheidensten Vorteile sind eine Verringerung der Dateigröße und der Ausführungszeit. Es erfordert weniger Zeit, ein Byte wie „1A" zu inerpretieren als eine Reihe von Buchstaben wie die PRINT-Instruktion.

Um ein in EBASIC geschriebenes Programm dann auszuführen, benutzen Sie einen anderen Anteil namens RUN.COM. Er interpretiert komprimierte Instruktionen der Datei SOURCE.INT. RUN.COM wird in den Kernspeicher geladen und beginnt mit der Ausführung, danach wird SOURCE.INT geladen und von RUN.COM interpretiert. EBASIC überwacht diesen Vorgang.

EBASIC ist für kommerzielle Anwendung nicht besonders geeignet. Dennoch, weil es eine der ersten höheren Programmiersprachen für CP/M-80 ist und so wenig kostet, wurde es relativ frühzeitig benutzt, um CP/M-80-Systeme zu erweitern. EBASIC ist bereits eine Erweiterung des Standard-BASIC von Dartmouth, ihm fehlen jedoch gewisse Charakteristika für eine brauchbare geschäftsorientierte Sprache.

Glücklicherweise hat Eubanks mit EBASIC nicht aufgehört. Er benutzte die aus der Entwicklung von EBASIC gewonnene Erfahrung, gründete eine kleine Firma und entwickelte CBASIC (später auch CBASIC2). CBASIC ist eine Verbesserung von EBASIC; viele EBASIC-Programme werden mit CBASIC laufen, aber nicht notwendig umgekehrt. Viele Programmentwickler haben CBASIC bzw. CBASIC2 benutzt (es arbeitet übrigens mit der für EBASIC beschriebenen 3-Schritte-Prozedur. Das Programm CRUN bewirkt die Ausführung gemäß dem CBASIC-Zwischenprogramm).

Während EBASIC und CBASIC sich zu einer primären Hochsprache für CP/M-80 entwickelten, begannen zwei junge Männer aus Washington in einer Firma namens Microsoft mit der Entwicklung einer anderen BASIC-Version. Zu den Kunden von Microsoft gehören MITS – die Firma, die die Revolution der Personal Computer eingeleitet hat –, Radio Shack, Apple, Texas Instruments, Exidy, Ohio Scientific, Osborne, IBM und andere Mikrocomputer-Hersteller. Was als ein auf Kassette basierender BASIC-Interpreter bescheidenen Umfangs begann, ist zu einer umfassenden, auf Platte basierenden Sprache geworden. Die ersten Versionen von Microsoft-BASIC waren:

8K BASIC

Normalerweise mit Kassetten angewandt; in verschiedenen Formen bei MITS, Ohio Scientific, Apple, Radio Shack und Exidy erhältlich.

Disk BASIC

Platten-BASIC, ursprünglich für MITS entwickelt, aber für Radio Shack- und CP/M-80-Systeme modifiziert.

Erweitertes Disk BASIC

Die laufende Version des Interpreters, wie er von CP/M-80 und CP/M-86 benutzt wird.

Bis 1980 waren alle Versionen des Microsoft BASIC Interpretersprachen höherer Ordnung, im Gegensatz zu Compilersprachen wie CBASIC. Während Sie hier zur Programmeingabe einen Editor benutzen, geben Sie mit einem BASIC-Interpreter Programme direkt in den Kernspeicher ein. Der BASIC-Interpreter hat einen eingebauten Zeilen-Editor; Sie laden BASIC mit einem Kommando (MBASIC<cr>), und danach akzeptiert der Interpreter gültige Statements. Diese werden sofort übersetzt (jeweils direkt nach dem CARRIAGE RETURN). Jede Zeile wird sofort bearbeitet.

Wieder spart ein komprimierendes Schema Raum und Zeit in der aktuellen Ausführung eines Programms. Das Statement wird, sofort nachdem es für direkte Ausführung eingegeben wurde, komprimiert.

Anders als eine Compiler- erlaubt eine Interpretersprache, jedes Programmsegment individuell zu entwickeln und zu testen. Sie können jederzeit die Eingabe von Instruktionen beenden und durch „RUN<cr>“ mit der Ausführung von Teilprogrammen beginnen. Wenn ein Instruktionsfehler erkannt wird, wird die Zeile identifiziert und eine Meldung über den Typ des Fehlers ausgegeben. Sie können die betreffende Zeile sofort ausgeben, den Fehler korrigieren und das Programm erneut starten.

Vergleichen Sie dies mit der Arbeitsweise eines Compilers. Sie müssen BASIC verlassen, zu CP/M zurückkehren, den Editor aufrufen, die Programmdatei editieren und dann erneut kompilieren – alles, bevor Sie das Programm erneut laufen lassen können. Alles dies führt, abgesehen von

wenigen Details, ein Interpreter für Sie aus. So wurde Microsoft BASIC populär, obwohl es weitestgehend Merkmale und Instruktionen von EBASIC und CBASIC duplizierte.

Um 1980 gab Microsoft eine Compilerversion ihres Extended Disk BASIC heraus. Ein eindeutiger Vorteil des Microsoft BASIC ist, daß für die Programmentwicklung der Interpreter benutzt werden und die letzte Version kompiliert werden kann. Es herrscht die Meinung vor, daß mit Microsoft BASIC die derzeitig schnellste Ausführung von BASIC gefahren werden kann.

Computer gibt es seit den 50er Jahren, und wie Computer haben sich auch ihre Sprachen weiterentwickelt. Durch CP/M konnten viele Sprachen in den Gebrauch für Mikrocomputersysteme einbezogen werden.

Microsoft gab COBOL und FORTRAN heraus, bald nachdem CP/M verfügbar war. Da diese Sprachen zu den populärsten auf größeren Systemen gehören, war es logisch, daß sie als erste für die Arbeit unter CP/M entwickelt wurden. Nahezu zwei Jahre lang konnten CP/M-Benutzer nur zwischen verschiedenen BASIC-Dialekten, COBOL und FORTRAN wählen, aber mit der Zeit wurden mehr Sprachen zur Auswahl gestellt.

Fast jede höhere Programmiersprache kann benutzt werden, um eine gegebene Aufgabe auszuführen – dennoch sind sie sehr unterschiedlich. Deshalb wollen wir jetzt die Gründe diskutieren, eine Sprache vor einer anderen auszuwählen und ein Kurzprofil populärerer Sprachen aufzuzeichnen.

Auswahl einer Sprache

Wenn Sie eine Sprache auswählen, sollten Sie die beiden folgenden Regeln beachten:

1. Viele Anwenderprogramme erfordern für ihre Arbeit Laufzeit-Module (run-tim modules). Manchmal stellen die Händler die notwendige Software zur Verfügung, aber meistens gehen sie davon aus, daß Sie selbst dafür Sorge tragen. Ein typisches Beispiel sind Programme in einem BASIC-Dialekt, besonders CBASIC. Viele exzellente Buchhaltungsprogramme sind heutzutage in CBASIC geschrieben, aber diese Programme benötigen das Modul CRUN. Deshalb müssen Sie die CBASIC-Sprache extra kaufen, um ein in CBASIC geschriebenes Programm laufen zu lassen, es sei denn, Ihr Verkäufer hätte bereits CRUN in das Software-Paket eingeschlossen.
2. Wenn Sie bereits mit einer höheren Programmiersprache vertraut sind, wählen Sie für den Gebrauch unter CP/M eine ähnliche Sprache, wenn nicht die Aufgabenbereiche sehr weit gefächert sind.

Wenn Ihre Erfahrung mit Computern begrenzt ist, oder wenn Sie eine andere Sprache lernen möchten, um Ihre Neugier zu befriedigen oder Ihren Wissensstand zu erweitern, lesen Sie die nächste Sektion gründlich. Dies wird

Ihnen helfen zu entscheiden, welche Sprache(n) Ihren Zielen am ehesten gerecht werden.

Ein Lexikon der Sprachen

Es folgen Kurzbeschreibungen derzeit gebräuchlicher Sprachen für Operationen unter CP/M. In Anhang G finden Sie eine Liste der Firmen, die Sprachen zur Zeit anbieten. Diese Liste ist keineswegs vollständig, denn ständig werden neue Gesellschaften gegründet und die Angebote bestehender Firmen erweitert und dem Markt angepaßt. Journale wie **BYTE** und **Personal Computing** halten Sie über den letzten Software-Entwicklungsstand auf dem Laufenden. Ein bedeutender Faktor in bezug auf die Software von Microsoft ist, daß dort, anders als bei den meisten Gesellschaften, keine Handbücher separat verkauft werden. Wenn Sie ein Software-Paket kaufen, stellen Sie sicher, daß Sie mit der Software auch die Handbücher auffrischen.

ADA

ADA ist eine neue, vom Verteidigungsministerium entwickelte Sprache. Sie ist nach Augusta Ada Byron benannt, die als „erste Programmiererin" angesehen wird. ADA wurde entwickelt, weil alle anderen vom Ministerium bewerteten Sprachen (Pascal, PL/I, COBOL, FORTRAN usw.) den Anforderungen militärischer Software-Systeme nicht entsprachen. Alle Programme für das Department müssen nun in ADA geschrieben sein. Es ist eine hochstrukturierte und kultivierte Sprache, die für alle Typen von Programmen einfacher Anwendung bis zu hochtechnischen Systemen optimal ist.

Assembler-Sprachen

Außer dem Assembler, den Digital Research für CP/M anbietet, sind heutzutage mehrere fortgeschrittene Assembler-Sprachen verfügbar. Wenn Sie mit der Materie vertraut sind, kann ein fortgeschrittenes Produkt seine Anschaffung wert sein. Sollte Ihr System mehr auf einer Z80- als auf einer 8080-Maschine basieren, so prüfen Sie es nach, weil Ihnen ein fortgeschrittener Assembler erlaubt, direkt in Z80- wie auch 8080-Codes zu programmieren, wobei Sie aus dem stärkeren Instruktionssatz von Z80 Nutzen ziehen.

BASIC

BASIC steht für „Beginner's All-purpose Symbolic Instruction Code". Diese Sprache wurde in den 60er Jahren von Dartmouth College entwickelt, um Computergebrauch und -programmierung zu lehren. Obwohl BASIC teilweise von FORTRAN abgeleitet wurde, hat es eine einfachere Syntax; es ist für den gelegentlichen Leser von Programmen verständlicher.

Es ist leider schwierig, BASIC zu generalisieren. Da gibt es eine Definition des BASIC-Instruktionssatzes vom American National Standards Institute (ANSI), aber dieser Standard wurde definiert, nachdem die Sprache sich

selbst als populärste Programmiersprache für Mikrocomputer etabliert hatte. Zu dieser Zeit waren die Standards bereits gesetzt, und nahezu jede existierende Version des BASIC wich davon ab. Tatsächlich fügen Sprachentwickler BASIC häufig Instruktionen anderer Sprachen hinzu (vor allem von Pascal), um Schwächen zu überwinden.

Hier sind in einfachsten Begriffen einige Vorteile von BASIC aufgeführt:

- Es ist weitestgehend verfügbar; nahezu jeder Mikrocomputer benutzt eine Version von BASIC.
- Es ist leicht erlernbar. In Hunderten von Büchern wird es angeboten. Viele Schulen benutzen BASIC für einführende Programmierkurse, und die meisten Computereinzelhändler bieten es an.
- Es ist leicht zu verstehen; BASIC benutzt dem Englischen angepaßte Worte und Phrasen. Wie Radio Shack, Commodore und Apple durch Computerverkäufe an graduierte Schulen aufgezeigt haben, können Kinder mit geeigneter Anleitung die Kommandos lernen und Programme schreiben.
- Eine große Anzahl von in Computer-Magazinen publizierten Programmen ist in BASIC geschrieben, und sie geben so dem BASIC-Benutzer Beispiele für Programmierstil und -logik.
- Die meisten BASICs sind Interpreter; Sie können eine Instruktion tippen und sofort das Resultat sehen. Das macht BASIC zu einem ausgezeichneten Lehrmittel.
- Es gibt mehr BASIC-Dialekte als andere Sprachen (jedenfalls für CP/M-Systeme). Wahrscheinlich werden Sie einen finden, der zu Ihnen paßt.

Einige der Nachteile von BASIC sind:

- Es gibt keine eigentliche Standardisierung. Während die am häufigsten benutzten Instruktionen in allen BASICs generell konstant bleiben, gibt es doch viele Erweiterungen und sogar hinzugefügte Instruktionen. Diese können den individuellen Programmierer erfreuen, sie erschweren jedoch die Standardisierung.
- BASIC-Programme arbeiten langsam, verglichen mit anderen Sprachen. Nicht eingeschlossen sind hier gelegentliche Compiler-Versionen von BASIC, wobei doch die meisten BASICs die Interpretierungsebene hinzufügen. Aber selbst BASIC-Compiler generieren Programme, die langsam arbeiten. Die Struktur der Sprache und des resultierenden Programms machen vom Kernspeicher keinen optimalen Gebrauch.
- Sie können ein BASIC-Programm nachlässig schreiben und es wird trotzdem arbeiten. Für BASIC ist nicht viel Struktur erforderlich. Leider fördert BASIC dadurch auch einen schludrigen Programmierstil („off the top of the head").
- Standard-BASIC hat mit anderen Entwicklungen in der Computerindu-

strie nicht Schritt gehalten. BASIC ist über 20 Jahre alt, und Computer haben sich in dieser Zeit radikal geändert. Die begleitenden BASIC-Versionen sind kaum noch standardgemäß.

Trotz alledem ist BASIC wahrscheinlich die brauchbarste Sprache für CP/M-Benutzer. Die vielen in BASIC geschriebenen Programme, die vielen darüber geschriebenen Bücher und seine weite Verbreitung: das alles wird Ihnen helfen, die Sprache zu verstehen, wenn Sie sie brauchen.

C

C wurde von Bell Laboratories entwickelt; es ist ein integraler Bestandteil des Betriebssystems von Bell: UNIX. Zur Zeit werden mehrere Cs für CP/M angeboten, und wahrscheinlich wird ihre Anzahl noch zunehmen, wenn diese Sprache noch populärer wird.

Die meisten Programmiersprachen sind relativ dicht: C, ALGOL, PL/M und Pascal. Diese vier erfordern alle **strukturierte Programmiertechniken**. In Begriffen der Instruktionsausführung sind strukturierte Programme einfach nachzuvollziehen, aber nicht aus der Westentasche zu schreiben.

Die Sprache C unterscheidet sich in mehrfacher Hinsicht von anderen Sprachen. Sie benutzt oft Abkürzungen oder kurzgefaßte Instruktionen, wo Pascal und andere Sprachen mit kompletten Worten und Phrasen arbeiten. Z. B.:

```
C       INT X;              aber
Pascal X: INTEGER;
```

Im übrigen erfordert C mehr Verständnis und Gedächtnisleistung von seinen Anwendern, jedoch weniger Tippaufwand.

Obwohl es nicht schwer zu erlernen ist, funktioniert C exakt. C zeichnet sich durch seinen Instruktionssatz klar gegenüber anderen Sprachen aus, indem es durch seine Erweiterungen befähigt wird, viele Aufgaben zu erfüllen, die sonst nur in Assembler gelöst werden können.

Mit C können Module für mehrfachen Gebrauch entwickelt werden, ohne die Instruktionen nochmals aufzurufen. Dies kann sowohl intern (durch vom Benutzer definierte „Prozeduren") als auch extern (durch die in C eingebaute #INCLUDE-Funktion) geschehen. Wiederholte Routinen erfordern also nicht jedesmal den gleichen Programmieraufwand.

Außerdem ist C übertragbar. Da es von Bell Laboratories geschaffen wurde, leidet es nicht an der Aufsplitterung in unzählige Versionen. Ihr „C" wird bei einer viel größeren Anzahl von Mikrocomputern laufen als Programme, die in irgendeiner anderen Sprache geschrieben worden sind.

Der Hauptnachteil von C ist, daß es so neu ist. Zum Erlernen gibt es nur wenige Grundlagen. Für das CP/M-Environment werden zur Zeit nur wenige Anwenderprogramme verkauft oder propagiert.

COBOL

COBOL ist die Abkürzung von „Common-Business-Oriented Language" (= allgemeine geschäftsorientierte Sprache). COBOL hört sich wie normales Englisch an. Selbst COBOL-Sequenzen werden als **Sätze** interpretiert und benutzen eine Anzahl vorher definierter Wörter.

Hier ein Beispiel:

```
PROCEDURE CALCULATION.
DETERMINE-COST.
  COMPUTE OUR-COST = LIST-PRICE – 1Ø.Ø5
  IF OUR-COST > ZERO
    SET PRICE-TO-US TO OUR-COST
    MOVE „OK" TO VALID-PRICE-CODE
  ELSE
    MOVE „NO" TO VALID-PRICE-CODE.
```

Sie sehen, daß die Sätze mit einem Punkt enden. Indem Sie sie laut lesen, erkennen Sie, was das Programm tut. COBOL enthält in der Sprache wenig unverständlichen Computer-Code.

Der Ursprung von COBOL ist in der Computerwelt einzigartig. COBOL ist eine von einem Komitee entworfene Computersprache. Sie wurde von einer Gruppe namens CODASYL (Conference On Data Systems Languages) etabliert, die auch eine standardisierte Datenbasisstruktur für geschäftsorientierte Programmierung definiert hat. Daher ist COBOL im Gegensatz zu den anderen in diesem Buch beschriebenen Sprachen extrem standardisiert. Ein in ANSI COBOL (die standardisierte Definition) geschriebenes Programm sollte auf jedem Computer laufen, der diese Sprache benutzt. Die Regierung der United States überwacht Standard-COBOL-Versionen und gibt periodisch öffentliche Auskunft, ob COBOL-Versionen dem akzeptierten Minimalstandard gerecht werden. Gegenwärtig ist COBOL eine der wenigen Sprachen, die von der US-Regierung auf Standard-Definition hin geprüft werden.

Da COBOL im Hinblick auf geschäftsorientierte Anwendung entwickelt wurde, würde man auch annehmen, daß sie in erster Linie Geschäftszwecken dient. Für die größeren Computer stimmt das auch, aber in der Welt der Mikrocomputer haben einige Faktoren den Gebrauch von COBOL geschmälert.

Erstens ist COBOL eine umfangreiche Sprache. COBOL benutzt ganze Sätze, um eine Aufgabe durchführen zu lassen, und es ist eine sorgsam durchstrukturierte Sprache. Die Statements müssen in einer bestimmten Reihenfolge erscheinen und auch eine Anzahl systemspezifischer Statements muß eingeschlossen sein. Infolgedessen ist COBOL nicht besonders gut an Mikrocomputer angepaßt. Dies traf besonders früher zu, als Kernspeicher eine relativ kostspielige Komponente innerhalb von Mikrocomputer-Systemen waren. Mit der Senkung der Kernspeicherpreise (48K- und 64K-Mikrocomputer sind üblich) sind verschiedene Mikrocomputerversionen von COBOL erschienen.

COBOL ist nicht sehr effizient; die CPU eines Mikrocomputers braucht viel Zeit, um die langen Programm-Statements überhaupt zu entziffern. Dies bezieht sich in erster Linie auf die Kompilierung, in zweiter Linie auch auf die Programmausführung. Doch auch hier ist COBOL, verglichen mit anderen Sprachen, relativ langsam. Auf größeren Computersystemen fiel dies durch die höhere Ausführungsgeschwindigkeit nicht so sehr auf und Aufgaben wurden oft im Batch-Modus (Stapelverarbeitung) abgehandelt, weniger als interaktive Prozesse. Mikrocomputer hingegen werden oft nach der Anzahl interner Manipulationen durch COBOL taxiert.

COBOL wurde zu einer Zeit entworfen, als interaktive EDV relativ unbekannt war. Es kann keinen effizienten Gebrauch von der Konsoleinheit in einem CP/M-System machen. Beide, Microsoft COBOL und CIS COBOL, die beiden populärsten Microcomputer-Versionen der Sprache, schließen Extra- und insofern Nichtstandard-Instruktionen ein, um Gebrauch von den hochschnellen Konsoleinheiten zu machen, die für die meisten erhältlichen Mikrocomputer charakteristisch sind.

FORTH

FORTH ist eine der am meisten mißverstandenen Sprachen für Mikrocomputer. Als eine relativ neue Sprache (in den frühen 70er Jahren von Charles Moore erfunden) ist FORTH als alles von „einer Assemblier-Sprache wie BASIC" bis hin zu „einer Religion" beschrieben worden.

Ein FORTH-Programm ist nicht einfach zu entziffern; es ist eine **gewundene** Sprache. Sie benutzen seine Basis-Aufbaublocks, um größere daraus zu formen. Es ist in der Tat schwierig, zu beschreiben, wie man in FORTH programmiert. Betrachten Sie das folgende Programm (abgedruckt aus **BYTE**, August 1980, Seite 158):

```
 Ø BREAKFORTH/MMSFORTH, BY ARNOLD SHAEFFER, PART 5 OF 6)
 1
 2 : CLR
 3  XPOS @ 2– 124 AND 2+ DUP 4 + SWAP DO YPOS @ I DCLR LOOP
 4  YPOS @ 27 – ABS SCORE+ ! Ø 32 PTC SCORE ? BOP
 5  YDIR @ MINUS YDIR !
 6 :
 7
 8 : BALLCHKY DIR @ YPOS+ ! XDIR @ XPOS + ! XCHK YCHK PCHK
 9              YPOS @ XPOS @ D? IF CLR THEN
10 :
11
12 : BALL YPOS @ XPOS @ DCLR
13              BALLCHK DUP Ø = IF YPOS @ XPOS @ DSET THEN;
14
15 : GAMECHK SCORE @ 18ØØ MOD Ø= IF 191 15616 32Ø FILL THEN;
```

Ganz hübsch einschüchternd, nicht wahr? FORTH schließt viele Aspekte ein, mit denen sich ein Anfänger in einer Computersprache am liebsten nicht beschäftigen möchte. Es benutzt häufig Abkürzungen; numerische Manipulationen werden „polnisch rückwärts" notiert (umgekehrte, klammerfreie Praefix-Schreibweise ähnlich wie bei einigen Taschenrechnern); und FORTH erlaubt dem Benutzer die Erfindung neuer Kommandos (hier z. B. BALLCHK).

Warum existiert FORTH, wenn es so viele offensichtliche Nachteile hat? Einmal ist es schnell und besonders an Anwendungen angepaßt, die schnelle Bildschirm- oder Plattenmanipulationen erfordern. Zum zweiten ist der von FORTH generierte Maschinen-Code genrell viel kleiner als bei anderen Sprachen (ein weiterer Pluspunkt für die Schnelligkeit). Und schließlich ist FORTH die Sprache, die Sie daraus machen. Es ist ein starker Vorteil, daß der Programmierer die Sprache nach Belieben ausdehnen kann; Programme können auf die Aufgabenstellung zugeschnitten werden.

Wenn Sie an einer genaueren Beschreibung der Eigenschaften von FORTH interessiert sind, lesen Sie „**Discover FORTH**"*.

FORTRAN

FORTRAN ist eine weitere Abkürzung, es steht für „Formula Translator Language" (Formelübersetzer). Es wurde für komplexe mathematische Berechnungen entworfen, bei denen Geschwindigkeit der wichtigste Faktor ist. FORTRAN ist nicht gut in der Behandlung von Zeichen (z. B. Buchstaben des Alphabets) noch effizient im Umgang mit Ein-/Ausgabeeinheiten; auch für wechselwirksamen (interaktiven) Gebrauch wurde es nicht entworfen (obwohl es modifiziert werden kann).

In vieler Hinsicht ist FORTRAN ein Elternteil von BASIC; viele BASIC-Statements sind Abkömmlinge von FORTRAN. Die Hauptunterschiede liegen in der günstigen Behandlung von Konsol-Ein- und -Ausgaben durch BASIC und seinem geschickten Umgang mit Zeichenketten. In FORTRAN muß das schwer zu verstehende FORMAT-Statement angewandt werden, um Information für die Übertragung auf die Konsole oder andere Einheiten vorzubereiten.

FORTRAN ist wie C und COBOL ein wirklicher Compiler. Programme werden mit Hilfe eines Editors eingegeben und dann auf Maschineninstruktionen reduziert. Der für Mikroprozessoren am häufigsten benutzte FORTRAN-Compiler ist die Microsoft-Version, ein Teil von FORTRAN IV. Neue Programme werden selten in FORTRAN geschrieben; hauptsächlich wird es gebraucht, um vorhandene Programme auf einem Mikrocomputer laufen zu lassen.

* Thom Hogan, **Discover Forth** (Berkeley: Osborne/McGraw-Hill, 1982).

Pascal

Wie C und COBOL ist Pascal eine strukturierte Sprache; Programme müssen also einem festgelegten Strukturkonzept folgen. Statements müssen in festgelegter Reihenfolge ausgeführt werden; es können Blocks von Programm-Codes identifiziert werden, um diese auch an anderer Stelle verwenden zu können. Der gleiche Block kann in verschiedene Programme eingebunden werden. Auch BASIC, FORTRAN und mehrere andere Sprachen erlauben dem Programmierer, innerhalb der Programmausführung von einer Reihe von Statements zu einem anderen Set in einem völlig verschiedenen Bereich zu springen. Die logische Gruppierung von Programmstatements in **Subroutinen** (Unterprogramme, die eine Funktion ausführen und dann die Kontrolle an das aufrufende Statement zurückgeben) ist in BASIC ebenfalls möglich, aber nicht mit der in Pascal angebotenen Flexibilität.

1968 von Niklaus Wirth in der Schweiz entwickelt, bezieht sich Pascal direkt auf **Algol**. Eine typische Sektion eines Pascal-Programmes könnte folgendermaßen aussehen:

```
VAR HOUR, MINUTES, SECONDS: INTEGER;
BEGIN
  HOURS:=1;
    MINUTES:=1;
    SECONDS:=15;
    REPEAT
      WRITELN(,TIME LEFT =')
      WRITE(HOUR, ,:')
      WRITE(MINUTES, ,:')
      WRITE(SECONDS, ,:');
      WHILE SECONDS>=Ø DO
        SECONDS:=SECONDS-1;
      END;
      IF SECONDS=-1 THEN
        SECONDS:=59;
        MINUTES:=MINUTES-1;
        IF MINUTES=-1 THEN
          MINUTES:=59;
          HOURS:=HOURS-1;
    UNTIL (HOURS=Ø) AND (MINUTES=Ø) AND (SECONDS=Ø);
    WRITELN(,TIME UP!');
END.
```

Betrachten Sie das Erscheinungsbild eines Pascal-Programms. Zum einen hat es eine festgelegte Struktur. Die Einrückungen sind nicht gefordert, aber sie empfehlen sich sehr, um die Struktur des Programmes zu überblicken. Zum anderen benutzt Pascal wie COBOL **Sätze**, deren Bedeutungen sofort erfaßbar sind. Im Gegensatz zu COBOL hat Pascal jedoch relativ wenig vordefinierte Statements und die Sätze tendieren mehr zur Kürze. Pascal-Programme

sind, was die Länge angeht, nicht ganz so unhandlich wie in COBOL, aber sie bleiben doch lesbar.

Auch hat Pascal ein Attribut, das als **Rekursivität** bekannt ist. Zu unterschiedlichen Graden haben Algol, PL/I-80, FORTH und C das auch. Es bedeutet, daß ein Programmblock sich selbst aufrufen und schachteln kann. Dies ist eine nützliche Eigenschaft für komplexe Prozeduren, die innerhalb eines Programms wiederholt auftreten. Lange Divisionen schließen eine rekursive Funktion ein; Sie machen wiederholt vom selben Konzept Gebrauch, bis Sie auf Null oder eine Periode kommen. Durch seine rekursive Natur ist Pascal für Aufgaben, die überwiegend wiederholte Prozeduren einschließen, besonders geeignet.

Die meisten Programmierer mögen Pascal, weil ihre Programmkonzeptionen direkt zu den strukturellen Auflagen von Pascal passen. Tatsächlich benutzen viele Programmierer eine Pseudo-Pascal-Schreibweise zur Darstellung des logischen Flusses der Programmausführung.

```
WHILE RESULT HOT ZERO DO
  COMPUTE NEW VALUE
  IF NEW VALUE > OLD
    THEN PERFORM FUNCTION X
  ELSE
    PERFORM FUNCTION Y
END WHILE
```

Der Anteil eines Pascal-Programmes, der der Logik des letzten Beispieles entspricht, könnte so aussehen:

```
WHILE RESULT NOT Ø DO
  FIRSTNUM:= SECONDNUM-OFFSET;
  NEWVALUE; ←————————————  übergibt die Steuerung an eine Routine,
  IF NEWRESULTRESULT THEN   die NEWRESULT errechnet
          X
   ELSE
          Y; ←————————————  X und Y repräsentieren Prozeduren,
                            die bei jeder Ansteuerung ausgeführt
END{WHILE}                  werden
```

Wir haben das Erscheinungsbild von Pascal-Programmen aus mehreren Gründen diskutiert. Pascal wird wegen seiner leicht verständlichen Struktur häufig als die Sprache der Zukunft gepriesen. Auch ist sie auf Grund ihrer Geradlinigkeit eine gute Sprache für Anfänger. Auch wenn Sie nicht versuchen, Pascal-Programme aus dem Ärmel zu schütteln, werden Sie doch in der Lage sein, laufende Programme zustandezubringen. Natürlich kostet es immer einiges an konzentrierter Anstrengung und an Zeit, die Programmierung eines Mikrocomputers zu erlernen; deshalb empfiehlt es sich, Anleitung zu suchen (viele Universitäten bieten unentgeltlich oder gegen geringe Kostenbeteiligung Kurse für Pascal-Anfänger an).

PL/I-80

Mit der Einführung von PL/I-80 Mitte 1980 durch Digital Research ist jetzt nahezu jede größere Programmiersprache für die Anwender von Mikrocomputern erhältlich. PL/I-80 kombiniert die Struktur von Pascal sowie den einfachen Aufbau von Pascal und BASIC mit der Fähigkeit, komplexe Operationen mit verschiedenen peripheren Einheiten durchzuführen.

PL/I-80 und ihr Cousin, Intel PL/M, werden öfters als Sprachen zur **System-Entwicklung** beschrieben, also Sprachen, die von Computerherstellern und Systemintegratoren benutzt werden, um neue Software zu entwickeln. In der Tat wurde EBASIC und ein guter Teil von CP/M in PL/M geschrieben, desgleichen mehrere andere Hochsprachen.

Der PL/I-Compiler von Digital Research benutzt einen Subset (= Untermenge), genannt Subset G, der Originalsprache. PL/I-Programme arbeiten schnell und sind generell verschieblich.

Andere Sprachen sind

RPG (Report Program Generator) wurde ursprünglich von IBM für Nicht-Programmierer entwickelt. Es ist weniger eine Sprache als eine Methode, Daten von einer Plattendatei zu selektieren und (u. U. mit Einschluß von Gruppenwechseln) in aufbereiteter Form auszugeben. Das einzig fertig verfügbare RPG (für Mikrocomputer) wird von Cromemco vermarktet; es läuft dort mit dem CDOS-Betriebssystem (s. Kap. 6). Wenn RPG Ihnen empfohlen wurde oder Sie eine einfache Methode haben möchten, Information zu erstellen, zu pflegen und aus einer Datenbasis zu selektieren, sollten Sie auch einige kultiviertere Systeme des Datenmanagements wie Selector (von MicroAp), PEARL (von Relational Systems International), HDBMS (von Micro Data Base) oder Dbase II (von Ashton-Tate) in Erwägung ziehen.

LISP (List Processing) ist eine interpretierende Sprache, die in Zusammenhang mit Untersuchungen über künstliche Intelligenz an der Stanford University entwickelt wurde. Ihre hauptsächliche Anwendung ist die Verarbeitung von Zeichenketten, für mathematische Anwendung ist sie wenig brauchbar.

APL steht für „A Programming Language". Der bescheidene Name deutet nicht im mindesten auf die immensen Fähigkeiten dieser Sprache hin. Mehr noch als Worte benutzt APL graphische Zeichen, um bestimmte Kommandos zu repräsentieren. Das folgende APL-Programm berechnet den Durchschnitt einer Nummernliste und demonstriert die Präzision von APL.

```
average
"Enter the numbers you want averaged"
NUM ← □
  SUM ← +/NUM
  ANS ← SUM -:- pNUM
  "The sum of your numbers is ";ANS
```

Beachten Sie den Gebrauch von Symbolen ($\sqrt{}$, □, ←), um ein Konzept wiederzugeben. Ein Statement wie NUM←□ spezifiziert eine ganze Reihe von Computer-Operationen. Die getippte Zahl wird jedesmal an die Variable NUM übermittelt, statt daß man jede Operation durch ein Statement bezeichnet. BASIC und andere Sprachen durchbrechen eine solche Begrenzung manchmal durch Statements, die die Ausführung anderer Anweisungen wiederholen.

Textverarbeitung

Eines der nützlichsten käuflichen Anwendungsprogramme ist ein **Textverarbeitungsprogramm** bzw. **Text-Editor**. Der Editor von Digital Research (ED.COM, s. Kap. 3) ermöglicht die grundlegenden Edit-Funktionen. Möglicherweise brauchen Sie für CP/M-80 und CP/M-86 aus folgenden Gründen einen vielseitigen Editor:

- Um Programme zu erstellen und auszugeben
- Um Daten zu erstellen und Dateien auszugeben
- Um kritische Situationen in Textdateien zu beherrschen
- Um Dokumente zu erstellen und zu editieren (Briefe, Artikel, Dokumentation usw.)
- Um Dokumente zu formatieren und zu drucken.

Sofort kommen einem einige Fragen in den Sinn. Die erste mag sein: „Warum muß ich das alles tun? Ich dachte, die Programme, die ich gekauft habe, würden alles übernehmen."

Ihr Computer erstellt und speichert in erster Linie Information. Eine Schreibmaschine tut dasselbe und benutzt dabei Papier als Speichermedium. Sie würden nicht versuchen, eine ernsthafte Angelegenheit ohne Möglichkeit der Fehlerkorrektur zu tippen, und genauso erfordert ernsthafte EDV eine Möglichkeit, Platteninformation zu korrigieren. Ein guter Editor übernimmt diese Funktion für Sie.

Sie können als Editor ED.COM verwenden. Digital Research verkauft außerdem ein begleitendes Programm zur Text-Formatierung namens TEX. Dennoch gelten ED und TEX als relativ simple Edit-Software. Heutzutage gibt es eine Anzahl extrem kultivierter Text-Editoren, die Sie in Ihre Überlegungen einbeziehen sollten.

Magic Wand (= Magisches Zepter). Magic Wand (demnächst als Peach Wand und auch als geringfügig modifizierte Version von SuperWriter erhältlich) ist ein sinngeladener, geradliniger Editor für Textverarbeitung. Er opfert weder einfache Anwendung noch das Spektrum der Möglichkeiten. Magic Wand besteht aus zwei Modulen, EDIT und PRINT.

Das EDIT-Modul besteht aus zwei Anwendungsarten: „**command**" und „**edit**". Um ein Dokument zu editieren, brauchen Sie nur einige wenige Kommandos zur Cursor-Bewegung (= Schreibmarken-Positionierung) so-

wie zur Zeicheneinfügung und -löschung zu erlernen. Das EDIT-Modul ist bildschirmorientiert; Sie sehen die durchgeführten Änderungen. Ein Kommando, bestehend aus einem Tastendruck, schaltet vom Edit- in den Command-Modus um. Im Command-Modus können Sie eine Anzahl anderer Instruktionen an Magic Wand geben. Die meisten dieser zusätzlichen Instruktionen verändern Information in Diskettendateien oder instruieren Magic Wand, sie zu struktuieren und auszugeben.

Das PRINT-Modul druckt das mit Magic Wand erstellte Dokument. Abhängig von diesem Dokument kann dies automatisch geschehen oder vor dem Druck zusätzliche Information von Ihnen erfordern.

Magic Wand kann in ein zu druckendes Dokument Information von verschiedenen anderen Dateien einschießen. Dies ist generell für die Generierung von Formbriefen oder für die Beigabe einer persönlichen Note zu einem Standarddokument nützlich. Magic Wand kann Namen und Adressen in einen Brief einfügen und genausogut Umschläge drucken.

Electric Pencil (= Elektrischer Bleistift). Elektric Pencil war eines der ersten dynamischen bildschirmorientierten Textverarbeitungsprogramme für das Umfeld der CP/M-80. Mehrere Versionen sind erhältlich. Sie brauchen besonders wegen des kernspeichergesteuerten Videosystems in Ihrem Computer die für Sie passende Version. Electric Pencil wird nicht mit Serien-Terminals arbeiten.

Einmal geladen, schreibt Electric Pencil eine Copyright-Meldung auf den ansonsten leeren Bildschirm. Danach können Sie den Eingabe-, Platten- oder Druck-Modus wählen. Den Eingabe-Modus erhalten Sie, indem Sie anfangen zu tippen, den Platten-Modus durch ^K, den Druck-Modus durch ^P.

Von Electric Pencil erstellte Dateien sind nicht direkt für CP/M-80 brauchbar, weil Electric Pencil CARRIAGE RETURN ohne LINE FEED in den Text einfügt. Wenn Sie mit Electric Pencil Programme oder andere Dateien erstellen wollen, brauchen Sie das Programm CONVERT von der Michael Shrayer, Inc. Auch von der CP/M-Anwendergruppe wird ein Konvertierungsprogramm angeboten.

WordStar. WordStar ist ein ausgesuchten Wort-Prozessoren nachempfundenes Textverarbeitungs-Programm. Dann und wann erscheint ein solches Programm, das speziell für Computerneulinge besonders geeignet ist.

Nachdem Sie WordStar einmal in den Computer geladen haben, können Sie den Umfang des Hilfs-Menu spezifizieren, mit einer Kommandoliste, die jeweils nach Ausführung eines Kommandos angezeigt wird. Die Tatsache, daß das Programm hervorragende Bildschirmhilfen gibt, ist allerdings kein Grund, das Lesen des Handbuches zu unterlassen.

Vor der ersten Anwendung müssen Sie WordStar Informationen über Betriebssystem und Gerätebelegung geben. Benutzen Sie das zusammen mit WordStar erhältliche Programm INSTALL. Wenn Sie eine „Standard"-

Ausrüstung haben, ist der Installationsprozeß geradlinig. Wenn Ihre Ausrüstung von INSTALL nicht als eine der vordefinierten Wahlmöglichkeiten ausgewiesen wird, konsultieren Sie Ihren Computerhändler oder WordStar-Verkäufer.

Dieses Buch wurde mit WordStar geschrieben. Drei Eigenschaften bestimmten die Wahl von WordStar: die Fähigkeit, das Textformat auf dem Bildschirm so aufzuzeigen, wie es dann gedruckt wird, die Fähigkeit, die für zahlreiche wichtige Überarbeitungen notwendigen Manipulationen vorzunehmen und die Tatsache, daß der Buchsatz direkt von den Disketten des Autors erfolgen kann.

SELECT, Benchmark, Memorite, Electric Blackboard. Seit der ersten Ausgabe dieses Buches sind eine Anzahl anderer guter Textverarbeitungsprogramme erschienen. Sie haben alle ihre guten Seiten, und Sie sollten alle prüfen, bevor Sie sich für eines entscheiden.

Die Hauptvorteile von SELECT sind seine einfache Anwendung und die eingebaute Möglichkeit, sie sich selbst beizubringen. In anderer Hinsicht ist SELECT WordStar sehr ähnlich. Auch Benchmark hat viele ähnliche Eigenschaften wie WordStar, und einige Anwender ziehen seine einfacheren Edit-Kommandos vor. Memorite ist ein extrem flexibles und überzeugendes Textverarbeitungsprogramm, aber im Zusammenhang mit Systemen der Vektor-Grafik eingeschränkt. Elektric Blackboard ist insofern einzigartig, als es dem Anwender erlaubt, den Bildschirm in viele „Fenster" aufzuteilen, die individuell manipuliert werden können. Außerdem eignet sich Electric Blackboard am besten dazu, Text mit mehr als der normalen 65-Zeichen-Grenze auf 8½ × 11-Zoll-Papier zu drucken.

Kapitel 6
MP/M, CP/NET und CP/M-Verwandte

Die große Verbreitung von CP/M hat eine Anzahl ähnlicher Betriebssysteme hervorgebracht. Wie Cromemco's CDOS sind einige dieser Systeme direkte Ableitungen von CP/M, leider aber nicht total kompatibel. Andere „work-alikes" (= ähnlich arbeitende), z. B. I/OS und TP/M behalten CP/M-Kompatibilität und beanspruchen, die charakteristischen Merkmale des CP/M zu übertreffen. So erweiterte Digital Research die Fähigkeiten von CP/M mit MP/M und CP/NET.

Die CP/M-Verkaufsraten liegen über denen aller anderen Mikrocomputer-Betriebssysteme, inkl. Apple DOS und Radio Shack TRSDOS. CP/M-80 ist jetzt für Apple wie Radio Shack lieferbar und CP/M-86 mit einem zusätzlichen 8088-Prozessor auch für den Computer Apple II.

Aus der Popularität von CP/M resultierte eine wachsende Bibliothek CP/M-kompatibler Software. Verglichen mit Anwenderprogrammen für andere Betriebssysteme, ist CP/M-kompatible Software für geschäftsorientierte Anwendung weit verbreitet.

Vielleicht haben Sie CP/M bewußt gewählt, vielleicht hatten Sie keine Wahl oder Sie haben ein anderes Betriebssystem gekauft in der Überzeugung, daß es Ihren Anforderungen besser angepaßt sei. Jedenfalls sollten Sie die Beziehung zwischen Ihrem Betriebssystem und CP/M erforschen um zu prüfen, inwieweit Ihr Computer mit CP/M-kompatibler Software umgehen kann.

Der Grad der Kompatibilität zwischen CP/M und anderen Betriebssystemen variiert. So kann CDOS nicht auf einem auf der 8080 basierenden System gefahren werden, weil CDOS einige Z80-Instruktionen enthält. Außerdem enthält CDOS gewisse Erweiterungen; Programme, die davon Gebrauch machen, laufen nicht mit CP/M. Andere Betriebssysteme wie I/OS und TurboDOS erheben den Anspruch, verschiedene CP/M-„Fehler" zu korrigieren, aber dennoch CP/M-kompatibel zu sein. Manche, wie 86-DOS sind nicht wirklich kompatibel, denn sie benutzen ein anderes Dateienformat als CP/M-80 und CP/M-86.

Irrtümliche Annahmen in bezug auf die Kompatibilität von Betriebssystemen und Anwenderprogrammen führen stets zu Problemen. In diesem Kapitel werden deshalb klärende Informationen über die Kompatibilität ausgewählter Betriebssysteme geliefert und die Charakterzüge von MP/M und CP/NET, zweier Erweiterungen der CP/M-Familie von Digital Research, diskutiert.

Gemeinschaftsbenutzer- und Multitasking-Systeme

Bis jetzt haben wir Computersysteme beschrieben, die jeweils eine Aufgabe für einen Benutzer ausführen.

Multitasking-Computersysteme erfüllen mehrere Pflichten gleichzeitig. Andere Namen, die sich auf so eine Multitasking-Computereinrichtung beziehen, sind beispielsweise Multiuser (= Gemeinschaftsbenutzer), Timesharing (= Teilnehmer-Rechnersystem), Time-slicing (= Zeitunterteilung), Multiprogramming (= Mehrprogrammbetrieb) und Multiterminal (= Anschluß mehrerer Terminals). Leider hat die Mikrocomputerindustrie einige Definitionen geändert, die sich vorher auf Minicomputer und Mainframe-Computersysteme (Hauptrahmen einer CPU) bezogen haben. In diesem Text bezieht sich Multitasking auf ein Mikrocomputersystem, das zwei oder mehr Aufgaben gleichzeitig ausführt.

Wie arbeitet ein Multitasking-Computersystem? In einem normalen Computer tritt für jede Aufgabe (Task] eine Folge von Ereignissen (Events) auf. Wenn Sie z. B. eine Taste der Tastatur drücken, tritt die folgende (vereinfachte) Sequenz auf:

1. Sie drücken die Taste
2. Die Konsole sendet das Zeichen in den Computer
3. Die I/O-Einheit innerhalb des Computers empfängt das Zeichen
4. Die CPU übernimmt das Zeichen von der I/O-Einheit
5. Die CPU verarbeitet das Zeichen gemäß dem bereits im Kernspeicher befindlichen Instruktionssatz.

Während sich die Schritte 1 bis 3 auf einer Tastatur abspielen, könnte die CPU Schritt 5 (Verarbeitung eines Zeichens) für eine zweite Konsole ausführen. Denn während die CPU auf das Zeichen von der ersten Tastatur wartet, ist sie müßig.

Multitasking fordert also in seiner einfachsten Form, daß die CPU eine zweite Aufgabe aufnimmt, während sie auf die Fortführung der Verarbeitung für die erste Aufgabe wartet. Es scheint so, als ob der Computer zwei Dinge gleichzeitig täte, denn die Zeittakte für solche Aufgaben werden in Tausendstel oder Millionstel von Sekunden gemessen.

Wäre alles so einfach wie in unserer Beschreibung, hätten alle Computersysteme die Einrichtung zum Multitasking. Wir müssen ein Problem ansprechen, um das Konzept multipler Aufgaben auf einem einzelnen Computer völlig zu beschreiben. Das Problem, einfach ausgedrückt, lautet: Woher wissen Sie, daß zwei Aufgaben sich nicht in die Quere kommen? Woher weiß die CPU, wann sie an welcher Aufgabe zu arbeiten hat?

Die Antwort ist, daß Computer-Designer Standardvoraussetzungen vorgeben, und normalerweise eine von zwei konventionellen Annäherungsmethoden wählen.

- Jedem Benutzer wird ein **Zeitabschnitt (time slice)** zur Verfügung gestellt. Wenn dieser Abschnitt von CPU-Zeit eine bestimmte Länge hat, werden die beiden Benutzer des Systems die Gegenwart des anderen auf dem Computer

kaum bemerken noch eine merkliche Verlangsamung des Systems. Vielfach muß der Computer auf eine Benutzeraktion warten. Multitasking hat daher gewisse Vorteile. Bei I/O-orientierten Systemen, wo viel Zeit für die Ein- und Ausgabe von verschiedenen Einheiten verbraucht wird – z. B. ein Textverarbeitungssystem – wartet der Computer auf das Drücken von Tasten; so ein System ist sicher ein guter Kandidat für **Time-slicing**.

- Die CPU kann auch jedesmal dann Aufgaben umschalten, wenn sie auf eine andere Einheit warten muß. Eine spezifische Methode verarbeitet eine Aufgabe, bis eine andere die Aufmerksamkeit der CPU erfordert (dieses wird ein **interrupt-driven** (= unterbrechungs-gesteuertes) System genannt). Die Unterschiede zwischen diesen beiden Methoden sind subtiler als sie zunächst erscheinen.

Diese Annäherungen mögen zufriedenstellende Methoden für die Teilung von CPU-Zeit sein oder auch nicht. Wenn die CPU in einer Aufgabe selten auf I/O wartet, wird die andere möglicherweise für eine ganze Weile nicht aktiviert. Auf der anderen Seite werden, wenn eine Einheit konstante Aufmerksamkeit erfordert, die anderen Einheiten ignoriert. Das Resultat ist dasselbe: Eine Aufgabe bekommt Priorität über die anderen.

Diese Methoden sind ungenau als **polling systems** (= Sendeaufruf-Systeme) bezeichnet worden. „Polling" bedeutet die Prüfung, ob eine Einheit bereit ist. Ein Zeitabschnitts-System kann diese Prüfung einschließen; es ist jedoch generell möglich, Multitasking-Systeme ohne Polling zu implementieren.

Multitasking-Systeme von Mikrocomputern verbinden oft diese beiden Methoden. Jeder Benutzer hat einen Time-slice; wenn aber die CPU während des Zeitabschnittes arbeiten muß oder eine andere Einheit die Zuwendung erfordert, wechselt die CPU zu einer anderen Aufgabe, bevor der Zeitabschnitt zu Ende ist. Dies löst die Probleme, die aus unausgeglichenen Anforderungen an die CPU entweder durch die Einheiten oder den Operator resultieren.

Eine Art, wie Aufgaben die Zuwendung der CPU erfordern, schließt **Interrupts** (= Unterbrechungen) ein. Wenn Sie eine Taste drücken, muß ein Zeichen verarbeitet werden; Ihre Eingabe muß die CPU erreichen, bevor Sie die Taste loslassen. Es wird eine spezielle von der CPU gewartete Zeile aufgerufen und die CPU schaltet direkt zu der Routine um, die diesen Prozeß ausführt. Nach Ausführung der **interrupt task** (= der unterbrechenden Aufgabe) kehrt die CPU zur **interrupted task** (= der unterbrochenen Aufgabe) zurück.

Eine Methode des „Pseudo-Multitasking", daß nämlich mehrere miteinander verbundene Computer Einheiten teilen, ist als **networking** (= Netzwerk-Methode) bekannt. Dieses Konzept schließt zwei Überlegungen ein.

- Die Computer müssen durch ein physisches Mittel (wie eine Telefonverbindung oder ein simples Kabel) und ein Software-Mittel verbunden sein. Das Software-Mittel heißt **protocol** (= Protokoll).

– Die Verbindungsmethode zwischen den Computern hängt von den Interface-Möglichkeiten eines jeden Computers im Netzwerk ab. In einigen Fällen können alle Computer alle Einheiten aller anderen Computer benutzen. In anderen Systemen wird **hierarchisch** gearbeitet: eine Maschine fungiert in bezug auf mehrere andere Maschinen als Gastgeber, oder es sind nur bestimmte Verbindungen erlaubt.

Mit dieser Einführung ins Multitasking wollen wir unsere Diskussion nun den Eigenschaften von MP/M und CP/NET zuwenden.

MP/M

Generell betrifft diese Diskussion das MP/M II; jedoch werden die Benutzer der früheren MP/M-Version die meisten Informationen ebenfalls nützlich finden.

Die Buchstaben MP/M stehen für „Multi-Processor Monitor Control Program" (= mehrfach-verarbeitendes Monitor-Kontrollprogramm). MP/M ist ein Betriebssystem, das mehr als ein Konsolenterminal und mehr als ein Programm auf jedem Terminal kontrollieren kann. So kann jeder von mehreren Benutzern mehrere Programme „simultan" auf einem Computer laufen lassen.

Unterschiede zwischen MP/M und CP/M

Sie werden drei generelle Unterschiede zwischen dem Verhalten des MP/M und des CP/M bemerken: die Anforderung ist unterschiedlich, da sind neue Control-character (= Kontrollzeichen), und es gibt neue Kommandos. Lassen Sie uns diese Unterschiede zwischen MP/M und CP/M untersuchen.

1. **The prompt** (= die Anforderung).
   Wie CP/M, zeigt MP/M den Laufwerks-Identifikator des aktiv steuernden Laufwerks (A:, B:, C: usw.) gefolgt von einem „>" an, aber MP/M schließt auch die derzeitig aktive Benutzernummer (0 bis 15) vor dem Laufwerks-Identifikator ein.

CP/M prompt	A>Drive A is the Default (= Laufwerk A: ist vorgegeben)
(= Anforderung)	B>Drive B is the Default (= Laufwerk B: ist vorgegeben)
	C>Drive C is the Default (= Laufwerk C: ist vorgegeben)
MP/M prompt	0A>User 0 on drive A (= Benutzer 0 auf Laufwerk A:)
(= Anforderung)	4F>User 4 on drive F (= Benutzer 4 auf Laufwerk F:)
	3D>User 3 on drive D (= Benutzer 3 auf Laufwerk D:)

   Jede Benutzernummer ist mit einer Gruppe von Plattendateien verbunden.

Geben Sie ein USER-Kommando ein, um auf einen anderen auf Platte befindlichen Benutzerbereich überzuwechseln.

2. **Extra control characters** (= Extra-Kontrollzeichen). MP/M kennt mehr Kontrollzeichen als CP/M. Es sind

   ^D Detach console from current job (= Hänge die Konsole von der laufenden Aufgabe ab)
   ^Q Restart console after ^S pressed (= Starte Konsole nach ^S neu)
   ^Z End input from console (= Beende Eingabe von Konsole)

   CONTROL-D läßt Sie die Konsole von einer Aufgabe abhängen. Dies ist nützlich, wenn eine Aufgabe wenig oder keine Eingabe von der Konsole erwartet oder wenn Sie eine Aufgabe unterbrechen wollen, während Sie eine andere starten. Sie können eine Aufgabe durch ein ATTACH<cr> nach dem MP/M-Prompt weiterlaufen lassen.
   Das Zeichen „^S" stoppt wie in CP/M die Konsolenanzeige. Jedoch wird, anders als in CP/M, nur ein ^Q die Konsole nach dem Gebrauch von ^S erneut starten.
   CONTROL-Z beendet die Eingabe von der Konsoleinheit. Sie werden ^Z selten benötigen; wenn die Konsole wie eine Platteneinheit funktioniert, sendet ^Z einen „**end-of-file**-marker" (= Dateiende-Merker).

3. **Additional commands and utilities** (= Zusätzliche Kommandos und Dienstprogramme).
   MP/M hat die folgenden neuen Kommandos:
   DIR SYS
   Zeigt eine Directory **inklusive** Systemdateien an
   ERAQ
   Erlaubt einen Abfrage-Modus für die Löschung von Dateien
   CONSOLE
   Zeigt die Konsolennummer an
   DSKRESET
   Erlaubt dem Benutzer, Disketten zu wechseln
   GENHEX
   Erstellt eine Hex-Datei aus einer .COM-Datei
   PRLCOM
   Erstellt eine .COM-Datei von einer speziellen .PRL-Datei
   GENMOD
   Erstellt eine .PRL-Datei aus einer speziellen .HEX-Datei
   SPOOL
   Sendet Druckausgabe auf eine Spool-Einheit (= Speicher-Einheit zum Sammeln aufgelaufener Druckdateien zum späteren Ausdruck)
   STOPSPLR
   Stoppt den Spooler-Output
   TOD
   Setzt Uhrzeit und Datum oder zeigt sie an

SCHED
Steuert (Schedule) einen automatischen Arbeitsablauf
ABORT
Bricht eine Aufgabe ab, selbst wenn diese von der Konsole losgelöst ist
ATTACH
Läßt eine „abgehängte" Aufgabe weiterlaufen
MPMSTAT
Zeigt den MP/M-Status an
PRINTER
Wählt den zu benutzenden Drucker aus
SDIR
Zeigt Directory und Optionen an
SHOW
Zeigt den Platten-Status an
SET
Setzt den Platten- und Dateienstatus, Kernwörter (passwords) und Zeiteinteilung

Detaillierte Beschreibungen dieser neuen Kommandos folgen im nächsten Abschnitt.

MP/M-Kommandos

Wenn Sie MP/M kaufen, erhalten Sie eine Anzahl von nur zu MP/M gehörigen Programmen. Alle MP/M-Kommandos außer denen für die Kontrollzeichen sind transiente Kommandos oder Programme. Wir wollen einen kurzen Überblick über jedes MP/M-Programm geben.

DIR[SYS]
Tippen Sie „DIR*.*" gefolgt von einem Leerzeichen und einem „S" bei MP/M 1.1 oder „DIR[SYS]" bei MP/M 2.0, und Sie erhalten eine Directory, die **System**dateien einschließt. Systemdateien würden sonst nicht in der Anzeige der Directory erscheinen. Normalerweise werden alle Programmdateien in MP/M zu Systemdateien gemacht, so daß alle Benutzer Zugriff zu ihnen haben.

Zusätzlich zu den normalen DIR-Kommandos kennt MP/M 2.0 folgende Möglichkeiten:

DIR[G8]
Gibt die Directory von Benutzer 8 aus

DIR file,file
Gibt das Inhaltsverzeichnis mit den Datei-Spezifikationen der ersten und der zweiten Datei aus. Jede weitere Spezifikation muß durch ein Komma getrennt werden.

ERAQ
Das Kommando ERAQ ist eine Erweiterung des Kommandos ERA (erase)

von CP/M. Wenn Sie ERAQ anstelle von ERA eingeben, fragt MP/M nach jeder Datei; das „Q" steht für **query** (= Anfrage), zum Beispiel:

```
ØF>ERAQ *.*<cr>
F:IRAN      OIL? y
F:SAUDI     OIL? n
F:ISRAELI   MEN? n
F:AFGHAN    OIL? n
ØF>
```

In unserem Beispiel löschten wir IRAN.OIL, aber nicht die anderen drei Dateien auf der Diskette. ERAQ anstelle von ERA zu benutzen vermindert die Wahrscheinlichkeit irrtümlicher Löschungen.

CONSOLE

In MP/M können 16 unabhängige Konsoleinheiten angeschlossen sein. Diese Terminals sind numeriert.

Die Konsolnummer existiert unabhängig von der Benutzernummer, diese Unterscheidung ist wichtig. Jedem Terminal ist eine feste Konsolnummer zugeordnet. Die Benutzernummer dagegen ist mit einer Gruppe von Dateien auf Disketten gekoppelt.

Um zu sehen, welche Konsole Sie benutzen, tippen Sie

```
CONSOLE<cr>
```

MP/M antwortet

```
Console = x
```

wobei „x" Ihre Konsolennummer ist.

DSKRESET

Möglicherweise werden mehrere Benutzer sich ein oder zwei Plattenlaufwerke teilen. Was ist nötig, wenn ein Benutzer die Platte im Laufwerk wechseln oder ein anderes Programm laden muß? Bedenken Sie, nach Plattenwechseln mit CP/M müssen Sie einen Warmstart durchführen. Mit MP/M wollen Sie den Warmstart nur für Ihre Aufgabe durchführen, nicht für die aller anderen.

DSKRESET wechselt Disketten ausgewählt. Tippen Sie das Kommando ohne Parameter, um alle Plattenlaufwerke des Systems auf Null zu setzen. Lassen Sie dem Kommando eine Liste gültiger Laufwerks-Identifikatoren (A:, B: usw.) jeder durch ein Komma getrennt, folgen, um nur die spezifizierten Laufwerke auf Null zu setzen, Zum Beispiel wird das Kommando

```
DSKRESET E:,N:,D: <cr>
```

die Laufwerke „E:", „N:" und „D:" auf Null setzen.

MP/M erlaubt Ihnen nicht, ein Laufwerk auf Null zu setzen, wenn irgendeine Task (Aufgabe) Dateien auf diesem Laufwerk benutzt. Wenn das geschieht, sehen Sie die Meldung:

```
Disk reset denied, Drive x: Console y Program z
```

Diese Meldung informiert Sie, welches Konsol und Programm noch offene Dateien auf der Platte haben. Wenn Ihre Konsole spezifiziert wird, vollenden Sie Ihren Gebrauch des Programmes (zumindest beenden Sie den Zugriff auf die angesprochenen Dateien) und versuchen Sie dann DSKRESET neu. Wenn Ihre Konsole nicht gelistet wird, müssen Sie warten, bis die anderen Benutzer geendet haben, bevor Sie das Plattenlaufwerk auf Null setzen, oder Sie müssen nach einem anderen verfügbaren Laufwerk suchen.

Sie **müssen** DSKRESET benutzen, bevor Sie eine Platte aus dem System entfernen oder Sie riskieren, einem anderen Benutzer „unrecoverable errors" (= nicht-reparierbare Fehler) zuzufügen.

GENHEX

Dieses Programm ist das Gegenteil von LOAD; es liest eine Datei vom Typ „.COM" und erstellt daraus eine Datei vom Typ „.HEX", die man mit LOAD oder GENMOD aufrufen kann.

PRLCOM

Dateien, die mit dem Typen „.PRL" für MP/M bestimmt sind, können mit PRLCOM in Dateien (z. B. des Typs „.COM") umgewandelt werden, die in auf CP/M basierenden Systemen verarbeitet werden können. Das Format dieses Kommandos ist

```
PRLCOM prltype.fil comtype.COM<cr>
```

Diese Kommandozeile bricht aus der üblichen NEW = OLD-Konvention der meisten Computerkommandos aus.

GENMOD

In CP/M ändert LOAD eine vom Assembler kreierte „.HEX"-Datei in eine „.COM"-Datei um, die durch Tippen ihres Namens geladen und ausgeführt wird. MP/M lädt ein individuelles Anwenderprogramm an den Beginn des diesem Benutzer zugeordneten Speicherbereiches, aber „.COM"-Dateien werden stets beginnend mit 0100 hex geladen. GENMOD konvertiert eine „.HEX"-Datei in eine „.PRL"-Datei, die an irgendeine andere Kernspeicheradresse geladen werden kann.

GENMOD liest eine Datei, die aus zwei assemblierten Versionen eines Programmes (jede Assemblierung mit einer unterschiedlichen Startadresse) besteht, und erzeugt eine „.PRL"-Datei. Die beiden Assemblierungen mit unterschiedlicher Anfangsadresse sind erforderlich, da GENMOD die beiden resultierenden Sätze von Object-Code vergleicht, um herauszufinden, wo im Programm spezifische Adressen vorkommen. Da MP/M keine bestimmte Anfangsadresse für ein Programm annimmt, identifiziert es Teile des Programms, die bestimmte Adressen benutzen, so daß diese geändert werden können, um den Kernspeicher-Bereich, in den das Programm geladen wurde, zu entsprechen.

SPOOL

Mehrere Benutzer eines Systems können sich einen einzelnen Drucker teilen. Bei MP/M senden Sie Ihren Output normalerweise an eine Plattendatei anstatt direkt an den Drucker; dann benutzen Sie SPOOL, um die Datei zu drucken.

SPOOL handhabt eine **queue** (= Schlange) von Dateien, die gedruckt werden sollen. Wenn Benutzer 1 mit SPOOL den Druck einer Datei anfordert und Benutzer 2 etwas später nach einer gedruckten Liste verlangt, wird SPOOL zuerst den Druck der Liste für Benutzer 1 vollenden und danach die Information für Benutzer 2 drucken. Wenn Sie SPOOL aufrufen, listet es einen Überblick über die Warteschlange und kehrt dann mit der MP/M-Anforderung auf Ihre Konsole zurück. Sie sind frei für andere Aufgaben; der Drucker druckt automatisch Ihr Dokument entsprechend der Schlange. Um Ihre Dateien der Spooler-Schlange zu übergeben, tippen Sie

```
SPOOL filename.typ, filenamy.typ<cr>
```

Sie können eine beliebige Anzahl von Dateien, jede durch ein Komma getrennt, bis zur Begrenzung der Länge einer Kommandozeile hinzufügen.

STOPSPLR

Um Die SPOOL-Funktion zu stoppen, tippen Sie STOPSPLR gefolgt von Ihrer Konslennummer. Wenn Sie irgendwelche Dateien haben, die noch auf Ihren Druck warten, werden sie alle aus der Schlange getilgt. Dies ist eine drastische Aktion. Es kann auch in einer nur teilweise ausgedruckten Liste resultieren. Vergewissern Sie sich, daß Sie den Ausdruck Ihrer Dateien abbrechen wollen.

TOD

MP/M unterhält eine Uhr, die Datum und Zeit aufzeigt. Erwarten Sie nicht, daß diese absolut genau ist. Erstens startet MP/M, wenn der Computer für einen beliebigen Zeitraum – selbst den Bruchteil einer Sekunde – abschaltet, die Uhr neu. Zweitens hängt die Genauigkeit der Uhr davon ab, wie Ihr MP/M-System implementiert wurde. Wenn Sie einen Computer von Digital Microsystems oder ein MDS-basierendes System von Intel benutzen, wird Ihre Uhr genauso funktionieren, wie Digital Research es vorhatte. Andere Implementationen müssen nicht exakt funktionieren.

Um die Zeit zu setzen, tippen Sie

```
TOD mm/dd/yy hh:mm:ss<cr>
```

Geben Sie den Monat, Tag, Jahr, Stunden, Minuten und Sekunden in genau dieser Reihenfolge und diesem Format ein. Hier ist eine gültige Eintragung

```
TOD Ø1/16/83 1Ø:ØØ:ØØ<cr>
```

Dies sagt dem Computer: es ist der 16. Januar 1983, 10.00 Uhr. MP/M antwortet auf das Komando

```
Strike key to set time
THU Ø1/16/83 1Ø:ØØ:ØØ
```

Drücken Sie irgendeine Taste, um die Uhr zu starten und gehen Sie zurück ins MP/M. Nun sehen Sie jedesmal, wenn Sie TOD <cr> tippen, das laufende Datum und die Uhrzeit, wie MP/M sie wartet. TOD P<cr> zeigt beständig Datum und Zeit an, bis Sie eine andere Taste drücken.

SCHED

Eine Uhr erlaubt eine automatische Steuerung von Tasks. Der Computer könnte eine Aufgabe ausführen, wenn kein anderer das System benutzt. Dies ist besonders nützlich für lange Listen, die die Antwortzeit des Computers verlängern könnten. Für ein Task-Schedule tippen Sie

```
SCHED mm/dd/yy hh:mm task<cr>
```

wobei „task" eine gültige MP/M-Kommandozeile ist und „mm/dd/yy hh:mm" exakt Datum und Zeit angibt, an der die Ausführung der Aufgabe beginnen soll. Sie können die Ausführung von Tasks nicht in Zeitintervallen angeben, die kürzer sind als eine Minute.

ABORT

Um die Durchführung einer Aufgabe abzubrechen, tippen Sie ABORT task <cr>. Wenn die Task von einer anderen Konsole aus aufgerufen worden war, müssen Sie deren Nummer mit angeben.

```
ABORT task<cr>
```

Die interne Struktur von MP/M

MP/M ist einer Struktur in CP/M-80 sehr ähnlich. Die Ausrüstungsanforderungen von MP/M sind

- Eine 8080-, 8085- oder Z80-CPU (MP/M-86 benutzt eine 8088 oder 8086)
- Mindestens 32K Kernspeicher
- Mindestens ein Plattenlaufwerk
- Eine Tastatur für die Eingabe von Zeichen
- Ein Drucker oder CRT für die Ausgabe von Zeichen
- Eine uhrzeitgesteuerte Unterbrechung

Betrachten Sie, wie CP/M in den Kernspeicher geladen wird (siehe Kap. 1). Laden Sie MP/M mit einem Kaltstart-Lader wie bei CMP/M oder benutzen Sie den speziellen MP/M-Lader – ein transientes CP/M-Programm – um für den weiteren Gebrauch MP/M zu laden und auszuführen.

Die interne Struktur von MP/M ist ähnlich wie bei CP/M. Anstelle der CP/M-Bereiche

TPA

Transient Program Area (= Transienter Programm-Bereich)

CCP
Console Command Processor (= Prozessor für Konsolen-Kommandos)

BDOS
Basic Disk Operating System (= Basis-Platte-Betriebssystem)

BIOS
Basic Input/Output System (= Basis-Ein/Ausgabe-System)

hat MP/M die folgenden Teile:

TPA
Transient Program Area

MEMSEG
Zwischen 1 und 8 Kernspeicher-Segmenten, eines für jeden Benutzer

XDOS
Erweitertes Platte-Betriebssystem

BDOS
Basis-Platte-Betriebssystem des CP/M mit einem Umschaltungs-Dienstprogramm für Datenbanken

XIOS
Erweitertes Ein-/Ausgabe-System

Obwohl MP/M zusätzliche Möglichkeiten zur CP/M-Struktur hinzufügt, ändert es CP/M doch nicht signifikant. Erinnern Sie sich, MP/M wurde erweitert, um 16 Benutzerterminals, 16 Drucker und 16 Laufwerke, jedes mit einer Kapazität bis zu 512 MB (Megabytes), einschließen zu können. MP/M unterstützt nicht wie CP/M PUNCH- und READER-Einheiten. Wo CP/M-BIOS ein Benutzerterminal, einen Leser, einen Stanzer und einen Drucker beschreibt, beschreibt MP/M-XIOS Terminals und Drucker für jeden Benutzer. Mit anderen Worten, das Konzept wurde nicht geändert, aber das Betriebssystem wurde erweitert, um Information über alle Benutzer einzuschließen. Dies ist ein Grund, warum MP/M einen höheren Kernspeicherbedarf hat als CP/M.

MP/M-Funktionen unterscheiden sich geringfügig von denen des CP/M. CP/M lädt ein Dienstprogramm stets an die Anfangsadresse 0100 hex und speichert es in eine „.COM"-Datei. MP/M behält diese Eigenschaft bei, aber addiert die Fähigkeit, „.PRL"-Dateien (page relocatable = seitenverschiebbar) zu benutzen; ihre Speicherung und Ausführung beginnt an einem beliebigen Intervall von 0100 hex (0100, 0200, 0300 usw.).

Angenommen, zwei Benutzer hätten 48K zur Verfügung und jede Aufgabe erforderte nur 20K. Wenn beide Aufgaben an Speicherstelle 0100 hex beginnen müßten, entstünde ein Hauptproblem: jeder Benutzer würde versuchen, die 20K von 0100 hex an zu benutzen und die oberen 28K würden nicht benutzt.

Einige Programme, die Sie von anderen Verkäufern als Digital Research

kaufen, erhalten Sie nur als „.COM"-Dateien. Sie müssen von 0100 hex an geladen und ausgeführt werden. Können zwei Benutzer dies Programm gleichzeitig ausführen?

Nein. Aber glücklicherweise gibt es eine Lösung. Die meisten MP/M-Computer benutzen ein Charakteristikum genannt **bank selectable memory** (= bank-selektiver Speicher). Die Mikroprozessoren 8080/8085/Z80 können direkt nur 64K RAM adressieren. Das Schlüsselwort ist **direkt**. Stellen Sie sich vor, Sie können eine Kernspeichersektion anweisen, alle Prozesse zu ignorieren, bis sie von der CPU ein speziell codiertes Signal erhält. Eine andere Sektion des Kernspeichers bearbeitet alle Prozesse, bis es dieses Signal erhält und dann nicht weiter. So etwa funktioniert ein bank-selektives Speichersystem.

Tabelle 6–1. Eine MP/M-Kernspeicher-Karte

Bank 1 64K Bytes RAM	Kernspeicher-Adresse (hexadezimal)	Bank 2 64K Bytes RAM
	0000	
Reserviert für MP/M		Reserviert für MP/M
	0100	
TPA Programm des Benutzers 1		TPA Programm des Benutzers 2
	C000	
Nicht gebraucht		Programm des Benutzers 3 (wahlfrei)
	E000	
MP/M		
	FFFF	

Bei MP/M hat gewöhnlich jeder Benutzer eine Kernspeicher-Bank. Wenn Sie in Ihrem System zwei **banks** von je 64K RAM haben, bekommt der erste Benutzer 48K von der ersten Bank, der zweite 48K von der zweiten Bank, und 16K gemeinsamen Kernspeicher-Bereiches werden in der ersten Bank für MP/M reserviert. In Tabelle 6-1 wird ein Beispiel für eine **memory map** (= Kernspeicher-Karte) gezeigt. Beachten Sie die **Adressen** in der Mittelspalte. Beide Benutzer (1 und 2) verwenden die gleichen Adressen für ihre Programme; sie kommen sich nicht in die Quere, weil MP/M nur von der derzeit aktiven memory bank Gebrauch macht. Es ist nicht Ihre Aufgabe, dem System mitzuteilen, wann es die Banken wechseln soll. MP/M tut dies automatisch.

Wenn die Anfangsadresse in Ihrem Kernspeicher anders als 0100 hex ist, müssen Sie „.PRL" statt „.COM" als Datei-Typ benutzen, um Programme zu speichern. Digital Research stellt seine Programme als „.PRL"-Datei zur

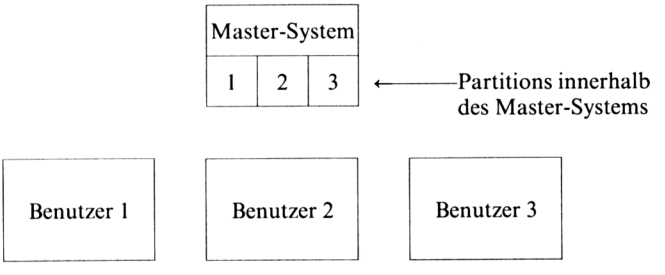
Verfügung. Zusätzlich kann der MAC-Assembler von Digital Research „.PRL"-Dateien aus Assembler-Programmen erstellen. Das geht auch mit „.ASM", aber unter größeren Schwierigkeiten. Digital Research bietet jetzt auch RMAC (einen verschieblichen Assembler), LINK-80 und eine Bibliothek allgemeiner Routinen an, Sie können also „.PRL"-Dateien erstellen.

CP/NET

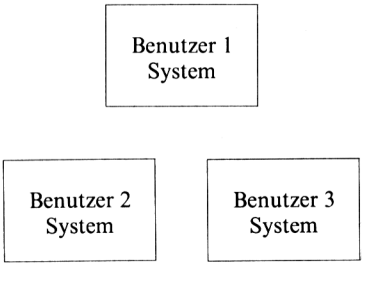
CP/NET ist die Hinzufügung der Netzwerkverarbeitung von Digital Research zur CP/M-Familie von Betriebssystemen. CP/M wurde für einen einzelnen Benutzer auf einem einzelnen Computersysstem, MP/M für mehrere Benutzer auf einem einzelnen Computersystem, CP/NET für mehrere Benutzer auf mehreren Computersystemen entworfen.

Das Konzept von Computernetzwerken unterscheidet sich etwas von Multitasking. Multitasking tendiert dazu, hierarchisch zu sein.

Jeder Benutzer ist ein separates Wesen; Multitasking erlaubt wenig oder keine Interaktion zwischen Benutzern.



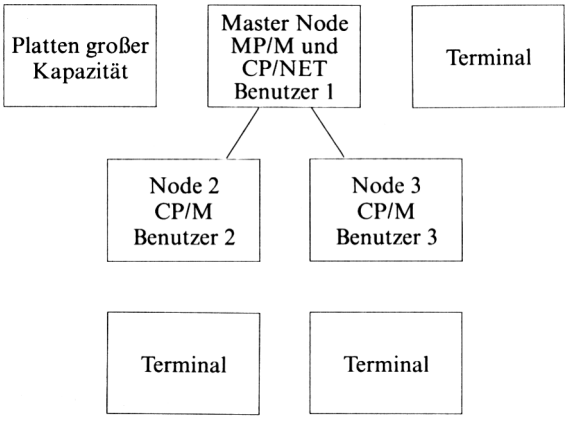
CP/NET operiert nach einem unterschiedlichen Konzept.



Jeder der drei Benutzer repräsentiert ein **System**, nicht einfach einen Anwender. Das logische **linking** (= Verknüpfung) zwischen den Benutzern ist ziemlich direkt, es gibt kein **Master**-System (hierarchisches Prinzip).

Tatsächlich differiert CP/NET etwas vom illustrierten Konzept. CP/NET erfordert zumindest einen **node** (= Knotenpunkt), der als Master (Haupt-

steuerung) fungiert. Ein Benutzer verwaltet das Netzwerk. Der **master node** (= Hauptsteuerungs-Knotenpunkt) muß MP/M und Platten-Laufwerke haben. Andere Knotenpunkte können nur aus einer 8080/8085/Z80-CPU und einem Minimum von 16K Kernspeicher bestehen. Wenn auch diese „nodes" keine anderen Komponenten brauchen, so kann doch ein Terminal, mehr Kernspeicher und Plattenlaufwerke die Funktionstüchtigkeit eines Node-Computers erhöhen. Unter dieser Voraussetzung für ein CP/NET-System wollen wir unser Diagramm wie folgt modifizieren:



Es gibt zwei hauptsächliche Vorteile für ein Computernetzwerk: gemeinsame Nutzung der Resourcen und höhere Ausführungsgeschwindigkeit.

In erster Linie werden in einem Mikrocomputernetzwerk Plattenlaufwerke hoher Kapazität gemeinsam benutzt. Hartplattenlaufwerke sind teuer. Es kann unvernünftig sein, drei Hartplattenlaufwerke für drei Computersysteme zu kaufen. Anwendungen mit großer Datenbasis (Speicher, Wartung und Gebrauch von Dateien großen Umfangs) können von allen Anwendern erfordern, eine Datei oder Datei-Gruppe gemeinsam zu benutzen.

Ein anderes gemeinsam benutztes Mittel ist gewöhnlich ein Drucker. Drucker mit Schreibmaschinen-Qualität sind sowohl teuer als auch langsam. Da mag an einen Node (Computer) ein Schreibmaschinen-Drucker, an einen anderen ein schnellerer aber weniger flexibler Punktmatrix-Drucker angeschlossen sein. Für gröbere Skizzierungen werden Sie vielleicht den schnelleren Punktmatrix-Drucker benutzen. Aber da jeder Benutzer in diesem Netzwerk Ausgaben an jeden Drucker senden kann, werden Sie vielleicht den Endzustand Ihres Schriftsatzes mit Schreibmaschinen-Qualität drucken wollen.

Jeder Node an einem CP/NET-System kann unabhängig funktionieren; jeder Knotenpunkt kann ein separater Computer sein. Knotenpunkte werden nur deshalb an das Netzwerk angeschlossen, um sich Resourcen teilen zu

können. In der Geschäftswelt kann CP/NET benutzt werden, um verschiedene unterschiedliche Maschinen in einem System zu vereinen.

CP/NET-Knotenpunkte können alle MP/M- oder CP/M-Kommandos und -Programme benutzen; die Ausnahmen sind minimal. Zusätzlich enthält CP/NET die folgenden neuen Kommandos:

LOGIN
Verzeichnet den Benutzer im Master-Node

LOGOFF
Koppelt den Benutzer vom Netzwerk ab

SNDMAIL
Sendet Post (Textnachricht) an einen anderen Benutzer

RCVMAIL
Empfängt Post von einem anderen Benutzer

BROADCST
Sendet Nachricht vom Master-Node an alle Benutzer

MRCVMAIL
Empfängt Nachricht von allen Benutzern an den Hauptknotenpunkt

NETWORK
Befähigt einen Knotenpunkt zum Gebrauch von Netzwerkeinheiten

LOCAL
Klammert lokale Einheiten aus dem Gebrauch für das ganze Netzwerk aus

ENDLIST
Sendet ein CONTROL-Z zur Listeinheit

Seit seiner Einführung hat CP/NET noch keine große Zahl von Anwendern gefunden. Das liegt hauptsächlich daran, daß viele Computerhersteller das BIOS in CP/M durch ein spezielles „networking" (Netzwerk-) BIOS ersetzen, welches CP/M vortäuscht, es habe selbst die Netzwerk-Resourcen. Diesen Weg beschreitet eine Reihe von Mikrocomputer-Netzwerken. Der einzige Nachteil ist, daß der Entwickler eines Netzwerk-Systems einige Extraprogramme vorsehen muß, um Streitigkeiten über Netzwerk-Resourcen zu lösen und die Nachrichten-Funktionen zur Verfügung zu stellen, die CP/NET anbietet.

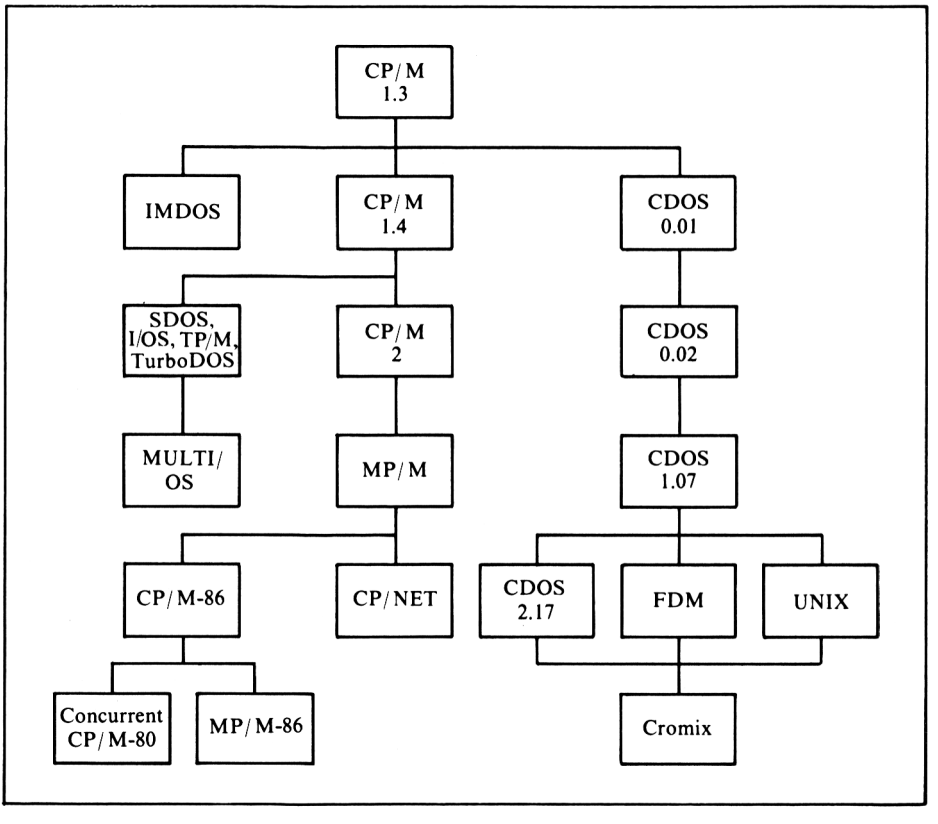
CP/M – Ähnliche Betriebssysteme

Viele Software-Entwickler haben versucht, die Kontrolle von CP/M über den EDV-Prozeß zu verbessern. Die meisten dieser Versuche basierten auf CP/M-80 Version 1.4.

Eine Anzahl anderer Hersteller bietet CP/M-80-kompatible Betriebssysteme an. Einige wie ADDS (Applied Digital Data Systems) haben ihre eigenen Betriebssysteme geschrieben, die wie CP/M-80 funktionieren sollen.

Andere haben einfach CP/M-80 von Digital Research in Lizenz genommen und lediglich einige Möglichkeiten hinzugefügt, die wesentlichen Fähigkeiten ihrer Anlage entsprechen.

Abbildung 6-1 gibt einen Überblick über die Entwicklungsfolge der bekanntesten CP/M-ähnlichen Betriebssysteme. Verschiedene andere Betriebssysteme mögen ebenfalls in diesen Umriß hineinpassen. Subtile Änderungen innerhalb eines Betriebssystems (z. B. CP/M 1.41 gegenüber 1.42) werden in dieser Illustration nicht berücksichtigt; nur die wichtigen Änderungen werden angezeigt.



Cromemco CDOS

Cromemco war einer der ersten Mikrocomputer-Hersteller. Sein erstes Produkt waren nicht Mikrocomputer, sondern Komponenten für Mikrocomputer von IMSAI und Altair. Schnelles Wachstum und sorgfältige Überwachung der Entwicklungs-Reihenfolge neuer Produkte erlaubten Cromemco, ein komplettes System einzuführen.

Cromemco's erstes auf Platte basierendes System war der Computer Z-2. Dieses System befand sich in einem groben Industriegehäuse, und enthielt zwei eingebaute Plattenlaufwerke. Später entwickelte Cromemco das System 3, einen geschäftsorientierten Computer (Environment). Beide Computersysteme wurden mit einem Betriebssystem eingeführt, das von einer Firma namens InfoSoft (danach TSA) aus CP/M-80 entwickelt worden war; das Betriebssystem wurde dann CDOS – Cromemco Disk Operating System – genannt.

Ursprünglich war CDOS wenig mehr als eine Neuauflage von CP/M-80; sein Vorteil beruhte auf den gestiegenen Fähigkeiten des Z80-Mikroprozessors. CDOS wurde jedoch kontinuierlich weiterentwickelt – mit mehr subtilen Verfeinerungen, als in CP/M-80 hervorgebracht wurden. Cromemco war in der Lage, CDOS upzudaten, weil es völlige Kontrolle über die Funktionsweise des Computersystems hat, das es benutzt. CDOS funktioniert nur mit dem **Cromemco disk controller** (Platte-Kontrollsystem) und einer Z80-CPU. Digital Research hat keine Kontrolle über die Mikrocomputer, die CP/M-80 benutzen; und CP/M-80 benötigt daher weniger Änderungen.

CDOS-Kompatibilität mit CP/M-80

Cromemco gibt mit jeder Kopie des CDOS folgende Anmerkung betreffs seiner Kompatibilität mit CP/M-80 heraus:

> **Anmerkung:** Das Cromemco Disk Operating System (CDOS) ist ein Original-Produkt, das von Cromemco, Inc. für ihre eigene Reihe von Mikrocomputern in Z80-Maschinencode entwickelt und geschrieben wurde. Wegen der großen Anzahl von Programmen, die gegenwärtig mit dem CP/M-Betriebssystem laufen, wurde CDOS aufwärts CP/M-kompatibel gestaltet. Cromemco wurde von Digital Research, dem Urheber von CP/M, zum Gebrauch der CP/M-Datenstrukturen und des Benutzer-Interface lizenziert. Das bedeutet, daß die meisten für CP/M (Versionen bis aufwärts zu 1.33) ohne Modifikation unter CDOS laufen werden. Das bedeutet außerdem, daß für CDOS geschriebene Pogramme nicht unbedingt unter CP/M laufen werden.*

CDOS entwickelte sich aus der ersten kommerziell verfügbaren Version von CP/M, der Version 1.3; es enthält keine Möglichkeiten, die Digital Research in späteren Versionen hinzugefügt hat. Dies ist, wenn Sie versuchen, für CP/M-80 Version 1.4 geschriebene Programme unter CDOS laufen zu lassen, nur eine geringere Unbequemlichkeit; die Unterschiede zwischen CP/M 1.3 und 1.4 sind nicht so extrem wie die zwischen 1.4 und 2.2.

Ein Programm, das die Merkmale der Version 2.2 von CP/M-80 benutzt (eigentlich jeder Version nach 2.0), wird auf einem mit CDOS ausgstatteten Computer nicht laufen. Andererseits werden die meisten für CP/M-80 2.2 geschriebenen Programme die hinzugefügten Charakteristika nicht notwendig brauchen. Ein EBASIC-Programm läuft mit 1.4 wie 2.2 und behält seine Kompatibilität mit CDOS. Versionen von Microsoft BASIC und CBASIC2,

* Aus **Cromemco Users Bulletin**, Issue # 1, Dezember 1978

besonders die neuen Compiler für diese Sprachen, werden auf einem mit CDOS ausgestatteten Computer möglicherweise nicht besonders gut laufen.

Ein für CP/M-80 geschriebenes Programm wird schon deshalb nicht korrekt auf Ihrem CDOS-System laufen, weil Sie es laden und ausführen können. Um korrekte Arbeitsweisen zu garantieren, können Sie

- die Programme gründlich testen o d e r
- für Ihren Cromemco-Computer eine Version von CP/M-80 kaufen.

Allgemein wird die zweite Wahl vorgezogen. Wenn Sie sich an Cromemco's Empfehlung bezüglich der Kompatibilität von CDOS und CP/M-80 erinnern: die CDOS-Datenstruktur ist dieselbe, nicht aber einige der Systemfunktionen. Benutzen Sie also besser auf Ihrer Cromemco-Maschine ein CP/M-80-Betriebssystem, um ein CP/M-80-Programm auszuführen. Das ist verläßlicher und ökonomischer als CP/M-80-Programme so zu modifizieren, daß sie zu CDOS passen.

CDOS – Kommandos und Dienstprogramme

Die folgenden CDOS-Kommandos arbeiten wie CP/M-80-Kommandos:

BYE
Kehrt zum Cromemco-System-Monitor zurück

DIR
Zeigt das Dateien-Inhaltsverzeichnis einer Diskette an

ERA
Löscht eine oder mehrere Dateien von einer Diskette

REN
Benennt eine Datei neu

SAVE
Sichert einen Teil des Kernspeicherinhalts in eine Diskettendatei

TYPE
Zeigt eine ASCII-Datei auf Konsole an

Folgende Dienstprogramme (transiente Kommandos) sind bei Cromemco eingeschlossen:

BATCH
Hält eine Liste von Kommandos zur Ausführung bereit

DUMP
Gibt die hexadezimalen Repräsentationen der Inhalte einer Plattendatei aus

EDIT
Ein einfacher zeichenorientierter Editor

INIT
Initialisiert eine Diskette (Formatierung und Vorbereitung)

WRTSYS
Kopiert CDOS von einer Diskette auf eine andere

XFER
Überträgt Dateien von einer Platte oder sonstigen Einheit auf andere

STAT
Zeigt Statistiken über Platten oder sonstige Einheiten an

Außer BYE und INIT werden alle anderen auch von der letzten Version von CP/M-80 anerkannt.

Hier wurde das Minimum der unterstützten Kommandos und Dienstprogramme aufgeführt. Neue Versionen von CDOS (1.07, 2.17 und spätere) schließen andere Dienstprogramme ein und erweitern zustäzlich Kommandos und Dienstprogramme. Frühe Versionen des Programmes STAT zeigen z. B. den freien Plattenspeicher, die Anzahl der Directory-Eintragungen, die Namen von Null-Dateien und eine Fehlermeldung für Diskettenprobleme an. Die letzten Versionen von CDOS schließen STAT-Programme ein, die außerdem spezifische Informationen über den Plattenstatus, den Gebrauch und Einheiten-Zuordnungen ausgeben. (Die Unterschiede zwischen jeder Version von DCOS und CP/M-80 detailliert zu beschreiben und zu erklären, wie alle CDOS-Kommandos arbeiten, würde ein weiteres Buch erfordern.)

Einige Versionen von CDOS schließen verschiedene weitere Dienstprogramme ein:

CDOSGEN
Entwirft ein anders proportioniertes Betriebssystem

DEBUG
Eine Testhilfe ähnlich wie DDT

LINK
Ein Verknüpfungsprogramm, das aus mit Cromemcos Sprachen kompiliertem Programm-Code eine ausführbare Programmdatei macht

SCREEN
Ein kultivierter Editor, der nur mit Cromemco-Terminals arbeitet und für Aufgaben der Systementwicklung oder der Textverarbeitung benutzt werden kann

MEMTEST
Testet den RAM von Gromemco und meldet Fehler

Außer DEBUG sind diese Dienstprogramme für CP/M-80 nicht verfügbar.

CDOS – Verwandte

Cromemco hat zwei Ableitungen von CDOS herausgebracht. Die erste ist kein geschlossenes Betriebssystem, eher eine Erweiterung von CDOS, nämlich ein Gemeinschaftssystem basierend auf dem Multi-User BASIC von Cromemco. Zweitens entwickelte Cromemco ein weiteres Betriebssystem namens **Cromix**, das einigen populären Betriebssystemen von Minicomputern ähnelt.

Das System „Mulit-User BASIC" erlaubt sieben Benutzern, BASIC-Programme unabhängig und simultan laufen zu lassen. Alternativ kann auch ein Benutzer bis zu sieben unabhängige Programme zugleich fahren. Zwischen CDOS-entwickelten und Programmen wie Plattendateien von Multi-User BASIC besteht fast vollkommene Kompatibilität.

Cromix arbeitet in einem völlig anderen Benutzer-Environment (Umgebung = Zustand aller Register, Speicherzellen und anderer Bedingungsanzeigen in einem System). Dieses Umfeld wurde von dem Minicomputer-Betriebssystem UNIX abgeleitet. Ein Cromix-System für einen einzelnen Benutzer erfordert den doppelten Kernspeicher (64K für das Betriebssystem plus 64K für den Anwender) wie CDOS (16K für das Betriebssystem und 48K für den Benutzer). Mit diesem Extra-Speicher ermöglicht Cromix folgendes:

- Dateinamen bis zu 24 Stellen
- Dateien als Directories anderer Dateien. Dies ist unter dem Namen **tree-structured directory** (Kern einer Baum-Struktur) bekannt; sollten Sie ein komplexes Dateien-System skizzieren, würde die Struktur wie ein umgekehrter Baum aussehen:
- Zwölf eingebaute Kommandos
- Über 35 Dienstprogramme
- **Privilegierte** Ebenen, d. h. Schutz von Dateien vor nicht-autorisierten Benutzern
- Multiple Tasks und Benutzer
- Angabe von Datum und Uhrzeit

Cromix ist nicht direkt kompatibel zu CDOS; es handhabt Platteninformation in unterschiedlicher Weise (ein Dateiname von 24 Stellen würde nicht in eine CP/M-80-Directory passen). Cromemco stellt jedoch Dienstprogramme zur Verfügung, die CDOS-Dateien in Cromix-Dateien und umgekehrt konvertieren. Deshalb können auch mit CDOS entwickelte Programme weiterhin benutzt werden. Sehen Sie Cromix und CP/M-80 als zwei völlig unterschiedliche Betriebssysteme ohne Kompatibilität an. Ein CP/M-80-Programm mit Cromix laufen zu lassen, erfordert einen Konvertierungs-Prozeß von vielen Schritten ohne Erfolgsgarantie. Cromix-Dateien und -Programme auf CP/M-80-Kompatibilität zu rekonvertieren ist theoretisch möglich, aber so schwierig, daß ein Nichtprogrammierer es besser nicht versuchen sollte.

Für einen Cromemco-Benutzer sind Cromix und Mulit-User BASIC der Betrachtung wert, weil beide Systeme spezifische Fähigkeiten haben, die die Möglichkeiten von CDOS wie auch CP/M-80 überschreiten. Allerdings bieten beide keine Verbesserungen, die ohne weiteres auf ein CP/M-80-Environment übertragen werden können.

CP/M-80 – Work-alikes (= ähnlich arbeitende Systeme)

Neuerdings existieren einige CP/M-80 – work-alikes. Generell hat jedes von ihnen Vor- und Nachteile gegenüber dem CP/M-80-Betriebssystem. Die populärsten work-alikes sind:

I/OS.

Früher als TSA/OS bekannt; I/OS wurde vom selben Team wie CDOS geschrieben. Der Hauptvorteil ist, daß I/OS die Kompatibilität zwischen CDOS und CP/M-80 aufrechterhält. Kürzlich wurde eine Gemeinschaftsbenutzer-Version von I/OS namens MULTI/OS entwickelt; Benutzer von MP/M-80 dürften daran interessiert sein.

TP/M.

TP/M ist im Wesentlichen eine Z80-Version von CP/M-80; es wird auf 8080- oder 8085-Computern nicht laufen. Der gewöhnlich im Zusammenhang mit TP/M zitierte Vorteil gegenüber CP/M-80 ist, daß der Gebrauch von Z80-Instruktionen die Arbeitsgeschwindigkeit des Betriebssystems bemerkenswert erhöht. Solche Ansprüche sind natürlich vom Equipment abhängig, und es ist die Frage, ob Sie den Geschwindigkeitsunterschied überhaupt bemerken.

SDOS.

Hier wurden von SD Systems einige kleinere und mehr kosmetische Änderungen an CP/M-80 vorgenommen, um SDOS zu entwerfen. Die derzeitige Version von SDOS ist jedoch eine Ableitung der Version 1.4 von CP/M-80 und enthält deshalb nicht die Verbesserungen, die Digital Research mit der Einführung von Version 2.2 vorgenommen hat.

TurboDOS.

Als relativer Newcomer führt TurboDOS eine Anzahl kunstreicher Software-Konzepte für das CP/M-80-Environment ein, woraus bei plattenorientierten Tasks ein wesentlich schneller arbeitendes Betriebssystem resultiert. TurboDOS versucht tatsächlich vorauszusehen, welche Sektionen der Diskette der Benutzer anfordern wird und sicherzustellen, daß sie bei Benutzeraufruf verfügbar sind. Außerdem sind einige der ärgerlicheren Aspekte von CP/M-80 – vor allem, daß Sie keine Disketten austauschen können, ohne es dem System mitzuteilen – ausgemerzt. Jetzt gibt es auch eine Gemeinschaftsbenutzer-Version von TurboDOS.

CP/M-80 für Zenith/Heath, Polymorphic und Radio Shack Computer.

Eigner von Computern wie dem frühen Zenith/Heath, einem Polymorphic oder einem Modell I oder III von Radio Shack müssen vorsichtig sein, wenn

sie für ihre Systeme ein CP/M-80 kaufen. Während CP/M-80 wie in diesem Buch beschrieben arbeiten wird, sind doch die Kernspeicheradressen geändert, um Design-Differenzen bei diesen Computern zu entsprechen. Um die Verschiebbarkeit von CP/M-80 und seinen Bezugspunkten im Kernspeicher aufrechtzuerhalten, müssen normalerweise austauschbare CP/M-80-Programme modifiziert werden, um auf einer anderen Maschine zu laufen. Vergewissern Sie sich, daß eine Programmversion zu Ihrer Maschine paßt, wenn Sie CP/M-80 benutzen und einen der folgenden Computer haben:

- Poly 88
- Polymorphic 8813
- Radio Shack TRS-80 Model I
- Radio Shack TRS-80 Model III
- Heathkit H8
- Heathkit H89
- Zenith Z89

In jedem dieser Fälle ist es möglich, das Equipment so zu modifizieren, daß es mit einem Standard-CP/M-80-Betriebssystem gefahren werden kann, aber solche Änderungen sollten von einem kompetenten Computer-Techniker vorgenommen werden.

Alternativen zu CP/M-86

Mit der Einführung des IBM Personal Computers im Jahre 1981 entstand aus dem Wettstreit der CP/M – Work-alikes das CP/M-86 mit all seinen Erweiterungen.

Das anfängliche Betriebssystem für den IBM Personal Computer wurde alternativ IBMDOS oder Microsoft DOS genannt. Tatsächlich gleicht es einem Produkt von Seattle Computer Products, das führer als das 86-DOS eingeführt wurde.

86-DOS wurde geschrieben, bevor CP/M-86 verfügbar war, und wahrscheinlich deshalb verkörpert es eine interessante Mischung der Charakteristika von CP/M-80 und CP/M-86. Die eingebauten Kommandos für 86-DOS werden anschließend aufgeführt:

86-DOS	**CP/M-86**
DIR	DIR
RENAME	REN
ERASE	ERA
COPY	(wie PIP)
TYPE	TYPE
CLEAR	(wie FORMAT)

Zusätzlich wurde das ursprüngliche 86-DOS-Betriebssystem mit den folgenden transienten Kommandos ausgestattet:

RDCPM
Kopiert Dateien von CP/M-80 auf das Format von 86-DOS um

MKRDCPM
Erstellt RDCPM für Benutzer-Versionen von CP/M-80

HEX2BIN
Wie LOAD.COM von CP/M-80

CHKDSK
Überprüft eine Diskette auf unbrauchbare Sektionen

SYS
Wie SYSGEN.COM von CP/M-80

EDLIN
Wie ED.COM von CP/M-80

ASM
Wie ASM-86 von CP/M-86

TRANS
Übersetzer von Z80- auf 8086-Quellcode

DEBUG
Wie DDT-86 von CP/M-86

Zusätzlich betrachtet der Kommando-Interpreter von 86-DOS (dem Äquivalent zum CCP) jede Datei des Typs „.BAT" als Unterstüzung für Stapelverarbeitung und führt die Anweisungen des Kommandos in der gelesenen Reihenfolge aus.

Microsoft kaufte die Rechte und paßte 86-DOS an den IBM Personal Computer an. Die mit diesem Betriebssystem ausgelieferte Dokumentation ist ausgezeichnet, und der Benutzer sollte im Zusammenhang mit diesem Buch in der Lage sein, mit 86-DOS, IBMDOS, Microsoft DOS oder wie sonst man dieses Betriebssystem nennen will, zu arbeiten. Die Wurzeln von CP/M-80 treten klar in der Auswahl des Formates und der Konventionen von Dateinamen hervor, desgleichen im Gebrauch von Kontrollzeichen, um die Kommandozeile auszugeben, der Anforderung auf dem Bildschirm und vielen anderen „sichtbaren" Aspekten des 86-DOS.

Hauptsächlich differiert 86-DOS in der inneren Reichweite des Betriebssystems. Die BDOS-Aufrufe sind unterschiedlich zu denen des CP/M-80 bzw. CP/M-86, obwohl sie offensichtlich auf Ähnlichkeit ausgelegt wurden. Von besonderer Wichtigkeit ist, daß das Layout der Disketten-Information sich zwischen 86-DOS und beiden Versionen des CP/M unterscheidet. Die kritische Differenz ist einfach: es handelt sich um den Aufbau der Directory und der Systemspuren. Hier werden beide Arten aufgezeigt, wie 86-DOS und CP/M beim gleichen Plattenformat den Speicherplatz verwenden:

	86-DOS	**CP/M-86**
Spur 0	DIRECTORY	SYSTEM
Spur 1	SYSTEM	SYSTEM
Spur 2	SYSTEM	DIRECTORY
Spuren 3–77	DATEN	DATEN

Die allerersten Versionen von 86-DOS benutzten ein 16-Byte-Eintragungsformat in die Directory, das jetzt allerdings auf 32 Bytes erweitert wurde, entsprechend CP/M. Jedenfalls erfordert 86-DOS nicht, daß der Benutzer das Betriebssystem benachrichtigt, wenn eine Diskette gewechselt werden soll.

Kurz, 86-DOS ist eine Alternative zu CP/M-86. Microsoft und Digital Research, einst enge Partner in der Software-Entwicklung, haben sich inzwischen zu Rivalen in bezug darauf entwickelt, welches der beiden Betriebssysteme denn nun „Standard" für auf 8086 basierende Systeme sei. Nun sieht es aber so aus, als ob keiner von ihnen gewinnen kann, denn Microsoft hat bereits sein Vorhaben bekundet,ein Superset von 86-DOS zu entwickeln, das mehr XENIX, ihrem von UNIX abgeleiteten Betriebssystem entspricht, Digital Research dagegen arbeitet an einem sogenannten „concurrent CP/M-86". Nur CP/M-86 ist direkt kompatibel zum Format der CP/M-80-Disketten. Dies kann allerdings der entscheidende Faktor für Benutzer sein, die von Z80- auf 8086-CPU umstellen, denn eine Diskette muß entweder konvertiert werden, um mit 86-DOS brauchbar zu sein, oder es muß erst ein besonderes Programm geladen werden, das das Diskettenformat übersetzt.

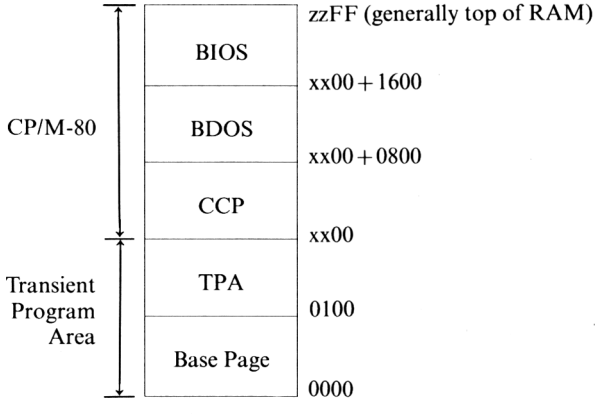
Kapitel 7
Technische Aspekte von CP/M

Im vorigen Kapitel haben wir die zur Anwendung nötige Information über CP/M repräsentiert. Dieses Kapitel beschreibt, was ein Assembler-Programmierer wissen sollte, um mit dem CP/M-Environment effektiv umgehen zu können. Wenn Sie daran interessiert sind, etwas über die interne Struktur von CP/M zu erfahren, dann lesen Sie dieses Kapitel. Viele Leser werden das hier präsentierte Material als zu technisch für ihre Zwecke empfinden.

Der Aufbau von CP/M

Wenn Sie in Ihrem Computer CP/M **kaltstarten**, wird CP/M-80 normalerweise in den obersten freien Kernspeicherbereich geladen. CP/M-80 hat einen Kernspeicherbedarf von ungefähr 6K, aber der maschinenabhängige Teil wird noch einmal zwischen 1K und 8K Kernspeicher erfordern. So sind in einem normalen CP/M-80-System die obersten 7K bis 14K Kernspeicher von den Instruktionen belegt, die wir unter dem Namen „CP/M-80" zusammenfassen (CP/M-86 wird in diesem Kapitel später diskutiert).

Außerdem belegt CP/M-80 die ersten 256 Bytes (genannt „base page") Ihres Kernspeichers. Ihre Programme und Daten dürfen den Bereich zwischen 0100 hex und dem Anfang des CP/M-80 belegen. Ein CP/M-80-System von 32K stellt diese also nicht insgesamt für Ihre Programme und Daten zur Verfügung, sondern überläßt Ihnen nur ungefähr 24K.



Anmerkung: Alle Adressen sind hexadezimal aufgeführt. „xx" und „zz" stehen für den Adreßanteil, der vom Kernspeicherumfang abhängig ist.

Wenn CP/M-80 zuerst gestartet wird, wird die Basis-Seite (base page) zunächst mit einiger Spezialinformation aufgefüllt. Diese Information wird in diesem Kapitel noch im Detail beschrieben, zunächst soll jedoch ein kurzer Überblick geboten werden:

Kernspeicher-Adresse	Funktion
0000–0002	Sprung zur BIOS-Warmstart-Routine
0003	IOBYTE
0004	Aktive Laufwerksnummer, aktive Benutzernummer
0005–0007	Sprung zum BDOS-Eingangsvektor
0008–0037	Reserviert für Maschinenunterbrechungen
0038–003A	RST7, von DDT benutzt
003B–003F	Reserviert für Maschinenunterbrechungen
0040–004F	Reserviert für „scratch" (= Ausstreichen) durch BIOS
0050	Laufwerkskommando wurde von CP/M 3 oder MP/M 2 geladen
0051	Adresse des Paßwortes für den ersten Standard-FCB (CP/M 3, MP/M 2)
0053	Länge des Paßwortes für den ersten FCB (CP/M 3, MP/M 2)
0054	Adresse des Paßwortes für den zweiten FCB (CP/M 3, MP/M 2)
0056	Länge des Paßwortes für den zweiten FCB (CP/M 3, MP/M 2)
0057–005B	Reserviert
005C–006B	Erster Standard-FCB (file control block = Dateienkontrollblock) (CP/M 3, MP/M 2)
006C–007F	Zweiter Standard-FCB (CP/M 3, MP/M 2)
0080–00FF	Standard-Plattenpuffer/Kommandopuffer

Anmerkung: CP/M 1.4, CP/M 2.2. und MP/M 1 benutzen alle nur einen FCB, der an 005C hex beginnt, wobei der Bereich von 007D–007F hex für die Position eines Satzes im „direkten Zugriff" (random) im CP/M 2.2 und MP/M 1 reserviert ist. Zur Zeit ist Digital Research dabei, die base page leicht zu verändern, um spezielle Funktionen hinzuzufügen wie Dateischutz und Vorkehrungen für Gemeinschaftsbenutzer.

Da Sie jetzt das grundsätzliche Struktur-Schema von CP/M-80 kennen, lassen Sie uns einen Blick auf seine Einzelteile werfen.

CCP – der Console Command Processor

Das CCP-Modul interpretiert die eingetippten CP/M-80-Kommandos. Normalerweise ist nur dieser Teil des CP/M relevant, wenn auf der Konsole ein „A" (oder die Anforderung für ein anderes Laufwerk) erscheint. Der CCP erkennt nur wenige Kommandos und Kontrollzeichen (s. Kap. 2).

Das nach „A" eingegebene Kommando wird zuerst auf Stelle 0080–00FF hex gespeichert. Wenn CCP das Kommando nicht erkennt, sucht es auf der Platten-Directory nach einer „.COM"-Datei, deren Namen in den ersten acht Stellen (oder bis zum ersten Leerzeichen in der Kommandozeile) übereinstimmt. Die „matched" Datei wird ab 0100 hex in den Kernspeicher geladen, und dort beginnt die Ausführung nach Vollendung des Ladevorgangs. Digital Research nennt so eine Datei ein „transientes Kommando".

Der Kommando-Puffer reicht von 0080 hex bis 0100 hex, es werden also von CP/M-80 128 Zeichen als maximale Kommandolänge anerkannt.

Da der Standard-Plattenpuffer nicht benutzt wird, um das Programm zu laden, beginnt das von CCP ausgeführte Kommando immer noch bei 0080 hex und kann für nachfolgende Verarbeitung vom Programm verwendet werden. So lange werden also Kommandozeilen wie „MBASIC FILENAME /F:3" verarbeitet. Hier würde also CCP zuerst die Inhalte von MBASIC.COM in den Kernspeicher laden und dann die Ausführung an 0100 hex weitergeben. Das Programm MBASIC würde dann die auf 0080 beginnende Kommandozeile miteinbeziehen, würde feststellen, daß zusätzliche Information eingegeben wurde, und mit dem Versuch fortfahren, Ihr Original-Kommando auszuführen.

Für fortgeschrittene Programmierer gibt es noch einen zusätzlichen Trick, das Layout des CCP einzubeziehen: ein Programm kann automatisch gestartet werden. Dazu müssen Sie einige spezielle Adressen innerhalb des CCP kennen:

CCP start + 07 hex Länge des Kommandos
CCP start + 08 hex Erstes Byte des Kommandos
.
.
.
CCP start + 87 hex Letztes Byte, das für Kommandos benutzt werden kann
CCP start + 88 hex Niederwertiges Byte des Pointers
CCP start + 89 hex Höherwertiges Byte des Pointers

Plazieren Sie Ihr Kommando einfach acht Stellen hinter dem CCP; Ihr Kommando kann bis zu 80 hex (128 dec) Stellen lang sein. Die Kommando-Zeichenkette muß für den automatischen Start mit einem Nullzeichen (00 hex) enden.

Die beiden „Pointer"-Bytes (Vektor) müssen die Adresse des ersten Bytes des Kommandos enthalten (das niederwertige Byte zuerst, danach das höherwertige, wie es in der 8080-Programmierung üblich ist). Der CCP benutzt diesen Pointer, um festzuhalten, wieviel vom Kommando er bereits verarbeitet hat. Wenn Sie ihn nicht vorher auf den Anfang setzen, wird CCP möglicherweise annehmen, daß es bereits einen Teil des Kommandos ausgeführt hat.

Es folgt ein kurzes Assembler-Programm mit dem Autostart-Konzept. Sie können dieses Programm an ein anderes Programm in Assembler-Sprache anhängen, um automatisch die Steuerung von einem Programm auf das andere zu übertragen.

```
; ---------------------------------------------------------------------------------------------------
; AUTOSTRT.ASM                          version  1.Ø  1Ø/12/81                                    ;
; Copyright 1981 by Thom Hogan                                                                    ;
;                                                                                                 ;
; May be used for non-commercial purposes without                                                 ;
; permission.                                                                                     ;
;                                                                                                 ;
; ---------------------------------------------------------------------------------------------------
;
BDOS        EQU     ØØØ5H       ;cp/m BDOS entry point
;
            ORG     Ø1ØØH
;
START:      LHLD    ØØØ1H       ;get address of BIOS + 3
            MVI     L,ØØH       ;in the zero low-order byte of HG
            MOV     A,H         ;get it where we can use it
            SUI     16H         ;subtract offset to CCP
            MOV     H,A         ;put it back in HL
            SHLD    CCP         ;save CCP location for later use
;
; DO ANY EXTRA PROCESSING YOU WISH IN HERE
;
MIDDLE:     LXI     D,FILENAME  ;point to command
            LXI     B.1Ø        ;set to length of command + 1
            LHLD    CCP         ;get CCP location
            MVI     L,Ø7        ;location of length is after CCP
            CALL    MOVE        ;move command string
            LHLD    CCP         ;get back CCP location
            MVI     L,88H       ;location of lsb pointer
            MVI     A,Ø8H       ;get default lsb start
            MOV     M,A         ;put it in place
            INX     H           ;location of msb pointer
            MOV     A,H         ;get default msb start
            MOV     M,A.        ;put it in place
            LHLD    CCP         ;get back CCP location
            PCHL                ;and go to it!
;
MOVE:       LDAX    D           ;get byte to move
            MOV     M,A         ;move it
            INX     H           ;increment destination
            INX     D           ;increment source
            DCX     B           ;decrement count
            MOV     A,B         ;get counter into A
            ORA     C           ;check if done
            JNZ     MOVE        ;. . . not done
            RET                 ;. . . done
```

```
;
CCP:        DS      2
FILENAME:   DB  ØØ,'XXXXXXXXX',ØØ
;
```

Kommandolänge Kommando abschließende Nullen

```
            END
```

Es gibt elegantere Methoden, diese Aktion zu programmieren (vor allem, wenn Ihr Computer auf Z80 basiert), aber der hier gezeigte Prozeß enthält keine Programmiertricks. Sie könnten das obige Beispiel sogar in BASIC recodieren. Es wäre langsam und mühselig, aber korrekt programmiert würde es dennoch die gleiche Funktion ausfüllen: ein gültiges CP/M-80-Kommando am Ende eines Programmes auszuführen.

BDOS – das Basic Disk Operating System

Jede Aktivität des Platte-Betriebssystems sowie die meisten der Konsole durchlaufen diese Sektion des CP/M-80. Auf BDOS kann keinesfalls durch direkte Konsolenkommandos zugegriffen werden. Stattdessen überstellt CCP oder ein transientes Programm die Nummer der gewünschten Funktion in das interne Register C des Mikroprozessors und führt dann eine CALL-Instruktion an die Adresse 0005 hex aus.

Einige BDOS-Funktionen erfordern außerdem Extra-Information in anderen internen Registern. Sollten Sie z. B. ein bestimmtes Zeichen auf der Konsole anzeigen wollen, würden Sie es im internen Register F angeben.

Andere BDOS-Funktionen geben Information an das aufrufende Programm (oder einen Teil des CP/M-80) zurück. Wenn sie mit BDOS ein Zeichen von der Konsole empfangen wollen, werden Sie es in Register A erhalten.

Wir werden kurz jede der BDOS-Funktionen klarlegen. Dazu ist es notwendig, die angewandte Notationsweise kurz zu erläutern.

Die 8080-CPU, für die CP/M ursprünglich geschrieben wurde, sieht sieben interne 8-Bit-Register für generelle Zwecke und acht Status-Bits vor. Sechs dieser Register werden oft als drei 16-Bit-Register gepaart. Die 8080-Register und ihr normaler Assembler-Gebrauch im CP/M sind folgende:

Register	**Gebrauch**
A (Akkumulator)	8-Bit-Zeichen, I/O- oder manipuliertes Datenregister
B & C	Einzeln 8-Bit-Register, gepaart 16-Bit-Adress-Register
D & E	Ebenso
H & L	Einzeln 8-Bit-Register, gepaart 16-Bit-Adress-Register im Kernspeicher
Status	Acht Flag-Bits für Übertrag, Gleichheit, Null usw.

Digital Research benutzt die Register für alle BDOS-Funktionen des CP/M-80 in durchdachter Weise, und zwar folgendermaßen:

A (Akkumulator)	Ein 8-Bit-Wert für oder von BDOS
B	Von BDOS nicht beansprucht
C	Aufrufs-Nummer der BDOS-Funktion
DE	16-Bit-Adreßvariable
HL	16-Bit-Wert von oder für BDOS und L; kopiert den Register-Inhalt von A

Um die Funktionsdefinitionen aus Tabelle 7-1 zu verstehen, sollten Sie auch über den FCB und ddas IOBYTE Bescheid wissen. Detaillierte Beschreibungen dieser beiden Konzeptionen erscheinen später in diesem Kapitel.

Um eine BDOS-Funktion anzuwenden, lädt ein Programm die Funktionsnummer nach Register C, andere Register wie erforderlich, und ruft dann BDOS über die Kernspeicheradresse 0005 hex auf. BDOS führt die Funktion aus und gibt die Kontrolle an das aufrufende Programm zurück.

Wir wollen dies einmal darstellen. Wie Tabelle 7-1 zeigt, ist die Funktionsnummer zum Ausdruck von Zeichenketten 09 hex. Das Registerpaar DE muß die Adresse der Zeichenkette enthalten, und diese muß mit einem „$" abschließen. Sie wird wie folgt angezeigt:

1. Ihr Programm muß die Zeichenkette irgendwo im Kernspeicher zur Verfügung stellen, durch ein „$" abgeschlossen.
2. Überstellen Sie 09 hex nach Register C.
3. Danach stellt Ihr Programm die Anfangsadresse der Zeichenkette in Registerpaar DE zur Verfügung.
4. Ihr Programm ruft dann BDOS über den Eingangsvektor auf 0005 hex auf (CALL).
5. BDOS führt dann die Funktion „print string" (= drucke Zeichenkette) aus und übergibt die Steuerung an Ihr Programm.

Hier ein Beispiel:

```
; ---------------------------------------------------------------
; PRINT A STRING EXAMPLE                                        ;
; Uses BDOS entry vector at ØØØ5 hex                            ;
; ---------------------------------------------------------------
;
BDOS        EQU     ØØØ5H       ;the BDOS entry vector
STRING      EQU     Ø9H         ;the PRINT STRING function #
;
            ORG     Ø1ØØH       ;start at the first TPA address
DOIT:       MVI     C,STRING    ;load the function number
```

Tabelle 7–1. BDOS-Funktionsdefinitionen für CP/M-80 Version 2.2

Funktion		Eingangs-Parameter	Ausgangs-Parameter	Erklärung
Nr.	Name			
00	SYTEM RESET	Keine	Keine	Startet CP/M-80 neu, indem sie nach Reinitialisierung des Platten-Subsystems die Kontrolle an den CCP zurückgibt.
01	CONSOLE INPUT	Keine	A = ASCII-Zeichen	Übergibt das nächste getippte Zeichen an das anfordernde Programm. Jedes nicht-druckbare Zeichen wird an den Bildschirm geechot (wie BACK-SPACE, TAB oder CARRIAGE RE-TURN). Die Kontrolle der Ausführung geht nicht an das aufrufende Programm zurück, solange nicht ein Zeichen getippt wurde. Standard-CCP-Kontrollzeichen werden erkannt und die entsprechenden Aktionen durchgeführt (CONTROL-P schaltet das Druckerecho ein oder aus).
02	CONSOLE OUTPUT	E = ASCII-Zeichen	Keine	Gibt das Zeichen in Register E auf Konsole aus. Standard-CCP-Kontrollzeichen werden erkannt und die entsprechenden Aktionen durchgeführt (CONTROL-P schaltet das Druckerecho ein oder aus usw.).
03	READER INPUT	Keine	A = ASCII-Zeichen	Übergibt das nächste Zeichen von der Leseeinheit an das aufrufende Programm. Die Kontrolle der Ausführung geht nicht an das aufrufende Programm zurück, solange nicht ein Zeichen empfangen wurde.
04	PUNCH OUTPUT	E = ASCII-Zeichen	Keine	Übermittelt das Zeichen in Register E an die Stanzeinheit.
05	LIST OUTPUT	E = ASCII-Zeichen	Keine	Übermittelt das Zeichen in Register E an die Listeinheit.
06	DIRECT CONSOLE IN	E = FF hex	A = ASCII	Wenn Register E FF hex enthält, wird die Konsoleneinheit befragt, ob ein Zeichen abrufbereit ist. Sonst wird dem aufrufenden Programm in Register A ein 00 übergeben, anderenfalls das erkannte Zeichen. Wenn Register E ein anderes Zeichen als FF hex enthält, wird dieses Zeichen an die Konsolanzeige übermittelt. Alle CCP-Kontrollzeichen werden ignoriert. Der Anwender muß das Programm gegen unsinnige Zeichen schützen, die von der Konsoleinheit ausgehen oder empfangen werden.
	DIRECT CONSOLE OUT	E = ASCII-Zeichen	Keine	

Anmerkung: CP/M-80 kopiert stets die Inhalte des Registers H in das Register A, sofern nichts Spezifisches in Register A auszugeben ist. Einige Hersteller, besonders Microsoft, machen davon Gebrauch, um die Informationsbewegung zwischen den Registern H und A zu verringern.

Tabelle 7–1. BDOS-Funktionsdefinitionen für CP/M-80 Version 2.2 (Fortsetzung)

Funktion		Eingangs-Parameter	Ausgangs-Parameter	Erklärung
Nr.	Name			
07	GET IOBYTE	Keine	A = IOBYTE	Kopiert das an Adresse 0003 hex gespeicherte Byte nach Register A, bevor es die Kontrolle an das aufrufende Programm zurückgibt.
08	SET IOBYTE	E = IOBYTE	Keine	Kopiert den Wert in Register E auf das Byte unter der Adresse von 0003 hex, bevor es die Kontrolle an das aufrufende Programm zurückgibt.
09	PRINT STRING	DE = Adresse der Zeichenkette	Keine	Sendet die Zeichenkette, deren Adresse im DE-Registerpaar gespeichert ist, an die Konsole. Alle nachfolgenden Zeichen werden gesendet, bis BDOS ein ASCII „$" (24 hex) entdeckt. CCP-Kontrollzeichen werden entdeckt und ihre Funktion ausgeführt.
0A	READ CONSOLE BUFFER	DE = Pufferadresse	Daten im Puffer	Diese Funktion führt im wesentlichen dasselbe aus, wie CCP es würde, indem sie die Zeichen, die der Benutzer eingibt, übernimmt und sie in den Puffer an der im Registerpaar DE gespeicherten Adresse speichert. Das erste durch das DE-Paar angezeigte Byte im Puffer muß die maximale Länge des Kommandos enthalten; BDOS wird die im zweiten Byte gezählte Anzahl von Zeichen durch das getippte Kommando beginnend mit dem dritten Byte der Kette, die das Registerpaar DE anzeigt, plazieren. Alle Standard-CCP-Edit-Zeichen werden während des Kommando-Einganges erkannt.
0B	GET CONSOLE STATUS	Keine	A = Status	BDOS prüft den Status der Konsol-Einheit und übergibt 00 hex, wenn kein Zeichen bereit ist, FF hex, wenn ein Zeichen getippt wurde.
0C	GET VERSION NUMBER	Keine	HL = Version	Wenn das im Register H angegebene Byte 00 hex ist, so ist das gegenwärtige Betriebssystem CP/M, 01 hex bedeutet MP/M. Das in Register L übergebene Byte ist 00 hex, wenn die Version vor CP/M 2.0 liegt, 20 hex, wenn sie 2.0 ist, 21 hex, wenn es sich um 2.1 handelt, 22 hex für 2.2 usw.

Anmerkung: CP/M-80 kopiert stets die Inhalte des Registers H in das Register A, sofern nichts Spezifisches in Register A auszugeben ist. Einige Hersteller, besonders Microsoft, machen davon Gebrauch, um die Informationsbewegung zwischen den Registern H und A zu verringern.

Tabelle 7–1. BDOS-Funktionsdefinitionen für CP/M-80 Version 2.2 (Fortsetzung)

Funktion		Eingangs-Parameter	Ausgangs-Parameter	Erklärung
Nr.	Name			
0D	RESET DISK SYSTEM	Keine		Wird angewandt, um CP/M die Anweisung zu geben, das Platten-Subsystem auf Null zu setzen. Sollte bei jedem Diskettenwechsel angewandt werden.
0E	SELECT DISK	E = Platten-nummer	Keine	Wählt die Plattennummer für nachfolgende Plattenoperationen aus. 00 hex in Register zeigt Laufwerk A:, 01 hex Laufwerk B: an, usw.
0F	OPEN FILE	DE = FCB-Adresse	A = „Gefunden"/ „Nicht gefunden"-Code	Wird benutzt, um eine Datei auf dem z. Z. aktiven Laufwerk unter der gegenwärtigen Benutzer-Nummer zu aktivieren. BDOS tastet die ersten 14 Bytes des angezeigten FCB-Blocks ab und versucht, den betreffenden Datei-Namen im Block zu finden. Ein ASCII „?" (3F hex) kann an jeder der Stellen des Dateinamens benutzt werden, um ein „don't care"-Zeichen anzuzeigen. Wenn ein Dateiname gefunden wird, wird von CP/M-80 die relevante Information über die Datei in den Rest des FCB's eingetragen. Ein Wert von 00 bis 03 hex in Register A zeigt an, ob die Operation erfolgreich war, während FF hex bedeutet, daß die Datei nicht gefunden wurde. Wenn Fragezeichen zur Identifizierung einer Datei benutzt wurden, wird der erste zugehörige Eintrag angenommen.
10	CLOSE FILE	DE = FCB-Adresse	A = „Gefunden"/ „Nicht gefunden"-Code	Übt das Gegenteil der OPEN FILE-Funktion aus. Die CLOSE FILE-Funktion muß jedesmal nach Vollendung des Gebrauchs einer Datei ausgeführt werden, in die Information geschrieben wurde.
11	SEARCH FOR FIRST	DE = FCB-Adresse	A = „Gefunden"/ „Nicht gefunden"-Code	Erfüllt dasselbe wie die OPEN FILE-Funktion mit dem Unterschied, daß der laufende Platten-Puffer mit dem 128-Byte-Satz gefüllt wird, der den Directory-Eintrag der zugehörigen Datei enthält.
12	SEARCH FOR NEXT	Keine	A = „Gefunden"/ „Nicht gefunden"-Code	Bewirkt dasselbe wie die SEARCH FOR FIRST-Funktion, außer daß die Suche vom letzten gefundenen Eintrag an fortgesetzt wird.

Anmerkung: CP/M-80 kopiert stets die Inhalte des Registers H in das Register A, sofern nichts Spezifisches in Register A auszugeben ist. Einige Hersteller, besonders Microsoft, machen davon Gebrauch, um die Informationsbewegung zwischen den Registern H und A zu verringern.

Tabelle 7–1. BDOS-Funktionsdefinitionen für CP/M-80 Version 2.2 (Fortsetzung)

Funktion		Eingangs-Parameter	Ausgangs-Parameter	Erklärung
Nr.	Name			
13	DELETE FILE	DE = FCB-Adresse	A = „Gefunden"/ „Nicht gefunden"-Code	Setzt auf dem Directory-Eingang für die vom FCB angezeigte Datei ein „flag" (Kennzeichen), so daß CP/M-80 sie nicht länger als gültige Datei anerkennt. Tatsächlich wird keine Information bei Ausführung dieser Funktion zerstört, obwohl untergeordnete Disketten-Schreibbefehle einige der Bereiche benutzen mögen, die vorher mit der „deleted" (vernichteten) Datei assoziiert waren.
14	READ SEQUENTIAL	DE = FCB-Adresse	A = Fehler-Code	Wenn eine Datei durch den Gebrauch einer OPEN FILE- oder MAKE FILE-Funktion aktiviert wurde, liest die READ SEQUENTIAL-Funktion den nächsten 128-Byte-Block der laufenden DMA-Adresse ein. In Register A wird der Wert 00 hex gemeldet, wenn das „READ" erfolgreich verlaufen ist, während jedes Nicht-Nullzeichen in Register A einen Fehler anzeigt.
15	WRITE SEQUENTIAL	DE = FCB-Adresse	A = Fehler-Code	Wenn eine Datei für den Gebrauch einer OPEN FILE- oder MAKE FILE-Funktion aktiviert wurde, schreibt die WRITE SEQUENTIAL-Funktion den 128-Byte-Block des Kernspeichers an der laufenden DMA-Adresse in den nächsten 128-Byte-Satz der genannten Datei.
16	MAKE FILE	DE = FCB-Adresse	A = DIR-Code	Erstellt eine neue Datei mit der durch den FCB angezeigten Information (Name). CP/M-80 prüft nicht, ob die Datei bereits existiert, deshalb müssen Sie die Prüfung durchführen, ob diese Datei existiert (oder sie nötigenfalls zerstören). Eine neu erstellte Datei muß nicht unbedingt eröffnet werden, da die MAKE FILE-Funktion auch die notwendigen OPEN-Operationen enthält.
17	RENAME FILE	DE = FCB-Adresse	A = DIR-Code	Ändert den durch die ersten 16 Bytes des FCB angegebenen Datei-Namen in den Namen der zweiten 16 Bytes.

Anmerkung: CP/M-80 kopiert stets die Inhalte des Registers H in das Register A, sofern nichts Spezifisches in Register A auszugeben ist. Einige Hersteller, besonders Microsoft, machen davon Gebrauch, um die Informationsbewegung zwischen den Registern H und A zu verringern.

Tabelle 7–1. BDOS-Funktionsdefinitionen für CP/M-80 Version 2.2 (Fortsetzung)

Funktion		Eingangs-Parameter	Ausgangs-Parameter	Erklärung
Nr.	Name			
18	RETURN LOGIN VECTOR	Keine	HL = eingeschaltete Platten	Die Bits im HL-Registerpaar spezifizieren, welche Laufwerke aktiv sind. Das erste Bit im Register L bezieht sich auf Laufwerk A:, das letzte Bit im Register H korrespondiert mit Laufwerk P:, der höchstmöglichen Laufwerksangabe. Ein Bit 1 zeigt einen aktiven Status an, Bit 0 ein inaktives Laufwerk.
19	RETURN CURRENT DISK	Keine	A = z. Zt. aktive Platte	Die Nummern 0 bis 15 bezeichnen nach Rückkehr aus dieser Funktion das z. Zt. aktive Standardlaufwerk.
1A	SET DMA ADDRESS	DE = DMA	Keine	Wird benutzt, um den Kernspeicherblock (128 Bytes) für die Pufferung aller Plattenübertragungen auszuwählen. Beim RESET von System oder Platte, bei einem Kalt- oder Warmstart, wird der Puffer in einem normalen CP/M-80-System auf 0080 hex gesetzt.
1B	GET ALLOC ADDRESS	Keine	HL = Zuordnungs-Adresse	Übergibt die Startadresse des Zuordnungsvektors, einer Tabelle, die im Kernspeicher für jedes „on line"-Laufwerk (jedes z. Zt. angeschlossene Laufwerk) geführt wird und den Teil der Diskette anzeigt, der gerade im Gebrauch ist.
1C	WRITE PROTECT DISK	Keine	Keine	Bewirkt einen temporären Schreibschutz für die Diskette im derzeitigen Standard-Laufwerk.
1D	GET R/O VECTOR	Keine	HL = Laufwerk-R/O	Übergibt im Registerpaar HL einen 16-Bit-Wert, der anzeigt, welche Laufwerke im System schreibgeschützt sind. Die Laufwerke werden wie im LOGIN VECTOR mit dem Bitwert 1 für Schreibschutz angezeigt.
1E	SET FILE ATTRIBUTES	DE = FCB-Adresse	A = DIR-Code	Setzt die Datei-Attribute, die die System-Directory und den R/O- bzw. R/W-Status der durch die FCB-Adresse angezeigten Datei bezeichnet.
1F	GET DISK PARMS	Keine	HL = DPB-Adresse	Ermittelt den Platten-Parameter-Block für das derzeitig aktive Laufwerk. Diese Parameter können benutzt werden, um verfügbaren Raum auf einer Diskette anzugeben oder die Charakteristika eines Laufwerks unter Benutzerkontrolle zu ändern.

Anmerkung: CP/M-80 kopiert stets die Inhalte des Registers H in das Register A, sofern nichts Spezifisches in Register A auszugeben ist. Einige Hersteller, besonders Microsoft, machen davon Gebrauch, um die Informationsbewegung zwischen den Registern H und A zu verringern.

Tabelle 7–1. BDOS-Funktionsdefinitionen für CP/M-80 Version 2.2 (Fortsetzung)

Funktion		Eingangs-Parameter	Ausgangs-Parameter	Erklärung
Nr.	**Name**			
20	GET USER CODE SET USER CODE	E = FF E = Benutzer-Code	A = Derzeitiger Benutzer oder niemand	Wenn Register E'FF hex' enthält, wird die laufende Benutzernummer in Register A zurückgegeben. Um die „User-Nummer" auf Null zu setzen, wird der entsprechende User-Code Register E übergeben. Während das USER-Kommando „User"-Nummern zwischen 0 und 15 erlaubt, kann diese BDOS-Funktion Nummern zwischen 0 und 31 setzen.
21	READ RAN-DOM	DE = FCB-Adresse	A = Fehler-Code	Liest die Nummer, die im 33., 34. und 35. Byte (eine 24-Bit-Adresse) enthalten ist, in den angezeigten FCB.
22	WRITE RANDOM	DE = FCB-Adresse	A = Fehler-Code	Schreibt die Information von der laufenden DMA-Adresse in den DA-Satz (Direct Access = direkter Zugriff) auf den durch die in der im 33., 34. und 35. Byte des angezeigten FCB hingewiesen wird.
23	COM-PUTE FILE SIZE	DE = FCB-Adresse	RRF-Set	Übergibt das derzeitige Format der DA-Datei in den drei Bytes, die im FCB das Feld für den DA-Satz bilden. Wenn das dritte Byte eine 1 enthält, so enthält die Datei die maximale Anzahl von 65.536 Sätzen, im anderen Fall repräsentiert der 16-Bit-Wert der ersten beiden Bytes den Datei-Umfang.
24	SET RAN-DOM RECORD	DE = FCB-Adresse	RRF-Set	Übergibt den nächsten DA-Satz (fügt seinen Wert in das DA-Satz-Feld des FCB ein) nach dem zuletzt sequentiell gelesenen Satz. Digital Research schlägt diese Funktion für index-sequentielle Dateibehandlung vor.
25	RESET DRIVE	DE = Null-Stellung der Laufwerk-Bits	A = Fehler-Code	Bewirkt die Nullsetzung der spezifizierten Laufwerk-Bits als ursprünglichen nicht-angeschlossenen Status.
26	WRITE RAN-DOM (ZERO)	DE = FCB-Adresse	A = Fehler-Code	Schreibt vor einem anderen Satz einen aus Nullen bestehenden Satz auf Diskette; nützlich, um leere DA-Sätze (Nullen statt Daten) zu identifizieren.

Anmerkung: CP/M-80 kopiert stets die Inhalte des Registers H in das Register A, sofern nichts Spezifisches in Register A auszugeben ist. Einige Hersteller, besonders Microsoft, machen davon Gebrauch, um die Informationsbewegung zwischen den Registern H und A zu verringern.

```
                LXI     D,LABLE         ;load the string pointer in
                                        ;reg. pair DE
                CALL    BDOS            ;do it!
;
GETOUT:         JMP     ØØØØH           ;restart CP/M (not part of
                                        ;the example, but necessary
                                        ;to make this a "runnable"
                                        ;example).
;
LABEL:          DB      'DOCTOR, MY PROBLEM IS $'
;
                END
```

Dieses Beispiel ist ein komplettes CP/M-80-Programm, und es sollte laufen, wenn Sie es, wie in Kapitel 4 erklärt, eintippen und assemblieren.

Die meisten erfahrenen Assembler-Programmierer benutzen die Pseudo-Operation EQU (equate = gleichsetzen), um den Werten für jede Funktion und für die Adresse von BDOS, wie im Beispiel vor Tab. 7-1 gezeigt wird, einen Namen zuzuordnen. Diese Praxis macht ein Programm viel lesbarer. Ein Anfänger könnte dasselbe Assembler-Programm wie folgt codieren:

```
                ORG     1ØØH
                MVI     C,9
                LXI     D,$+1Ø
                CALL    Ø
                DB      'DOCTOR,'
                DB      'MY PROBLEM IS'
                DB      '$'
                END
```

Dies führt die Aufgabe aus, aber umständlich.

Es ist klug, unter Benutzung der BDOS-Funktionen einen Satz von Standard-Routinen für Ihre Assembler-Programmierung aufzubauen, wenn möglich. Ein erfahrener Programmierer könnte das obige Beispiel wie folgt codieren:

```
; ---------------------------------------------------------------
; PROGRAM TO PRINT OUT A STRING:                ;
; ---------------------------------------------------------------
;
;
WMSTRT          EQU     ØØØØH           ;CP/M warm start address
BDOS            EQU     ØØØ5H           ;BDOS entry point
PSTR            EQU       Ø9H           ;print string function number
;
;
```

```
        ORG     Ø1ØØH       ;begin program here
BEGIN:  LXI     D,LABEL     ;load the string address
        CALL    PRINT       ;execute the print subroutine
        JMP     WRMSTRT     ;restart CP/M
;
PRINT:  MVI     C,RSTR      ;load print string function
        JMP     BDOS        ;no need to CALL BDOS here
                            ;RET performed by BDOS
;
LABEL:  DB      'DOCTOR, MY PROBLEM IS $'
;
        END
```

Wenn das Programm Zeichenketten mehrfach drucken soll, wird ein erfahrener Programmierer einige wiederholte Schritte eliminieren. Es wird Ihnen helfen, Ihre eigene Programmierung von Standard-Routinen (den BDOS-Aufrufen) zu trennen. Beachten Sie dabei, daß die PRINT-Routine nach BDOS springt, statt es aufzurufen. Dies spart eine Return-Instruktion (ein Byte) und etwas Ausführungszeit, weil die BDOS-Funktion in jedem Fall ein RET ausführt. Durch Aufruf der PRINT-Routine mit einem JMP nach BDOS wird jedesmal eine Instruktion gespart.

BIOS – das grundlegende Input/Output-System

Ein großer Teil von Verwirrung in bezug auf CP/M-80 rührt vom BIOS her. Mancher kauft CP/M-80 und stellt fest, daß es nicht sauber arbeitet. Das liegt dann an einem unkorrekten BIOS oder einem, das der gegebenen Maschinen-Konfiguration nicht entspricht. Ein CP/M-80-BIOS für eine Konfiguration aufzubauen, wird für Software-Entwickler kein Problem sein, für den Computeranfänger kann es jedoch frustrierend (und unpraktisch) sein. BIOS stellt eine Anzahl von Funktionen niederer Ebene für CP/M-Programme (und BDOS) zur Verfügung. Es wird vom Hersteller, Händler oder Software-Verkäufer, bei dem Sie CP/M-80 gekauft haben, angeboten, nicht aber von Digital-Research.

Meistens wird zusammen mit dem Vertrieb von CP/M-80 mit der Original-Diskette ein BIOS-Beispiel herausgegeben. Digital Research sieht sowohl ein BIOS-Skelett (nur die Knochen und eine Anzahl von Kommentaren; die Routinen enthalten keine spezifischen Instruktionen) als auch ein vollständiges BIOS für das angewandte System vor. Viele Hersteller von Microcumputern lassen das aus und schließen stattdessen ihr eigenes BIOS ein. Manche unterstützen nur die BIOS-Routinen, die u. U. geändert werden müssen (Drucker-, Stanzer- und Leser-Routinen).

Wenn Sie Assembler nicht kennen oder in Bezug auf Microcomputer ein Neuling sind, **versuchen Sie nicht, eine BIOS-Sektion für CP/M-80 zu schreiben!**

Das BIOS führt für CP/M-80 kritische, von der Hardware abhängige Funktionen aus. Es weist CP/M-80 an, wie auf die verschiedenen Einheiten, die Ihr Computer-System ausmachen, zuzugreifen ist (wie z. B. der Schreib-Lese-Kopf eines Platten-Laufwerks zu bewegen ist). Jeder Fehler in der BIOS-Sektion von CP/M-80 kann Ihren Computer veranlassen, falsch oder überhaupt nicht zu arbeiten.

Das BIOS für CP/M-80 Version 2.2 beginnt 1600 hex Bytes hinter dem CCP mit einer Serie von Sprungbefehlen, der sog. „jump table" (Sprung-Tabelle). Diese Instruktionen müssen ordnungsgemäß angeordnet und vollständig sein, damit BIOS sauber arbeitet. Die erste Sektion eines BIOS muß also folgendermaßen aussehen:

```
;  ----------------------------------------------------------------
; BIOS FOR TYPICAL CP/M-8Ø COMPUTER                    ;
;  ----------------------------------------------------------------
;
; The following JMPs must be left in the order in which
; they are presented.
;
; The following ORG statement applies to CP/M-8Ø version 2.2
;
                ORG     CCP+16ØØH  ;start of BIOS
;
                JMP     COLDSTART  ;restart CP/M from scratch
                JMP     WARMSTART  ;restart CP/M
                JMP     CONSTATUS  ;console device status
                JMP     CONINPUT   ;console device input
                JMP     CONOUTPUT  ;console device output
                JMP     LISTOUT    ;list device output
                JMP     PUNCH      :punch device output
                JMP     READER     ;reader device input
                JMP     HOME       ;reset disk head to home
                JMP     SETDISD    ;select disk to use
                JMP     SETTRACK   ;select track to use
                JMP     SETSECTOR  ;select sector to use
                JMP     SETDMA     ;select DMA address to use
                JMP     READDISK   ;read current sector
                JMP     WRITEDISK  ;write current sector
;
; The following two jumps do not apply to CP/M-8Ø version
; 1.4, but are necessary for version 2.2.
;
                JMP     LISTSTATUS ;list device status
                JMP     SECTORTRAN ;sector translation
```

```
; End of standard CP/M-8Ø jump table, but you may extend
; the table to add other functions, if you desire.
;
```

Jeder der „jumps" im obigen Beispiel führt zu einer bestimmten BIOS-Routine. Es folgt ein Beispiel einer BIOS-Routine für die Ausgabe an einen Drucker (IMP LISTOUT):

```
; ---------------------------------------------------------------
; LIST OUTPUT ROUTINE                                           ;
; ---------------------------------------------------------------
;
; This routine provides output to the Vector Graphic
; Bitstreamer II I/O board addressed at port 2.
; All CP/M-8Ø list output will be directed to this
; routine.
;
; Assumptions:
;               - Bitstreamer is initialized by cold start.
;               - Character to be printed is in C.
;               - Bitstreamer is addressed at port 2.
;
LISTST:     EQU     Ø3H         ;status port
LISTDAT:    EQU     Ø2H         ;data port
;
LISTOUT:
            IN      LISTST      ;get current status
            ANI     1           ;check if ready
            JNZ     LISTOUT     ;not ready
;
; Printer is ready if routine gets this far. Print it!
;
LISTON:
            MOV     A,C         ;CP/M-8Ø has char in C
            OUT     LISTDAT     ;send it
            RET                 ;done, return
;
```

Jede Routine innerhalb Ihres BIOS schließt mit einer RET-Instruktion (return = Rücksprung). Unser Beispiel besteht aus nur sechs Instruktionen; die Kommentare erklären die entsprechenden Operationen.

Wie bei BDOS-Routinen, setzt das CP/M-80 auch hier gewisse Annahmen in bezug auf die Eingangs- und Ausgangs-Werte jeder Routine voraus. Eine komplette Auflistung aller benötigten BIOS-Routinen einschließlich Eingangs- und Ausgangs-Parameter wird in Tabelle 7-2 gegeben.

Tabelle 7–2. Definitionen für BIOS-Routinen im CP/M-80

Label in Jump-Table	Eingangs-Parameter	Ausgangs-Parameter	Erklärung
COLDSTART	Keine	C = 0	Ihre Routine sollte alle notwendigen Start-Operationen einschließlich der Initialisierung der Werte in der Basis-Page ausführen. Davor muß das C-Register auf Null gesetzt werden.
WARMSTART	Keine	C = Laufwerk	Ihre Routine soll alle notwendigen Restart-Operationen ausführen, aber ohne Reinitialisierung der Basis-Seite. Das Register C sollte beim Ausgang die Nummer des aktuellen Laufwerks enthalten.
CONSOLE STATUS (CONSTATUS)	Keine	A = Status	
CONSOLE* INPUT	Keine	A = Zeichen	
READER* INPUT	Keine	A = Zeichen	Ihre Routine soll ein Zeichen in der angegebenen Einheit abwarten und es dann in Register A wiedergeben.
CONSOLE* OUTPUT	C = Zeichen	Keine	
LIST* OUTPUT	C = Zeichen	Keine	
PUNCH* OUTPUT	C = Zeichen	Keine	Ihre Routine soll das in Register C enthaltene Zeichen auf der angegebenen Einheit ausgeben.
HOME DISK	Keine	Keine	Der Schreib-Lese-Kopf des Laufwerks soll auf die Home-Position (Spur 0, Sektor 0) gesetzt werden.
SELECT DISK	C = Laufwerk	HL = DHA	Das HL-Register sollte nach Ausführung in Register C die Adresse des Kopf-Etiketts des betreffenden Platten-Parameters enthalten.
SET TRACK	C = Spur	Keine	Der durch Register C angezeigte Wert sollte als nächste Spur für den Zugriff durch das Laufwerk angegeben werden.
SET SECTOR	C = Sektor	Keine	Der durch Register C angezeigte Wert sollte als nächster Sektor für den Zugriff durch das Laufwerk angegeben werden.

* Alle Konsolen- und Einheiten-I/O sollten nach einem Blick auf das IOBYTE (0003 hex) vorgenommen werden.

Tabelle 7–2. Definitionen für BIOS-Routinen im CP/M-80 (Fortsetzung)

Label in Jump-Table	Eingangs-Parameter	Ausgangs-Parameter	Erklärung
SET DMA ADDRESS	BC = DMA-Adresse	Keine	Die vom BC-Registerpaar angezeigte DMA-Adresse sollte für alle Informationsübertragungen vom Kernspeicher zur Diskette und umgekehrt gesetzt werden.
READ DISK	Keine	A = Status	Lies die laufende Spur samt Sektor und übertrage die Daten an die bereits gesetzte DMA-Adresse. Ein während der Übertragung angezeigter Fehler wird durch 01 hex gemeldet.
WRITE DISK	Keine	A = Status	Schreibe die laufende Spur und den Sektor von den Daten an der DMA-Adresse.
SECTOR	BC = Logischer Sektor	HL = Physischer Sektor	
TRANSLATION	DE = Tafel-Adresse des Sektors		Eine Spezial-Routine für Daten in anderen als 128-Byte-Blöcken. Der logische Sektor am Eingang wird umgewandelt und dann als entsprechender aktueller Sektor auf der Diskette wiedergegeben.
LIST STATUS	Keine	A = Status	Ihre Routine sollte die angegebene Einheit daraufhin abfragen, ob sie zur Zeichenübergabe bereit ist oder nicht. Wenn ja, übergibt Register A ein 00 hex, wenn nein, FF hex.

* Alle Konsolen- und Einheiten-I/O sollten nach einem Blick auf das IOBYTE (0003 hex) vorgenommen werden.

Wie BIOS modifiziert wird

Um ein neues modifiziertes BIOS zu erstellen, führen Sie folgende Schritte aus:

1. Auf eine frisch initialisierte und formatierte Diskette kopieren Sie das Beispiel.BIOS, ein Editor-Programm, ASM.COM, DDT.COM, SYSGEN.COM und MOVCPM.COM. Kopieren Sie mit SYSGEN.COM ein nicht-modifiziertes CP/M-80-System auf die Diskette. Danach benutzen Sie für Ihr neues BIOS eine neuformatierte Diskette, so daß Sie nicht aus Versehen ein arbeitendes System zerstören.
2. Wenn Sie können, drucken Sie eine Kopie des Beispiel-BIOS aus (vielleicht müssen Sie dazu das BIOS ändern, um Printer-Routinen einzuschließen). Studieren Sie die Kopie sorgfältig. Werden Sie mit ihrer Struktur, den

individuell angeforderten Routinen und den in den Beispielen enthaltenen Kommentaren vertraut.

3. Geben Sie das Beispiel-BIOS mit ED.COM oder einem anderen Text-Editor aus. Fangen Sie mit Kommentaren über das Datum, Ihren Namen, den Gründen, warum Sie das BIOS ändern, und anderen Informationen an, die Ihnen und anderen vermutlich helfen werden, später auftretende Probleme zu lösen.
4. Spätere Hinzufügungen oder Änderungen sollten nicht zu schwierig sein, vorausgesetzt, Sie verstehen die Assembler-Sprache und wissen, wie man eine gewünschte Einheit manipuliert. Es folgt ein Beispiel einer kleineren Modifikation. Dies könnten Sie als einen Teil in einer COLDSTART-Routine finden:

```
SIGNON:
            DB      14H,ØDH,ØAH   ;clear screen, move down
            DB      'CP/M-8Ø version 2.2', ØDh,ØAh, '$'
MOVEIT:     LXI     H,SIGNON      ;point to signon message
KEEPON:     MOV     A,M           ;get a byte
            CPI     '$'           ;check for end of message
            RZ                    ;. . . end of message
            MOV     C,A           ;not end, get ready
            ; ---------------------------------------------------------------------------------------
            ; WARNING! Your CONOUT routine may destroy or undesirably alter the
            ; contents of the registers used in MOVEIT. To be sure, PUSH the contents of
            ; any registers not needed by CONOUT onto the stack, then POP them off the
            ; stack into the original register(s).
            ; ---------------------------------------------------------------------------------------;
            PUSH    H             ;out of harm's way
            CALL    CONOUT        ;send data to console
            POP     H             ;all safe and sound
            INX     H             ;increment memory location
            JMP     KEEPON        ;do it again
```

Diese Routine gibt die CP/M-80-Bereitschaftsmeldung aus und initialisiert den Bildschirm des Terminals mit 14 hex als Anfang der Meldung. Was aber, wenn Ihr Terminal zur Initialisierung (CLEAR) 04 hex erfordert? Wir wollen das obige BIOS in bezug auf das neue Terminal verändern und die ausgegebene Meldung persönlicher gestalten.

```
SIGNON:
            DB      Ø4H,ØDH,ØAH   ;clear screen, move down
            DB      'Thom Hogan's Vectrola 1.1',ØDH,ØAH
            DB      '----------',ØDH,ØAH,ØAH
            DB      'CP/M-8Ø 2.2-1Ø/17/80 last update', ØDH,ØAH
            DB      'no printer installed',ØDH,ØAH,ØAH,ØAH, '$'
MOVEIT:     LXI     H,SIGNON      ;point to signon message
```

```
KEEPON:   MOV     A,M         ;get a byte
          CPI     '$'         ;check for end of message
          RZ                  ;. . . end of message
          MOV     C,A         ;not end, get ready
          PUSH    H           ;out of harm's way
          CALL    CONOUT      ;send data to console
          POP H               ;all safe and sound
          INX H               ;increment memory location
          JMP     KEEPON      ;do it again
```

Das Beispiel-BIOS meldet folgendes:

```
<clear screen>
CP/M-8Ø version 2.2
A>
```

Die neue Version gibt aus:

```
<clear screen>
Thom Hogan's Vectrola 1.1
-----------------------------------

CP/M-8Ø 2.2-1Ø/17/8Ø last update
no printer installed

A>
```

In diesem Beispiel haben wir nichts Wichtiges geändert. Beachten Sie, daß die Zeile DB ‚$' so verbleibt, daß die Routine weiß, wo die Meldung zu Ende ist.

Es gibt drei typische Änderungen an BIOS:

- Einfügen neuen Materials
- Löschen alten Materials
- Ändern vorhandenen Materials

Wenn Sie in BIOS-Information einfügen, bleibt die vorherige Information unverändert. Sie können Ihre Schritte leicht zurückverfolgen und das BIOS in seiner ursprünglichen Form wiederherstellen. Wir schlagen vor, daß Sie festhalten, wo Sie neues Material eingefügt haben, damit es später leichter aufzufinden ist. Eine Zeile von Bindestrichen (----------) kann individuelle Routinen trennen. Eine Zeile von Gleichheits-Zeichen (= = = = = = = = = =) kann später hinzugefügte Instruktionen hervorheben. Eine BIOS-Sektion könnte folgendermaßen aussehen:

```
; ---------------------------------------------------------------------------------------------
; MODEM ROUTINES – substitute for PUNCH/READER
;
MODEM     EQU     TRUE        ;yes, we have a modem today
DCH       EQU     FALSE       ;no, it isn't a DC Hayes
;
```

```
IF          NOT DCH             ;if standard modem;
MODEMCTL    EQU     Ø3H         ;modem control port
MODEMSBIT   EQU     8ØH         ;modem send control bit
MODEMRBIT   EQU     4ØH         ;modem receive control bit
MODEMDATA   EQU     Ø2H         ;modem data port
ENDIF
;
; = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
; 1Ø/17/81 -    We finally got a D.C. Hayes Modem board.
;              Here are the EQUs that we think will
;              operate our new modem. When we've had a
;              chance to check these out, we'll go back
;              and change the DCH EQU to TRUE.
;
IF          DCH
MODEMCTL    EQU     82H         ;DC Hayes control port
MODEMSBIT   EQU     2           ;modem send control bit
MODEMRBIT   EQU     1           ;modem receive control bit
MODEMDATA   EQU     8ØH         ;modem data port
MODEMCTL2   EQU     81H         ;second control port needed
ENDIF
; = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
            LXI     H,Ø
            DAD     SP
            SHLD    STACK
            etc.
```

Die Routinen für das Modem von Hayes Microcomputer Products sind zu diesem Zeitpunkt noch nicht völlig implementiert, aber Sie können die neu eingefügte Programm-Sektion klar erkennen.

Wenn Sie eine Sektion aus einem existierenden BIOS löschen, vernichten Sie es nicht, machen Sie lieber einen Kommentar daraus. Schießen Sie vor jeder Zeile ein Semikolon ein, damit sie nicht verarbeitet wird. Fügen Sie eine Bemerkung für die Löschung hinzu.

```
RCVSOH:     MVI     B.1         ;timeout = 1 second
            CALL    RECV        ;get sector
            JC      RCVSTOT     ;got timeout
            MDV     D,A         ;D = block number
            MVI     B,1         ;timeout = 1 second
            CALL    RECV        ;get cma'd sect number
; THE NEXT JUMP CANCELLED 1Ø/1/8Ø TO DISABLE TIMEOUT
;           JC      RCVSTOT     ;got timeout
;           CMA                 ;calculate complement
```

```
;               CMP                     ;good sector number?
;               JZ      RCVDATA         ;yes, got data
; END OF DELETED SECTION OF CODE
;
;
;
;
;
```

Anmerkung: Diese Routine wurde MODEM527.ASM von Ward Christensen entnommen – einem Programm der Öffentlichen Dienste, bei der CP/M-Anwendergruppe erhältlich.

Indem Sie eine gelöschte Sektion als Kommentar erhalten, können Sie die Datei leicht in ihre Originalform zurückverwandeln; Sie entfernen einfach die Semikolons.

Schwieriger ist es, einen BIOS-Bereich zu ändern; solche Änderungen sind nicht leicht zu dokumentieren. Um es dennoch zu tun, machen Sie aus der Originalzeile einen Kommentar und fügen die neue Zeile darunter ein. Sollten Sie allerdings viele Änderungen vornehmen, wird das Resultat u. U. schwer lesbar sein. Stellen Sie stets eine Kopie Ihres Original-BIOS sicher; nennen Sie es BIOSOLD.ASM, BIOS001.ASM oder sonstwie, damit Sie merken, daß es nicht die laufende Version ist. Jedesmal, wenn Sie BIOS editieren, sichern Sie eine Kopie der vorherigen Version und datieren Sie sie alle.

5. Bevor Sie aus dem Editor aussteigen, hinterlassen Sie die Anfangsadresse der BIOS-Datei und prüfen Sie die ORG-Direktive (origin) und alle Marken wie „BIOS", „BIAS" oder „OFFSET". Es gibt unterschiedliche Methoden, die Start-Adresse von BIOS zu programmieren, entweder direkt (durch ein Statement wie „ORG 0EF00H) oder durch eine Berechnung der Art, wie sie von Digital Research vorgeschlagen wird:

```
CCP:        EQU     29ØØH           ;for 16K CP/M-8Ø version 1.4
BIOS:       EQU     15ØØH+CCP       ;location of BIOS
            ORG     BIOS
```

oder

```
CCP         EQU     34ØØH           ;for 20K CP/M-8Ø version 2.2
BIOS:       EQU     16ØØH+CCP       ;location of BIOS
            ORG     BIOS
```

Prüfen Sie in diesem Fall das Format Ihres CP/M-80. Wenn es sich gegenüber dem bisherigen System nicht geändert hat, springen Sie nach Schritt 6. Wenn das nicht zutrifft oder wenn Sie es nicht wissen, dann gehen Sie nicht weiter (Format resp. Umfang bedeutet 48K CP/M-80, 56K CP/M-80 usw.).

Leider gibt es keine absolute Regel für das Verhältnis der BIOS-Sektion zur Basis-Sektion (CCP) von CP/M-80. Daß Sie die Anfangsadresse von CP/M-80 kennen, heißt nicht unbedingt, daß Sie die Anfangsadresse der BIOS-Sektion errechnen können. Digital Research hat versucht, dieses zu standar-

disieren, aber mehrere Verteiler und Computerhersteller ändern weiterhin das Muster.

Mit DDT ist es relativ einfach, die BIOS-Sprungtabelle zu lokalisieren:

```
A> DDT <cr>
   DDT vers. 2.2

   -LØ,2 <cr>
   ØØØØ JMP 96Ø3
   ØØ3
   -^C

A>
```

Hier zeigt die JMP-Instruktion auf die Adresse der Warmstart-Routine. Da der Warmstart der zweite Eintrag in die Sprungtabelle ist, müssen drei Bytes von 9603 hex subtrahiert werden, so daß sich als Start von BIOS 9600 hex ergibt.

Ändern Sie also Ihre ORG-Direktive hinsichtlich der gefundenen Adresse – in diesem Beispiel 9600 hex.

6. Assemblieren Sie das neue BIOS. Wenn die Assembly keine Fehler enthält, gehen Sie nach Schritt 7, anderenfalls korrigieren Sie die Fehler vorher.
7. Zum Test eines neuen BIOS führen Sie folgende Schritte aus:
   a. Kreieren Sie ein neues CP/M-80-Image mit MOVCPM **.
   b. Sichern (SAVE) Sie das neue CP/M-80-Image mit „SAVE 34CPMxx.COM“ (xx ist der Umfang des CP/M-80-Systems, z. B. 32).

      Anmerkung: Bei einigen Versionen des CP/M-80 wird MOVCPM Ihnen eine andere Zahl als 34 empfehlen; wenn MOVCPM z. B.

      ```
      READY FOR "SYSGEN"
      ```

      oder

      ```
      "SAVE 39 CPM64.COM"
      ```

      anzeigt, lautet Ihr SAVE-Kommando:

      ```
      SAVE 39 CPM64.COM
      ```

   c. Laden Sie das Abbild von CP/M-80 mit DDT in den Kernspeicher. Wenn Ihre Version des MOVCPM korrekt arbeitet, gelten die folgenden Adressen:

      ```
      BOOT 9ØØH
      CCP   98ØH        (sometimes AØØH if long BOOT)
      BDOS 118ØH        (sometimes 12ØØH if long BOOT)
      BIOS 1F8ØH        (CP/M 2.2)
           1E8ØH        (CP/M 1.4)
      ```

   d. Benutzen Sie DDT, um das assemblierte BIOS in den korrekten Kernspeicherbereich zu laden. Hier die Methode:

```
-IBIOS.HEX <cr>
-Roffset
```

wobei **offset** Ihr BIOS an den richtigen Platz des CP/M-80-Abbildes lädt, nicht an seine mögliche Adresse.

Diese Distanz wird dadurch errechnet, daß man eine Zahl zur eventuellen Adresse von BIOS (der permanenten Adresse) addiert, so daß sich 1F80 hex (Version 2.2) oder 1E80 hex (Version 1.4) ergibt. Um diese Zahl zu errechnen, subtrahieren Sie zuerst die Lade-Adresse (1F80 hex oder 1E80 hex) von der eventuellen Adresse.

Für 56K CP/M-80 2.2 ist

```
DAØØH − 1FBØH = BABØH          FFFF − DAØØ + IF8Ø
```

guter Brauch, den Kernspeicher von 1F80 hex oder 1E80 hex mit Nullen aufzufüllen, bevor BIOS geladen wird; so läßt sich leicht feststellen, ob der Prozeß korrekt vonstatten ging.

e. Verabschieden Sie sich von DDT durch ein ^C und sichern Sie das neue CP/M-80-System mit einem „SAVE 34 CPMxxOK.COM", wobei xx den neuen Systemempfang repräsentiert.

f. Bei CP/M-80 2.2 können Sie auch SYSGEN benutzen:

```
A>SYSGEN CPMxxOK.COM <cr>
```

Ältere Versionen erfordern für SYSGEN:

```
A>DDT CPMxxOK.COM<cr>
−^C
A>SYSGEN<cr>
```

Das IOBYTE

Das Konzept des IOBYTE geht CP/M-80 um mehrere Jahre voraus. Gary Kildall hat es bereits in Beispiels-Implementationen von CP/M-80 benutzt, und es wird in den meisten Handbüchern von Digital Research dokumentiert. Andere Software-Erfinder, die BIOS-Module entwickelt haben, haben das Konzept des IOBYTE ausgearbeitet, aber seine Basisfunktionen nicht verändert.

Das IOBYTE zeigt die laufende Zuordnung von physischen zu logischen Einheiten an. Im CP/M-80 haben Sie die folgenden vier logischen Einheiten:

CON:	Console device
LST:	List device
RDR:	Reader device
PUN:	Punch device

Wenn Sie zwei verschiedene Drucker, Terminals oder Lochstreifenleser haben, zeigt das IOBYTE an, welches von beiden Geräten benutzt werden soll.

Die normale Adresse des IOBYTE ist 0003 hex. Das Byte wird in vier separate 2-Bit-Indikatoren aufgeteilt.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LIST		PUNCH		READER		CONSOLE	

Tabelle 7–3 zeigt, wie CP/M die zwei Bits für jede Einheit interpretiert. Die Einheitennamen sollten Ihnen bekannt sein, denn sie bezeichnen die Geräte, die von PIP und STAT adressiert werden. Ein IOBYTE-Wert von 00100100 bit (oder 24 hex) bedeutet, daß die physischen Einheiten z. Zt. folgenden logischen Funktionen zugeordnet sind:

- Die TTY: -Einheit betreibt die CONSOLE-Funktion
- Die PTR: -Einheit betreibt die READER-Funktion
- Die UPI: -Einheit betreibt die PUNCH-Funktion
- Die TTY: -Einheit betreibt die LIST-Funktion.

Eine BIOS-Routine, die Sie oder Ihr Computerverkäufer geschrieben haben, prüft das IOBYTE auf die Sende- oder Empfangsadresse der nötigen Information. Die Sprünge in der BIOS-jump-table sollten alle auf spezielle Routinen hinweisen, die über den Konsolenstatus, die Konsoleneingabe, die Konsolenausgabe, den Leser, den Drucker und den Stanzer Auskunft geben. Diese Routinen verfahren folgendermaßen:

1. Lies das IOBYTE.
2. Entscheide, welche Einheit benötigt wird.
3. Rufe die Routine für diese Einheit auf.

Plattenbetrieb

Eine gewisse Schilderung, wie CP/M-80 eine Information auf Diskette speichert, gehört auch zu diesem Kapitel. Leider läßt sich diese Information nicht exakt angeben, weil jede Art von Laufwerk auch einen etwas unterschiedlichen Satz von Parametern erfordert. Um Sie nicht zu verwirren, wollen wir hier nur über die 8-inch single-density CP/M-80 Version 2.2

Tabelle 7–3. Interpretation des IOBYTE

Logische Funktionseinheit	Physische Einheit			
	00	01	10	11
Konsole	CON:	TTY:	CRT:	BAT:	UC1:
Leser	RDR:	TTY:	PTR:	UR1:	UR2:
Stanzer	PUN:	TTY:	PTP:	UP1:	UP2:
Lister	LST:	TTY:	CRT:	LPT:	UL1:

(= 8-Zoll einfache Dichte CP/M-80 Version 2.2) sprechen und die Unterschiede zu anderen Versionen aufzuzeigen versuchen.

Sie wissen bereits, daß diese Information auf Diskette in „Sektoren" gespeichert ist, welche „Spuren" von Information in Blöcke unterteilen. Jeder Sektor enthält 128 Daten-Bytes; und bei „single-density 8-inch-Disketten" enthält jeder der 77 Spuren 26 Sektoren. Einige der gebräuchlichen Plattenformate unterscheiden sich also folgendermaßen:

Format	**Anzahl der Sektoren**	**Anzahl der Spuren**
8 Zoll einfache Dichte	26	77
8 Zoll doppelte Dichte	26	77
North Star einfache Dichte	20	35
North Star doppelte Dichte	40	35
Micropolis MOD II	32	77
Intertec Superbrain	40	35
Apple	32	35
Altos doppelte Dichte	48	77
Pickles and Trout TRS-80	64	77
Osborne 1 (256-Byte-Sektoren)	10	40

Bei den meisten Implementationen von CP/M-80 sind die ersten beiden Spuren als „System-Spuren" reserviert. So wird CP/M-80 von SYSGEN gespeichert, und manchmal enthält dieser Bereich auch Spezialroutinen, die nur von Ihrem jeweiligen Computer angewandt werden. Für das Inhaltsverzeichnis wird diese dritte Spur auf den meisten Floppy-Disketten besonders reserviert. Die restlichen Spuren sind für Daten freigehalten.

CP/M-80 erlaubt es, Dateisektoren in einer „Gruppe" zusammenzufassen. Eine Gruppe besteht aus einer festen Sektorenanzahl, 8 oder 16. CP/M-80 wird in jedem Falle für eine Datengruppe 1.024 Bytes freihalten (8 x 128 = 1024), obwohl ein „Block" nur 128 Bytes entspricht und selbst dann, wenn von dieser K-Einheit (1024 Bytes) nur 1 Byte benutzt wird. Obwohl diese Praxis Leerzeichen absorbiert, erlaubt es dem Inhaltsverzeichnis (Directory) mehr Effizienz, weil es über Spuren anstelle von individuellen Sektoren Buch führt.

Wenn an Ihr System eine Hartplatte oder ein Floppy-Laufwerk von doppelter Dichte angeschlossen ist, ist es wahrscheinlich, daß Sie pro Gruppe mehr als 8 Sektoren haben, also möglicherweise Gruppen von 2028, 4056 oder sogar 8112 Bytes.

Jeder Directory-Eintrag ist aus 32-Byte-Blöcken zusammengesetzt:

- Byte 1 zeigt an, ob die Datei aktiv oder gelöscht ist
- Byte 2 bis 9 enthält den Dateinamen
- Byte 10 bis 12 enthält den Dateityp

- Byte 13 ist die Extent-Nummer (= Bereichsnummer)
- Byte 14 und 15 sind für internen Gebrauch reserviert
- Byte 16 ist der Extent-Umfang in Sektoren
- In den Bytes 17 bis 32 sind die der Datei zugeordneten Gruppen gespeichert.

CP/M-80 benutzt auch ein Konzept, das „extents" genannt wird. Ein Extent ist eine Gruppe von Gruppen; wiederum ist die Anzahl gewöhnlich 8 oder 16. Jeder Directory-Eintrag gilt für ein Extent einer gegebenen Datei. Wenn die Datei so groß ist, daß 16 Gruppen zu ihrer Speicherung nicht ausreichen, wird ein weiterer Directory-Eintrag für die Datei vorgenommen – das zweite „extent". Der maximale Umfang einer Datei wird durch die maximale Anzahl von Extents limitiert, die Ihre Version des CP/M-80 zuläßt, wobei die Anzahl der Sektoren pro Gruppe mit 128 multipliziert wird.

Das ist nicht so schwierig, wie es scheint. Eine Datei besteht aus einer Anzahl von Extents, jede davon aus einer Anzahl von Gruppen, davon jede aus einer Anzahl von Sektoren. Um Ihnen einen Begriff von diesem Multi-Extent-Konzept zu geben, folgen hier vier Directory-Eintragungen, wie sie im hexadezimalen und ASCII-Format erscheinen würden (s. a. die direkt angeschlossene FCB-Sektion):

```
ØØ 41 53 4D 2Ø 2Ø 2Ø 2Ø 2Ø 43 4F 4D ØØ ØØ ØØ 4Ø .ASM     COM...@
1B 1C 1D 1E ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ................

ØØ 5A 53 4D 2Ø 2Ø 2Ø 2Ø 2Ø 43 4F 4D ØØ ØØ ØØ 3B .ZSM     COM...;
1F 2Ø 21 22 ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ . !."............

ØØ 4D 38 3Ø 2Ø 2Ø 2Ø 2Ø 2Ø 43 4F 4D ØØ ØØ ØØ 8Ø .MBØ     COM....
23 24 25 26 27 28 29 2A ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ #$%&'()* ......

ØØ 4D 38 3Ø 2Ø 2Ø 2Ø 2Ø 2Ø 43 4F 4D Ø1 ØØ ØØ Ø9 .M8Ø     COM....
2B ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ +...............
```

Dieses Inhaltsverzeichnis kommt von einer Diskette, die der Autor im Zusammenhang mit seinem Vector Graphic Computer-System benutzt. Die spezielle Implementation von CP/M-80 erlaubt 8 Gruppen pro Extent, aber die Datei M80.COM besetzt 9 Gruppen (23 hex bis 2B hex). So ist das zweite Extent der Datei direkt hinter dem Namen mit 01 hex markiert, das erste mit 00 hex.

FCBs – File Control Blocks (= Datei-Kontrollblöcke)

Der FCB ist eine durch CP/M reservierte Kernspeichersektion, um Information über die z. Zt. benutzte Datei festzuhalten. Er ist 36 Bytes lang und beginnt normalerweise bei der Adresse 005C hex. Hier sind die Bytes des Standard-CP/M-FCB (für CP/M-80 wie auch CP/M-86), und wofür sie benutzt werden:

Byte-Nr.	Funktion
1	Laufwerkscode 0 = benutze Standard-Laufwerk 1 = benutze Laufwerk A 2 = benutze Laufwerk B
2–9	Dateiname Acht Bytes; die ungenutzten Stellen werden mit ASCII-Leerzeichen aufgefüllt.
10–12	Dateityp Drei Bytes; die ungenutzten Stellen werden mit ASCII-Leerzeichen aufgefüllt Das höchstwertige Bit von Byte 10 wird benutzt, um zu zeigen, daß eine Datei R/O (schreibgeschützt) ist Das höchstwertige Bit von Byte 11 wird benutzt, um eine SYS-Datei anzuzeigen
13	Die laufende Extent-Nummer der Datei
14	Reserviert für den Gebrauch von CP/M
15	Reserviert für den Gebrauch von CP/M; sollte bei den Funktionen OPEN, MAKE oder SEARCH FOR FILE auf Null gesetzt werden
16	Satzzähler für das laufende Extent
17–32	Von CP/M mit Information über Satz-Adressen in der Datei aufgefüllt
33	Laufende Satz-Nummer zum Lesen oder Schreiben in einer sequentiellen Datei-Operation
34–35	DA-Satz-Nummer (Direct Access = direkter Zugriff)
36	Anzeige für Überlauf der DA-Satz-Nummer (Random)

Platte-Parameter-Tabelle

In diesem Kapitel wurde der Platten-Parameter-Header (= Kopfetikett) bereits kurz erwähnt. Dies ist ein spezieller Bereich des BIOS-Kernspeichers, der für einige wichtige Informationen reserviert ist, die CP/M-80 kennen muß, um Ihre Plattenlaufwerke richtig zu bedienen. Jedem Laufwerk ist ein 16-Byte-Parameter als Header zugeordnet, der die Information über das Laufwerk enthält und etwas Platz für temporäre Informationen von BDOS freiläßt.

Bytes	Titel	Beschreibung
1–2	XLT	Übersetzungsvektor: die Adresse der Sektorübertragungstabelle, die den Schrägungsfaktor für das Laufwerk enthält. 0000 hex bedeutet, daß keine Übersetzung vorgenommen wird. Laufwerke mit gleichem Schrägungsfaktor können dieselben Übersetzungstabellen benutzen.

3–8	entfällt	RAM-Notizblock-Speicher für den Gebrauch von BDOS.
9–10	DIRBUF	Directory-Puffer: die Adresse eines RAM-Bereiches von 128 Byte für Directory-Operationen des Laufwerkes. Alle Laufwerke teilen den gleichen Directory-Puffer.
11–12	DPB	Disk Parameter Block (= Platten-Parameter-Block): die Adresse des DPB. Laufwerke mit identischen Charakteristika können ihn gemeinsam benutzen.
13–14	CSV	Prüfung auf Diskettenwechsel: die Adresse eines „scratchpad"-Bereiches (= Notizblock-Speicherbereich) für die Prüfung auf Diskettenwechsel. Die Adresse muß für jedes Laufwerk unterschiedlich sein.
15–16	ALV	Disk Allocation Vector (= Platten-Zuordnungsvektor): die Adresse eines „scratchpad"-Bereiches für Information über die Zuteilung von Speicherbereich für das Laufwerk. Die Adresse muß für jedes Laufwerk unterschiedlich sein.

Die Platten-Parameter-Kopfetiketten für jedes Laufwerk erscheinen geordnet, eins direkt nach dem anderen. Die SELDSK-Routine in BIOS sollte die Basis-Adresse des Headers für das ausgewählte Laufwerk wiedergeben. Wenn ein Laufwerk nicht existiert, gibt SELDSK 0000 hex zurück.

Die Übersetzungsvektoren, auf die sich der DPH bezieht, können irgendwo im BIOS stehen, und sie sind einfach eine Auflistung der Sektoren, wie sie nacheinander gelesen werden sollen. Eine gültige Sektorenübersetzungstabelle könnte etwa folgendermaßen aussehen:

```
DISKXLATE:  dB   1, 7,13,19,25
            dB   5,11,17,23, 3
            dB   9,15,21, 2, 8
            dB  14,20,26, 6,12
            dB  18,24, 4,10,16
            dB  22
```

Die hier gezeigte Übersetzungstabelle enthält einen Schrägungsfaktor von sechs (das Laufwerk liest einen Sektor, danach den um die Anzahl sechs versetzten Sektor usw.). Die Änderung der Übersetzungstabelle ist eine Möglichkeit, die Plattengeschwindigkeit zu erhöhen.

Der DPB teilt CP/M-80 das Layout des Laufwerks mit:

Bytes	**Titel**	**Beschreibung**
1–2	SPT	Anzahl der Sektoren pro Spur
3–4	BSH/BLM	Blockverschiebungsfaktor
5	EXM	Zuweisungsumfang für einen Datenblock
6–7	DSM	Höchste Nummer eines Datenblocks

8–9	DRM	Anzahl der Directory-Eintragungen
10–11	ALO/1	Directory-Gruppenzuweisung
12–13	CKS	Prüfe Directory-Eingang
14–15	OFF	Anzahl der am Plattenanfang übersprungenen Sektoren

CP/M-80

Weitere Information über die interne Struktur von CP/M-80 finden Sie in den bibliographischen Hinweisen im Anhang F. Über spezifische Bestandteile des CP/M-80 sind eine Anzahl ausgezeichneter Artikel veröffentlicht worden.

Unterschiede zwischen CP/M-80 und CP/M-86

CP/M-80 und CP/M-86 wurden für die Arbeit mit unterschiedlichen CPUs entworfen. CP/M-80 arbeitet mit den Chips 8080, 8085 und Z80, CP/M-86 nur mit 8086 und 8088. Die ersteren sind 8-Bit-, die letzteren 16-Bit-Chips (korrekter ausgedrückt ist der 8088 ein 16-Bit-Chip mit einem 8-Bit-Datenbus).

Einige wenige Unterschiede zwischen CP/M-80 und CP/M-86 sind für Benutzer wichtig. In erster Linie kann CP/M-80 bis zu 1 MB (1 Megabyte = 1024 x 1024 Byte) adressieren (und damit benutzen), während CP/M-80 auf 64 K begrenzt ist (die Version 3.0 erlaubt zusätzliche 64 K Kernspeicher). CP/M-80 bezieht sich in absoluter Weise auf Kernspeicheradressen; d. h., ihm steht der Bereich zwischen 0000 hex und FFFF hex zur Verfügung. CP/M-80 bezieht sich auf Kernspeicheradressen durch eine „offset"-Methode (offset = Distanz), die auf den Werten in den Segmentregistern von 8086- oder 8088-Chips basiert.

Anstatt wie CP/M-80 einen BDOS-Call (= Aufruf) auf Adresse 0005 hex anzuwenden, benutzen CP/M-86-Programme den Software-Interrupt (= Unterbrechung) Nr. 244 für die gleiche Funktion; zuerst werden die Register mit den angemessenen Werten geladen, danach erfolgt der Interrupt. BDOS-Aufrufe des CP/M-86 benutzen nicht die Adressen 0006 und 0007 hex; CP/M-80 hält diese Adressen für aus dem CP/M-80-Environment übersetzte Programme frei, um den Kernspeicherumfang zu ermitteln.

Um die folgenden Beschreibungen der BDOS-Funktionen zu verstehen, müssen diese 8086-Register erklärt werden. Acht 8-Bit-Register stehen für allgemeine Zwecke zur Verfügung, und sie sind alle als 16-Bit-Registerpaare gruppiert.

8-Bit-Register	werden zu	**16-Bit-Registerpaaren**
AH + HL		AX
BH + BL		BX
CH + CL		CX
DH + DL		DX

Außerdem hat der 8086-Chip vier 8-Bit-Indexregister (auch Basis-Register genannt).

BP
SP
SI
DI

Schließlich sind vier Segmentregister (benutzt zur Errechnung erweiterter Adressierung über mehr als 64K) verfügbar:

Code
Data
Stack (= Stapel)
Extra

BDOS-Aufrufe für CP/M-86 werden in Tabelle 7-4 detailliert beschrieben.

In den meisten Fällen werden mit CP/M vertraute Programmierer CP/M-86 recht ähnlich finden. Zu dem Zeitpunkt, als dieses Buch überarbeitet wurde, hatte Digital Research gerade eine Neufassung des CP/M-86 vollendet und auch eine „Concurrent CP/M-86" genannte Version angekündigt. Speziell an den Besonderheiten von CP/M-86 interessierte Leser sollten sich an Digital Research wenden (die Adresse finden Sie in Anhang H).

Tabelle 7–4. BDOS-Funktionen für CP/M-86*

Funktion		Eingangs-Parameter	Ausgangs-Parameter	Erklärung
Nr.	Name			
00	SYSTEM RESET	DL = Abbruch-Code	Keine	Startet CP/M-86 neu, indem es nach Reinitialisierung des Platten-Subsystems die Kontrolle an den CCP übergibt. Der Abbruch-Code in DL wird auf 00 hex gesetzt, um das z. Zt. aktive Programm zu beenden und die Kontrolle an den CCP zurückzugeben. Wenn DL 01 hex ist, verbleibt das Programm im Kernspeicher und der Status der Speicherzuordnung unverändert.
01	CONSOLE INPUT	Keine	AL = ASCII-Zeichen	Übergibt das nächste getippte Zeichen an das aufrufende Programm. Jedes nichtdruckbare Zeichen wird auf dem Bildschirm geechot (wie BACKSPACE, TAB oder CARRIAGE RETURN). Die Ausführung durch das aufrufende Programm wird nicht fortgesetzt, solange kein Zeichen eingegeben ist. Standard-CCP-Kontrollzeichen werden erkannt und ihre Funktion durchgeführt (CONTROL-P beginnt oder beendet Druckwiedergabe usw.).
02	CONSOLE OUTPUT	DL = ASCII-Zeichen	Keine	Zeigt das Zeichen im DL-Register auf der Konsole an. Standard-CCP-Kontrollzeichen werden erkannt und ihre Aktion ausgeführt (CONTROL-P beginnt oder beendet die Druckwiedergabe usw.).
03	READER INPUT	Keine	AL = ASCII-Zeichen	Übergibt das nächste Zeichen von der Leseeinheit an das aufrufende Programm. Die Ausführung wird nicht fortgesetzt, solange das Zeichen nicht empfangen wurde.
04	PUNCH OUTPUT	DL = ASCII-Zeichen	Keine	Übermittelt das Zeichen im DL-Register zur Stanzeinheit.
05	LIST OUTPUT	DL = ASCII-Zeichen	Keine	Übermittelt das Zeichen im DL-Register zur Listeinheit.
06	DIRECT CONSOLE IN	DL = FF	AL = ASCII-Zeichen oder 00	Wenn Register DL FF hex enthält, wird die Konsoleneinheit befragt, ob ein Zeichen abrufbereit ist. Sonst wird dem aufrufenden Programm in Register A ein 00 übergeben, anderenfalls das erkannte Zeichen. Wenn Register DL ein anderes Zeichen als FF hex oder FE hex enthält, wird dieses Zeichen an die Konsolanzeige übermittelt. Alle CCP-Kontrollzeichen werden ignoriert. Der Anwender muß das Programm gegen unsinnige Zeichen schützen, die von der Konsoleinheit ausgehen oder empfangen werden.
	DIRECT CONSOLE STATUS	DL = FE	AL = Status	
	DIRECT CONSOLE OUT	DL = ASCII-Zeichen	Keine	

* **Anmerkung:** Bei CP/M-86 wird die Funktionsnummer in das CL-Register geladen. Dies entspricht dem Laden des Registers C in CP/M-80.

Tabelle 7–4. BDOS-Funktionen für CP/M-86* (Fortsetzung)

Funktion		Eingangs-Parameter	Ausgangs-Parameter	Erklärung
Nr.	Name			
07	GET IOBYTE	Keine	AL = IOBYTE	Kopiert das I/O-Byte nach Register AL, bevor es die Kontrolle an das aufrufende Programm zurückgibt.
08	SET IOBYTE	DL = IOBYTE	Keine	Kopiert den Wert in Register DL auf die Adresse des I/O-Bytes, bevor es die Kontrolle an das aufrufende Programm zurückgibt.
09	PRINT STRING	DX = Adresse der Zeichenkette	Keine	Sendet die Zeichenkette, deren Adresse im DX-Registerpaar gespeichert ist, an die Konsole. Alle nachfolgenden Zeichen werden gesendet, bis BDOS ein ASCII „$" (24 hex) entdeckt. CCP-Kontrollzeichen werden entdeckt und ihre Funktion ausgeführt.
0A	READ CONSOLE BUFFER	DX = Pufferadresse	Daten im Puffer	Diese Funktion führt im wesentlichen dasselbe aus wie CCP, indem sie die Zeichen, die der Benutzer eingibt, übernimmt und sie in den Puffer an der im Registerpaar DX gespeicherten Adresse speichert. Das erste durch das DX-Paar angezeigte Byte im Puffer muß die maximale Länge des Kommandos enthalten; BDOS wird die im zweiten Byte gezählte Anzahl von Zeichen durch das getippte Kommando beginnend mit dem dritten Byte der Kette, die das Registerpaar DX anzeigt, plazieren. Alle Standard-CCP-Edit-Zeichen werden während des Kommandoeingangs erkannt.
0B	GET CONSOLE STATUS	Keine	AL = Status	BDOS prüft den Status der Konsoleinheit und übergibt 00 hex, wenn kein Zeichen bereit ist, und 01 hex, wenn ein Zeichen getippt wurde.
0C	GET VERSION NUMBER	Keine	BX = Version	Wenn das im Register BH angegebene Byte 00 hex ist, so ist das gegenwärtige Betriebssystem CP/M; bei BL = 01 hex ist es MP/M. Das in Register BL übergebene Byte ist 00 hex, wenn die Version vor CP/M 2.0 liegt, 20 hex, wenn sie 2.0 ist, 21 hex, wenn es sich um 2.1 handelt, 22 hex für 2.2 usw.
0D	RESET DISK SYSTEM	Keine		Wird angewandt, um CP/M die Anweisung zu geben, das Platten-Subsystem auf Null zu setzen. Sollte bei jedem Diskettenwechsel eingegeben werden.

* **Anmerkung:** Bei CP/M-86 wird die Funktionsnummer in das CL-Register geladen. Dies entspricht dem Laden des Registers C in CP/M-80.

Tabelle 7–4. BDOS-Funktionen für CP/M-86* (Fortsetzung)

Funktion		Eingangs-Parameter	Ausgangs-Parameter	Erklärung
Nr.	Name			
0E	SELECT DISK	DL = Platten-Nummer	Keine	Wählt die Plattennummer für nachfolgende Plattenoperationen aus. 00 hex in Register DL zeigt Laufwerk A:, 01 hex Laufwerk B: an, usw.
0F	OPEN FILE	DX = FCB-Adresse	AL = DIR-Code	Wird benutzt, um eine Datei auf dem z. Zt. aktiven Laufwerk unter der gegenwärtigen Benutzernummer zu aktivieren. BDOS tastet die ersten 14 Bytes des angezeigten FCB-Blocks ab und versucht, den betreffenden Dateinamen im Block zu finden. Ein ASCII „?" (3F hex) kann an jeder der Stellen des Dateinamens benutzt werden, um ein „don't care"-Zeichen anzuzeigen. Wenn ein Dateiname gefunden wird, wird von CP/M-86 die relevante Information über die Datei in den Rest des FCB eingetragen. Ein Wert von 00 bis 03 hex in Register AL zeigt an, ob die Operation erfolgreich war, während FF hex bedeutet, daß die Datei nicht gefunden wurde. Wenn Fragezeichen zur Identifizierung einer Datei benutzt wurden, wird der erste zugehörige Eintrag angenommen.
10	CLOSE FILE	DX = FCB-Adresse	AL = DIR-Code	Übt das Gegenteil der OPEN FILE-Funktion aus. Die CLOSE FILE-Funktion muß jedesmal nach Gebrauch einer Datei ausgeführt werden, in die eine Information geschrieben wurde.
11	SEARCH FOR FIRST	DX = FCB-Adresse	AL = DIR-Code	Erfüllt dasselbe wie die OPEN FILE-Funktion mit dem Unterschied, daß der laufende PlattenPuffer mit dem 128-Byte-Satz gefüllt wird, der den Directory-Eintrag der zugehörigen Datei enthält.
12	SEARCH FOR NEXT	Keine	AL = DIR-Code	Bewirkt dasselbe wie die SEARCH FOR FIRST-Funktion, außer daß die Suche vom letzten gefundenen Eintrag an fortgesetzt wird.
13	DELETE FILE	DX = FCB-Adresse	AL = DIR-Code	Setzt auf dem Directory-Eingang für die vom FCB angezeigte Datei ein „flag" (Kennzeichen), so daß CP/M-86 sie nicht länger als gültige Datei anerkennt. Tatsächlich wird keine Information bei Ausführung dieser Funktion zerstört, obwohl untergeordnete Disketten-Schreibbefehle einige der Bereiche benutzen mögen, die vorher mit der „gelöschten" Datei assoziiert waren.

* **Anmerkung:** Bei CP/M-86 wird die Funktionsnummer in das CL-Register geladen. Dies entspricht dem Laden des Registers C in CP/M-80.

Tabelle 7–4. BDOS-Funktionen für CP/M-86* (Fortsetzung)

Funktion		Eingangs-Parameter	Ausgangs-Parameter	Erklärung
Nr.	Name			
14	READ SEQUENTIAL	DX = FCB-Adresse	AL = Fehler-Code	Wenn eine Datei durch den Gebrauch einer OPEN FILE- oder MAKE FILE-Funktion aktiviert wurde, liest die READ SEQUENTIAL-Funktion den nächsten 128-Byte-Block der laufenden DMA-Adresse. Im Register AL wird der Wert 00 hex gemeldet, wenn das „READ" erfolgreich verlaufen ist, während jedes Nicht-Nullzeichen im Register AL einen Fehler anzeigt.
15	WRITE SEQUENTIAL	DX = FCB-Adresse	AL = Fehler-Code	Wenn eine Datei für den Gebrauch einer OPEN FILE- oder MAKE FILE-Funktion aktiviert wurde, schreibt die WRITE SEQUENTIAL-Funktion den 128-Byte-Block des Kernspeichers an der laufenden DMA-Adresse in den nächsten 128-Byte-Satz der genannten Datei.
16	MAKE FILE	DX = FCB-Adresse	AL = DIR-Code	Erstellt eine neue Datei mit der durch den FCB angezeigten Information (Name). CP/M-86 prüft nicht, ob die Datei bereits existiert, deshalb müssen Sie prüfen, ob diese Datei existiert (oder sie nötigenfalls zerstören). Eine neu erstellte Datei muß nicht unbedingt eröffnet werden, da die MAKE FILE-Funktion auch die notwendigen OPEN-Operationen enthält.
17	RENAME FILE	DX = FCB-Adresse	AL = DIR-Code	Ändert den durch die ersten 16 Bytes des FCB angegebenen Datei-Namen in den Namen der zweiten 16 Bytes.
18	RETURN LOGIN VECTOR	Keine	BX = eingeschaltete Platten	Die Bits im BX-Register spezifizieren, welche Laufwerke aktiv sind. Das erste Bit bezieht sich auf Laufwerk A:, das letzte korrespondiert mit Laufwerk P, der höchstmöglichen Laufwerkangabe. Ein Bit 1 zeigt einen aktiven Status an, Bit 0 ein inaktives Laufwerk.
19	RETURN CURRENT DISK	Keine	AL = z. Zt. aktive Platte	Die Nummern 0 bis 15 bezeichnen nach Rückkehr aus dieser Funktion das z. Zt. aktive Standardlaufwerk.
1A	SET DMA ADDRESS	DX = DMA-Adresse	Keine	Wird benutzt, um den Kernspeicher-Block (128 Bytes) für die Pufferung aller Plattenübertragungen auszuwählen. Beim RESET von System oder Platte, bei einem Kalt- oder Warmstart, wird die Pufferadresse in einem normalen CP/M-86-System auf 0080 hex gesetzt.

* **Anmerkung:** Bei CP/M-86 wird die Funktionsnummer in das CL-Register geladen. Dies entspricht dem Laden des Registers C in CP/M-80.

Tabelle 7–4. BDOS-Funktionen für CP/M-86* (Fortsetzung)

Funktion		Eingangs-Parameter	Ausgangs-Parameter	Erklärung
Nr.	Name			
1B	GET ALLOC ADDRESS	Keine	BX = Zuordnungs-Adresse ES = Segment-Basis	Übergibt die Startadresse des Zuordnungsvektors, einer Tabelle, die im Kernspeicher für jedes „on line"-Laufwerk (jedes z. Zt. angeschlossene Laufwerk) geführt wird und den Teil der Diskette anzeigt, der gerade im Gebrauch ist.
1C	WRITE PROTECT DISK	Keine	Keine	Bewirkt einen temporären Schreibschutz für die Diskette im derzeitigen Standardlaufwerk.
1D	GET R/O VECTOR	Keine	BX = Laufwerk-R/O	Übergibt im Registerpaar BX einen 16-Bit-Wert, der anzeigt, welche Laufwerke im System schreibgeschützt sind. Die Laufwerke werden wie im LOGIN VECTOR mit dem Bitwert 1 für Schreibschutz angezeigt.
1E	SET FILE ATTRIBUTES	DX = FCB-Adresse	AL = DIR-Code	Setzt die Datei-Attribute im zugehörigen Teil des in DX angezeigten FCB.
1F	GET DISK PARMS	Keine	BX = DPB-Adresse ES = Segment-Basis	Ermittelt die Distanz und Segmentbasis des DPB für das derzeitige aktive Laufwerk.
20	GET USER CODE SET USER CODE	DL = FF DL = Benutzer-Code	AL = Derzeitiger Benutzer Keine	Wenn das DL-Register 0FF hex enthält, wird die laufende Benutzernummer zurückgegeben; im anderen Fall wird sie auf den Wert im DL-Register gesetzt.
21	READ RANDOM	DX = FCB-Adresse	AL = Fehler-Code	Liest den vom DA-Satz-Teil des FCB angezeigten DA-Satz (direct access = direkter Zugriff).
22	WRITE RANDOM	DX = FCB-Adresse	AL = Fehler-Code	Schreibt den DA-Satz aus dem im laufenden DMA-Puffer angezeigten Teil des FCB.
23	COMPUTE FILE SIZE	DX = FCB-Adresse	RRF-Set	Füllt den Dateiumfang in den FCB ein.
24	SET RANDOM RECORD	DX = FCB-Adresse	RRF-Set	Übergibt den nächsten DA-Satz (fügt seinen Wert in das DA-Satz-Feld des FCB ein) nach dem zuletzt sequentiell gelesenen Satz.

* **Anmerkung:** Bei CP/M-86 wird die Funktionsnummer in das CL-Register geladen. Dies entspricht dem Laden des Registers C in CP/M-80.

Tabelle 7–4. BDOS-Funktionen für CP/M-86* (Fortsetzung)

Funktion		Eingangs-Parameter	Ausgangs-Parameter	Erklärung
Nr.	Name			
25	RESET DRIVE	DX = Nullstellung der Laufwerk-Bits	AL = Fehler-Code	Bewirkt die Nullsetzung der spezifizierten Laufwerk-Bits als nicht-angeschlossenen Status.
28	WRITE RANDOM (ZERO)	DX = FCB-Adresse	AL = Fehler-Code	Schreibt vor der Information im DMA-Puffer Nullen in den DA-Satz.
32	DIRECT BIOS CALL	DX = BIOS-Deskriptor	Keine	Erlaubt BDOS, direkt auf BIOS-Routinen zuzugreifen. Der erste 8-Bit-Wert an der von DX angezeigten Adresse ist die BIOS-Funktionsnummer, die nächsten zwei 16-Bit-Adressen werden in die Registerpaare CX und DX von 8086 geladen, bevor der BIOS-Aufruf ausgelöst wird.
33	SET DMA BASE	DX = Basis-Adresse	Keine	Setzt die Basisadresse für nachfolgende DMA-Übertragungen.
34	GET DMA BASE	Keine	BX = DMA-Offset ES = DMA-Segment	Übergibt die Basisadresse der letzten DMA-Übertragung.
35	GET MAX MEM	DX = MCB-Offset	AL = Return-Code	Findet den größten verfügbaren Kernspeicherbereich, der mindestens die im laufenden MCB (Memory Control Block = Kernspeicher-Kontrollblock) enthaltene Anzahl von Bytes besitzt.
36	GET ABS MAX	DX = MCB-Offset	AL = Return-Code	Findet den größtmöglichen Kernspeicherbereich an der vom laufenden MCB angezeigten absoluten Paragraphengrenze.
37	ALLOC MEM	DX = MCB-Offset	AL = Return-Code	Weist entsprechend dem MCB Kernspeicher von der von DX angezeigten Adresse an zu.
38	ALLOC ABS MEM	DX = MCB-Offset	AL = Return-Code	Entspricht Funktion 37, außer daß in der Berechnung die absolute Basis-Adresse benutzt wird.
39	FREE MEM	DX = MCB-Offset	Keine	Gibt Speicherbereiche frei, die von einem Programm durch den von DX angezeigten MCB zugeordnet wurden.
3A	FREE ALL MEM	Keine	Keine	Gibt den ganzen Kernspeicher für den Systemgebrauch frei.
3B	PROGRAM LOAD	DX = FCD-Offset	AX = Return-Code BX = Basis-Page-Adresse	Lädt eine „.CMD"-Datei.

* **Anmerkung:** Bei CP/M-86 wird die Funktionsnummer in das CL-Register geladen. Dies entspricht dem Laden des Registers C in CP/M-80.

Kapitel 8
Systemkriterien

Obwohl CP/M ein relativ einfaches Betriebssystem ist, hat man es nicht immer verstanden. Während der ersten Jahre seiner Existenz wurde häufig die Dokumentation kritisiert, und manchmal wurden Hardware-Probleme als Fehler von CP/M hingestellt.

Der Verkauf von Computersystemen schließt die Verantwortung für gründliches Training und die Zuverlässigkeit des Produktes ein. In dieser Hinsicht hat die Microcomputerindustrie des öfteren gesündigt. Digital Research hat die Aufgabe, Benutzer für den Gebrauch von CP/M-80 und CP/M-86 zu schulen, den Computerherstellern und -händlern überlassen. Einige schulen Computerbenutzer gut, viele haben Schwierigkeiten mit diesem Problem.

Es ist nicht ausreichend, sich einfach über die Nachlässigkeit der Industrie zu beklagen; das Thema dieses Kapitels ist: wie lernt der Endanwender den Gebrauch von CP/M? Dieses Buch gibt eine Antwort, aber es ist dennoch nur eine von vielen benötigten Hilfen.

Auf den folgenden Seiten wird eine Reihe von Empfehlungen für CP/M-Benutzer geboten. Darunter befindet sich eine Systemlösung für möglicherweise auftretende Probleme. Für unsere Zwecke wollen wir den Begriff „System" hier so auslegen, daß er die Gesamtheit von Personen, Geräten, Papier, Information und Hilfsmitteln einschließt, die bei der Benutzung des Computers beteiligt ist. Einen Computer, etwas Software und dieses Buch zu kaufen, ist nicht genug. Sie müssen jede Komponente des Systems in Ihre derzeitigen Bedürfnisse und Abläufe integrieren.

Systemempfehlungen

Irreführende Werbung und fehlende Erfahrung bei den Verbrauchern führen häufig dazu, daß einfach das billigste Angebot gekauft wird. Wenn zwei Computerverkäufer das System XY anbieten, kauft der Kunde bei der Firma mit dem niedrigsten Preis. Das ist gefährlich. Während es möglich ist, diese Annäherung nach rein persönlichen Gesichtspunkten auszuwerten, wird man bei der Auswahl eines Computers für Geschäftszwecke unter Umständen daran scheitern, daß einige der Versprechungen einfach nicht eingehalten werden.

Bei der Entscheidung, wo Sie Ihren Computer kaufen, sollte die gebotene Unterstützung eine mindestens ebenso wichtige Rolle spielen wie der Preis. Unterstützung ist ein unbestimmter Begriff, der solche Faktoren einschließt wie:

- Training des Personals im Gebrauch der Anlage
- Aufbau und Prüfung des Computers

- Service in Notfällen
- ob der Verkäufer sich Mühe gibt, in der Auswahl des Typen und der Qualität Ihren Bedürfnissen gerecht zu werden und nicht einfach einen bekannten Markennamen zu verkaufen
- die Bereitschaft des Verkäufers, nach Geschäftsabschluß Fragen zu beantworten und Systemhilfe zu bieten

Wenn Sie Ihren Computer für Geschäftszwecke benutzen, sollten die Disketten von höchster Qualität sein. Wenn Ihr Computer zusammenbricht und für einige Zeit ausfällt, ist es genauso, als ob Sie alle Ihre Geschäftsunterlagen im Safe eingeschlossen hätten, sich aber nicht an die Schlüsselkombination erinnern könnten: mit anderen Worten: eine Information auf Diskette ist nutzlos, wenn Sie nicht auf sie zugreifen können.

Sie müssen also den Kaufpreis eines Systems gegen die laufenden Folgekosten abwägen. Seriöse Computerhändler bemühen sich nicht nur, dem Anwender bei der Lösung von Problemen behilflich zu sein, sondern sie stellen auch sicher, daß das Problem nicht noch einmal auftritt. Versandhausfirmen mögen gute Preise für Geräte anbieten, aber wiegt das die Tatsache auf, daß Sie nachher unter Umständen ohne Hilfe dastehen?

Ein oft vernachlässigtes Element ist es, sicherzustellen, daß Ausstattung und Software Ihren Bedürfnissen entsprechen. Viele Aspekte des Computersystems werden von CP/M-80 bzw. CP/M-86 diktiert. Um CP/M effektiv einzusetzen, sollte folgende Ausstattung verfügbar sein:

- zwei Laufwerke
- 500K oder mehr Plattenspeicher
- 48K Kernspeicher
- ein Terminal mit 24 Zeilen à 80 Zeichen
- ein Drucker
- ein Modem (Anschlußmöglichkeit ans Telefon)

Wir werden jede dieser Komponenten einzeln behandeln.

Zwei Laufwerke

Sie brauchen aus verschiedenen Gründen zwei Laufwerke. Wie bereits in Kapitel 1 festgestellt, sind Disketten zerbrechlich. Sie werden leicht verbogen, sind nutzlos, wenn schmutzig, es wird damit herumgespielt, sie können verloren oder gestohlen werden; kurz, sie sind kein permanentes Speichermedium. Sie müssen also Disketten zur *Sicherstellung* kopieren, und das geht mit zwei Laufwerken wesentlich einfacher und schneller.

Natürlich kann man ein Computersystem mit nur einem Laufwerk fahren – und in der Tat werden solche Systeme angeboten – nur können Sie mit Ihrem System nicht länger arbeiten, wenn das nicht funktioniert. Bei zwei Laufwer-

ken können Sie in solchen Fällen jedoch weiterarbeiten, wenn auch mit Einschränkungen.

Es ist unbequem, mit nur einem Laufwerk zu arbeiten und das kann dazu führen, es sein zu lassen. Viele ernsthafte CP/M-kompatible Anwendungsprogramme für Geschäftszwecke erfordern zwei Laufwerke einfach deshalb, weil der Programmkomplex selbst bereits eine ganze Diskette belegt. Außerdem werden professionelle Computeranwender bestätigen, daß es eine gute Angewohnheit ist, Daten- und Programmdisketten getrennt zu halten. Das ist jedoch mit nur einem Laufwerk unmöglich.

500K Bytes Plattenspeicher

500 000 Zeichen scheinen eine Menge Information zu sein. Beispielsweise enthält der Text dieses Buches etwa 450 000 Zeichen. Üblicherweise unterschätzen jedoch Computeranwender die Menge an Information, die sie benutzen und generieren.

Ernsthafte Buchhaltungsprogramme (wie das „Structured Systems Group Integrating Accounting"- oder das „Peachtree Integrated Accounting"-Programmpaket) benötigen 700K allein zur Speicherung der Programme, obwohl diese normalerweise auf mehrere Disketten verteilt werden. Die Menge an Informationen, die ein Unternehmen für Steuern und Buchprüfung benötigt, ist schwindelerregend.

Betrachten Sie ein typisches Einzelhandelsgeschäft mit durchschnittlich 100 Transaktionen pro Tag. Mit nur 100 Zeichen pro Transaktion werden Sie im Jahr doch annähernd 4 Millionen Zeichen speichern müssen.

Es ist unrealistisch, alle benötigten Informationen auf einer einzelnen Diskette speichern zu wollen. Stattdessen sollte man die Information in **logische** Blöcke untergliedern und jeweils einen Block auf einer Diskette sichern. Dieser Vorschlag hat einen einfachen Grund. Nehmen wir an, Ihre jährlichen geschäftlichen Transaktionen erfordern die Speicherung von 4 Millionen Zeichen und Sie wollen die Daten für die zweite Woche im August einsehen. Bei vielen Systemen muß das Programm zuerst Januar bis Juli durchstöbern, bevor es die August-Daten findet und das kostet Zeit. Es wäre praktischer und bequemer, die Information eines Monats einer einzelnen Diskette zuzuordnen. Buchhaltungssysteme benötigen selten alle Daten auf einmal.

Noch ein weiterer Vorteil spricht für Floppy-Systeme mit Kapazitäten zwischen 500K und 1MB, sofern Sie Buchhaltungs-Software benutzen. Viele Buchhaltungssysteme schließen jeden Monat alle Transaktionen ab und führen nur die Bilanzen fort. Normalerweise können Sie aus einem abgeschlossenen Monat nicht mehr Einblick in die Einzelheiten jeder Transaktion nehmen. Wenn Sie jedoch Ihre Daten duplizieren, geht es. Da es wegen der vielen unterschiedlichen Bewegungen unmöglich ist, Zahlen genau wiederzugeben, reicht eine einzelne Diskette mit einer Kapazität zwischen 241K und 500K

meist bequem für die Information eines Monats, bei weniger Speicherkapazität ist das unsicher.

Generell sind 500K bis 1MB Speicherkapazität für Buchhaltungszwecke ausreichend. Dagegen würde es wahrscheinlich mehr erfordern, das Inventar eines Händlers für elektronische Teile zu führen.

Zwischen dem Erscheinen der ersten und zweiten Auflage dieses Buches hat sich eine Formation von Anbietern CP/M-gestützter Niedrigpreiscomputer mit minimaler Plattenspeicherkapazität gebildet. Heath/Zenit, Osborne, IBM und Xerox sind Beispiele dafür. Die 160K bis 320K, die diese Maschinen an Plattenspeicher vorsehen, schränken ihre Anwendbarkeit auf vielen Gebieten ein. Wenn Sie auch solche Maschinen für Textverarbeitung, Statistiken, Buchhaltungsübersichten, Budgetkalkulationen oder Personalangelegenheiten verwenden können, würden sie doch für generelle Buchführung, Inventarmanagement oder ähnliche Aufgaben, wo die Manipulation großer Datenmengen nötig ist, nicht ausreichen.

48K Kernspeicher

WordStar, das Programm, mit dem dieses Buch geschrieben wurde, belegt 44K. Dann bleiben immer noch ungefähr 4000 Zeichen, die Sie auf einmal (ohne Zugriff auf Plattenlaufwerke) abtasten können. Heutzutage erfordern viele angebotene Programme zwischen 32K und 48K, einige sogar 64K. Nahezu alle kommerziell verfügbaren Programme in CP/M-80 werden mit 56K, die meisten sogar mit 48K Kernspeicher auskommen. Für CP/M-86 sollten Sie zugunsten einer effizienten Anwendung aufgrund der internen Struktur zwischen 96K und 128K zur Verfügung haben.

Jedoch ist der Programmumfang nicht die einzige Überlegung, die bei der Bestimmung einer angemessenen Kernspeicherkapazität eine Rolle spielen sollte. Die meisten Buchhaltungs- oder Datenbankprogramme müssen die Disketteninformationen vor der Auswertung sortieren. Ein gutes Sortierprogramm benutzt den gesamten Kernspeicher; je mehr zur Verfügung steht, desto schneller läuft die Routine. Es folgt ein einfacher Vergleich von Kernspeicherbedarf und Effizienz eines Sortierungsvorgangs:

Vorgang	**Erforderliche Zeit**
Ein ineffizienter Sortierungsvorgang, der mit 16K 706 Posten über zwei Felder bearbeitet.	3 Stunden, 10 Minuten
Ein effizienter Sortierungsvorgang der mit 48K 525 Posten über fünf Felder bearbeitet.	1 Minute, 25 Sekunden

Beachten Sie, daß hier außer der Kernspeicherkapazität auch andere Punkte einbezogen sind. Dem ineffizienten Sortierungsvorgang mehr Kernspeicher zuzuordnen, würde den Prozeß beschleunigen, aber nur um eine

Stunde. Anhand dieses Beispieles lassen sich Programmpakete für Geschäftszwecke einstufen, wie sie in Computerläden verkauft werden.

Kaufen Sie mindestens 48K für Ihr System. Kaufen Sie 64K, wenn Sie es sich leisten können. Wenn Sie mit CP/M-86 arbeiten, werden Sie möglicherweise Ihren Kernspeicher auf 128K ausdehnen wollen, so daß Ihnen auch die fortgeschritteneren Anwendungsmöglichkeiten zur Verfügung stehen.

Zeichen-Terminals

Das Terminal sollte einen Video-Bildschirm haben, der mindestens 24 Zeilen mit 80 Zeichen wiedergeben kann. Diese Empfehlung beruht auf verschiedenen Überlegungen. Eines der praktischsten Anwendungsgebiete für Computersysteme ist *Textverarbeitung*: das Schreiben, Speichern und Reproduzieren von Textinformation. Das Darstellungsvermögen eines Terminalbildschirms in Bezug auf die Zeichenanzahl bestimmt, wieviel Sie von einem Dokument oder Satz auf ihm erstellen oder ausgeben können; davon hängt die Lesbarkeit ab.

Ein Programm läßt möglicherweise Raum zwischen Informationsblöcken, um sie lesbarer zu machen. Andere drängen Zeichenketten zusammen. Einige der raffinierteren Textverarbeitungsprogramme benutzen zwei Bildschirmzeilen oder sogar mehr, um den Status eines Dokuments aufzuzeigen. WordStar zum Beispiel gibt von der jeweiligen Position an Informationen in folgender Form aus:

```
   A:CPMG     PAGE 5 LINE 26  COL 35                    INSERT ON
L ---- ! ---- ! ---- ! ---- ! ---- ! ---- ! ---- ! ---- ! ---- ! ---------- R
```

Das Beispiel zeigt, daß die Datei „CPM8" von Laufwerk A, Seite 5, Zeile 26 ausgegeben wird und der Cursor gerade auf Spalte 35 steht. „INSERT ON" bedeutet, daß Sie Information in einen vorhandenen Text eingeben. Die folgende Zeile zeigt durch „L" und „R" die linke und rechte Zeilenbegrenzung und durch „!" die Tabulatorstops an. Dies ist eine wertvolle Information, aber bei einem Bildschirm von 16 Zeilen bleibt um so weniger Raum für das Dokument übrig.

Leider haben manche Software-Autoren ihre Programme auf minimale Bildschirmkapazität ausgelegt, so daß Benutzer von Standardterminals (24 x 80 Zeichen) mit solchen Programmen wertvolles Fassungsvermögen verschwenden.

Bedenken Sie, eine sauber gedruckte Seite hat 66 Zeilen mit je 102 Zeichen (Elite-Typ), in Pica 85 Zeilen. Wenn es der Hauptzweck Ihres Computersystems ist, den Papierkram zu automatisieren, werden Sie eine Wiedergabe wünschen, die das Geschriebene möglichst dicht zusammendrängt. Wenn solche Terminals auch etwas mehr kosten, wird sich ihre Bequemlichkeit durch die gesparte Zeit mehr als bezahlt machen.

Eine andere Überlegung bei der Auswahl eines Terminals ist der Typus und das Layout der Tastatur. Fast alle Terminals sind in Bezug auf die Anordnung der Tasten etwas unterschiedlich. Einige imitieren die Tastatur der IBM-Selectric, andere haben ihr eigenes Layout. Auf folgendes sollte man bei einer Tastatur achten:

- Sind die SHIFT- und CONTROL-Tasten bequem angeordnet oder hindern sie möglicherweise beim Tippen?
- Sind Cursor-Pfeile vorhanden, um die Schreibmarken des Bildschirms zu bewegen?
- Ist die RETURN-Taste (manchmal auch mit ENTER beschriftet) bequem angeordnet?
- Wie fühlt sich die Tastatur an? Benötigt man Kraft, um Zeichen einzugeben?

Wichtig ist auch die Plazierung der Tastatur im Verhältnis zum Bildschirm. Wenn das Zentrum der Tastatur gegenüber dem Zentrum des Bildschirms auch nur leicht verschoben ist, werden Sie möglicherweise feststellen, daß Sie Ihre Finger ständig auf die falschen Tasten setzen. Deshalb ist eine bewegliche Tastatur besser als eine feste, die nicht genau zentriert ist.

Drucker

Sie erinnern sich, daß wir Disketten als zerbrechlich und nur für temporäre Zwecke dienlich beschrieben haben, die gesichert werden müssen, wenn Sie nicht einen versehentlichen Informationsverlust riskieren wollen. Natürlich kann die Information auf dem CRT ausgegeben werden, aber manchmal ist ein gedrucktes Dokument einfach angenehmer. Wir empfehlen also einen Drucker aus mehreren Gründen.

Der IRS (Internal Revenue Service) betrachtet Computer-Dokumente (Disketten-Dateien) nicht immer als verbindlich. In vielen Fällen hat der Gebrauch eines Computers für Buchhaltungs- und sonstige Informationsspeicherung IRS nicht davon abgehalten, gedruckte Kopien der Information zu verlangen. Wenn die Agentur einer Computer-Buchprüfung in allen Fällen glauben würde, müßte sie eine Zusammenstellung aller Fabrikate und der gesamten existierenden Software sammeln, um die Integrität der Information zu sichern. Für die Firmenbuchhaltung gibt es Standards, für die Rechenverfahren nicht unbedingt, höchstens in großen Gesellschaften.

Benutzer zögern oft, wertvolle Informationen einem Medium anzuvertrauen, das Sie nicht lesen können. Der erste Computer-Software betreffende Copyright-Prozeß befaßte sich besonders damit, daß es nicht möglich sei, die in einem Computer, einer Diskette oder einem ROM (Read-Only Memory) enthaltene Information zu **lesen**. Das bedeutet, daß die gleiche Information, auf Papier gedruckt, einen anderen Status hat. Vielleicht wird die gegenwärtige Generation unserer Kinder, die mit Fernsehen und Computern auf-

wächst, die *offizielle Anerkennung* von Information durchsetzen, die in Form magnetischer Impulse gespeichert ist. Bis dahin werden die Grenzen von einer Gesellschaft gesetzt, die speziellen Wert auf das gedruckte Wort legt. Der smarte Computeranwender erstellt also gedruckte Dokumente zur Ergänzung der magnetischen Speicherung von Information.

Ein anderer Grund, Daten gedruckt aufzubewahren, ist der Mangel von Standards zwischen Computern. Es gibt über 50 verschiedene Formate der Diskettenspeicherung (alle mit CP/M), die die Verläßlichkeit auf magnetische Medien für alle Leser unrealistisch machen. In der Tat, selbst wo Information auf einem Bildschirm dargestellt werden kann, ist es sinnvoll, Hardcopy-Versionen der Information zu sichern. Solange Auge und Gehirn nicht in der Benutzung einer Video-Informationswiedergabe geübt sind, werden Sie gelegentlich Schwierigkeiten haben, Fehler bei der Dokumentenwiedergabe auf dem Bildschirm zu entdecken, die Sie auf einer gedruckten Kopie sofort finden würden. Kurzum, auch wenn Sie das Lesen vom Papier gewohnt sind, wird es eine Weile dauern, bis Sie eine Video-Wiedergabe genauso lesen können.

Sie brauchen den Drucker also, um

- Information für Revisionszwecke zu sichern,
- Information transportabel zu machen, und um
- Information für Menschen lesbar zu machen.

Überdies kann man Papierkopien als zusätzliche Backup-Maßnahme betrachten, besonders, wenn sie nicht am gleichen Platz wie die Disketten aufbewahrt werden.

Modems

Viele Leute halten ein Modem für einen unverzichtbaren Bestandteil ihres Computersystems. Es erlaubt einem System, mit einem anderen über Telefon zu verkehren. Modems sind in allen Preisstufen und Ausführungen erhältlich, vom einfachen akustischen Koppler bis hin zu fortgeschrittenen, von Mikroprozessoren kontrollierten Einheiten. Einige der am weitesten verbreiteten Einheiten sind das D. C. Hayes Smartmodem, das PMMI MM-103 und das Novation CAT. Ihre Ausführungen schließen variierende Datenübertragungsraten, automatische Anrufbeantwortung sowie Tasten- (für Tonwahl) bzw. Wählscheiben-Modus ein. Wenn Sie ein Modem kaufen, vergewissern Sie sich, daß es alles enthält, was Sie benötigen; kaufen Sie aber auch kein teuereres mit Möglichkeiten, die Sie nicht brauchen.

Wenn Sie vorhaben, per Telefon mit einem anderen Computersystem zu kommunizieren und dabei auch Programme, Daten und Listen senden, empfangen und sichern zu können, brauchen Sie ein Modem-Kontrollprogramm. Es wird Ihnen helfen, den effizientesten Gebrauch des Modems im Zusammenhang mit Ihrem System zu gewährleisten, und es wird Ihnen ersparen,

unnötig viele Kommandoversionen und Abfragen im Kopf zu behalten. Angenommen, Sie haben ein Smartmodem und wollen eine Nummer über Tastatur wählen. Ohne Kontrollprogramm würden Sie ATDT und dann die Nummer tippen. Mit Kontrollprogramm würden Sie ATDT und dann die Nummer tippen. Mit Kontrollprogramm käme die Abfrage „NUMBER?". Sie würden die Nummer dann eingeben, und das Programm würde dann automatisch für Sie Buchstaben und Ziffern senden. Ohne Kontrollprogramm werden Sie sich auf viele verschiedene Kommandos besinnen oder sie nachschlagen müssen. Mit dem Kontrollprogramm entfällt dieser Aspekt. Viele Programme mit automatischer Anwahl erlauben Ihnen, allgemein benutzte Telefonnummern in Ihr System einzuspeichern.

Das bedeutet also, daß durch das Drücken weniger Tasten das Programm das Modem-Kommando wie auch die Telefonnummer senden wird. Mit MCI oder Sprint (Niedrigpreis-Telefondienste für weite Entfernungen) bzw. DATEX ergibt sich daraus eine reale Zeitersparnis.

Der entscheidende Aspekt eines Kontrollprogramms ist die Sicherung von Übertragungen auf Platte oder Band. Nehmen wir an, Sie haben einen Freund, der ein großes neues Spiel geschrieben hat und Ihnen eine Kopie geben möchte. Ohne Kontrollprogramm müßten Sie das Spielprogramm manuell in Ihr System eingeben. Mit Kontrollprogramm würde es automatisch auf Platte oder Band gesichert werden. Für die Geschäftswelt werden Modems und Kontrollprogramme von Tag zu Tag wertvoller, wenn es um die Übermittlung von Verkaufsdaten, Listen und anderer Daten geht. Unter MP/M mit Mehrfach-Telefonlinien und Modems wurden Möglichkeiten der Telekommunikation für geschäftsorientierte Anwendung entwickelt, die es früher nur in großen Systemen wie IBM 370 und ähnlichen Maschinen gab. Allgemein angewandte Kontrollprogramme schließen MODEM7, SMODEM 36, Crosstalk und ZTERM ein. MODEM7 und SMODEM 36 gehören zur Software der Öffentlichen Hand, wobei SMODEM 36 für den Gebrauch mit dem Smartmodem entwickelt wurde. Crosstalk ist ein populäres Programm mit einem Nachteil – es kann Speicherdaten für ein System nur senden und empfangen, wenn auch das andere System Crosstalk verwendet. ZTERM wird heutzutage als eines der besten erhältlichen Programme angesehen. Allerdings kann es nur auf einem Apple-Computer installiert werden. Wie beim Modem sollten Sie sich auch nach einem Kontrollprogramm umsehen, das Ihren individuellen Anforderungen entspricht.

Eine andere allgemeine Anwendungsform für Modems ist der Zugriff auf zwei populäre Timesharing-Netzwerke, Source und CompuServe. Anwender können die letzten Nachrichten, Ereignisse und Börsernkurse erfahren, Fremdsprachen lernen, Spiele probieren und vieles andere mehr. Diese Einrichtung ist dem deutschen Bildschirmtext vergleichbar, jedoch zahlen Sie in den USA $4 bis $10 pro Stunde und benötigen eine Kreditkarte. Wenn Ihnen die Kosten nichts ausmachen, lohnt es sich schon, sich über solche Netzwerke zu informieren.

In den USA sind bereits viele freie „Schwarze Bretter" für Computer verbreitet. Ihre Teilnehmer übermitteln sich gegenseitig Briefe, Verkaufsraten, gewünschte Daten, Termine und andere sachbezogene Neuigkeiten. Auch Software wird häufig kostenlos weitergegeben, so daß nur die Telefonkosten bleiben.

Wenn sie garnichts davon brauchen können, wäre ein Modem reine Geldverschwendung. Wenn Ihnen jedoch die Idee gefällt, mit anderen Systemen zu kommunizieren, ist ein Modem eine gute Investition.

Empfehlungen

Es folgen einige Empfehlungen zum klugen und effizienten Gebrauch von CP/M.

Analysieren Sie, was Sie von Ihrem Computer wollen (beachten Sie auch die Sektion über Systemdesign am Ende dieses Kapitels).

Lesen Sie alle Handbücher, auch wenn Sie sie zu diesem Zeitpunkt nicht immer verstehen. Wenn Sie Ihren Fähigkeiten im Gebrauch des Computers noch nicht ganz vertrauen, erbitten Sie die Hilfe eines Mikrocomputer-Experten, wie z. B. Ihres Computerhändlers.

Ermitteln Sie, wieviele Disketten Sie brauchen werden.

Berücksichtigen Sie dabei Original- und Backup-Disketten, und formatisieren Sie diese mit Ihrer Version des CP/M.

Erstellen Sie brauchbare Kopien aller Programm-Disketten.

Benutzen Sie nie die Orginal-Diskette. Das Original (incl. CP/M-Diskette) sollte an einem sicheren Platz wie einem Safe, Banksafe oder abgeschlossenen Dateienkabinett gelagert werden. Kennzeichnen Sie alle Disketten, sowie sie erstellt oder modifiziert werden. Datieren Sie das Etikett und vermerken Sie das Datum gesondert.

Kopieren Sie CP/M-80 mit dem Programm SYSGEN auf jede Diskette, die Sie zusammen mit Ihrem System gebrauchen werden (bei CP/M-86 kopieren Sie mit LDCOPY die Information für den CP/M-86-Lader auf die Systemspuren einer jeden Diskette, danach kopieren Sie CPM.SYS mittels PIP).

Kopieren Sie auf alle Disketten das gleiche CP/M; vermischen Sie die Versionen nicht.

Richten Sie eine vernünftige Backup-Prozedur ein und halten Sie sich daran.

Folgendes wird für geschäftsorientierte Anwendung empfohlen: Verlassen Sie sich nicht darauf, daß der Computer sich an alles erinnert. Beseitigen Sie nicht das laufende System, das Sie benutzen, um Informationen festzuhalten. Bewahren Sie alle Informationen auf, die der Computer ausdruckt. Legen Sie sie ab, wie Sie es bei anderer Information auch tun würden. Betrachten Sie einen Computer so, wie einen Ihrer Angestellten. Wenn Sie es für wichtig genug halten, für die Krankenversicherung Ihrer Angestellten zu zahlen, dann

tun Sie es auch für Ihren Computer; ein Servicevertrag sollte die maximale Dauer garantieren, die Sie ohne ihn auskommen müssen.

Wenn Sie den Computer anstelle eines laufenden Buchhaltungs- oder vergleichbaren Systems einsetzen, werfen Sie das alte nicht weg. Wenn der Computer Textverarbeitung vorsieht, verkaufen Sie nicht alle Ihre Schreibmaschinen. Entlassen Sie auch nicht alle Angestellten, die mit dem alten System vertraut waren. Gelegentlich wird Ihr Computer Probleme haben, und Sie sollten dafür gerüstet sein.

Computer dienen der Steigerung der Produktivität; sie sind keine Ersatzangestellten. Benutzen Sie ihn, um Ihr Operationsfeld zuerweitern und dessen Effizienz zu steigern, nicht um Menschen zu ersetzen. Geschäft oder Personaleinsatz effizienter zu gestalten, mag Ihren Personalbedarf senken, aber das sollte nicht Ihr Motiv sein, auch sollten Sie die politischen Folgeerscheinungen nicht außer Acht lassen.

Vermeiden Sie die „Computerisierung".

Warum ein Kommando wie PIP benutzen, wenn einfach COPY gemeint ist? Benutzen Sie das Kommando REN, um sinnvolle Programm-Namen zu bilden. Hier einige Vorschläge:

Ändern Sie	PIP	in COPY
	CRUN 37	in RUN
	ED	in EDIT
	CBAS2	in COMPILE
	ASM	in ASSEMBLE
	MOVCPM	in MOVE-CPM
	WS	in WORDSTAR

Auch eine Anzahl anderer Programme kann in dieser Weise geändert werden, so daß jeder Anwender den Zweck einer Plattendatei sofort erkennt und nicht durch unverständliche Abkürzungen verwirrt wird. Bitten Sie z. B. auch Ihren Computerhändler, eine Meldung wie

BDOS ERR ON X: in DISK ERR ON X:

zu ändern oder Ihnen den Änderungsvorgang zu zeigen.

Auch andere kleine Änderungen können den Computerjargon etwas mildern, der für Anfänger manchmal verwirrend ist. Folgende Änderungen an CP/M könnten beispielsweise sinnvoll sein:

- Für Disketten, die bei jedem CP/M-Start (oder Neustart) die automatische Ausführung einer Reihe von Instruktionen vorsehen, könnte folgende Meldung einbezogen sein:

  AUTOMATIC EXECUTION OF program name OCCURRING

  Da einige Programme längere Zeit zum Laden brauchen, aber dies nicht anzeigen, setzt manchmal Panik ein. „Was geht da vor?" ist ein häufiger

Ausruf an diesem Punkt, und er ist gewöhnlich von dem starken Wunsch begleitet, den RESET-Knopf für einen neuen Start zu drücken. Eine entsprechende Meldung könnte die Verwirrung mildern.

- Programme ohne angemessene Techniken zur Fehlerbehandlung können dadurch verbessert werden, daß sie dem Programmierer anzeigen, daß der Fehler erkannt wurde, und eine Möglichkeit vorsehen, den Vorgang erneut zu probieren statt das Programm abzubrechen und nach CP/M zurückzukehren. Fehlermeldungen sollten ausgedruckt werden, so daß sie nicht übersehen werden können. Sollten Sie zur Interpretation von Fehlermeldungen ein Handbuch benötigen, so zeigt dies eine Programmschwäche.
- Bilden Sie jeden aus, der mit dem Computer arbeiten soll. Ein Teil dieser Anforderung wird durch dieses Buch erfüllt. Wählen Sie einen Händler, der Sie nicht im Stich läßt, sondern Training und Ausbildung miteinschließt. Erlernen Sie die Computerprozeduren selbst. Es ist nicht klug, nur einen Angestellten zu haben, der „das System fahren" kann.

Wenn Sie diese Vorschläge einbeziehen, stellen Sie sicher, daß Sie sie mit den Entsprechungen in Ihren System-Handbüchern in Einklang setzen, sonst sind Sie vielleicht nachher verunsicherter als je zuvor.

Die Mikrocomputer-Industrie ist jung und befindet sich im Wettstreit. Teilen Sie Ihre Probleme mit anderen. Es gibt eine Reihe von Veröffentlichungen, die sich selbst als Anwalt des Anwenders anpreisen. Dazu gehören **Info World, Kilobaud Microcomputing, Creative Computing** und **Interface Age** (in Deutschland andere). Lassen Sie andere Ihre Probleme und Frustrationen wissen.

Die Systemkriterien

Das hier Gesagte bezieht sich nicht direkt auf CP/M, kann aber Ihre Fähigkeit erhöhen, es einzusetzen. Viele Computerbenutzer sollten darauf besser verzichten, Computer zu benutzen. Unsere Gesellschaft glaubt häufig, jedem Problem mit neuerer und besserer Technologie begegnen zu können. Das ist einfach nicht wahr. Um sicherzustellen, daß Sie nicht in die Technologiefalle stolpern, *halten Sie inne* und analysieren Sie Ihren gegenwärtigen Gebrauch des Computers. Prüfen Sie, ob Sie die folgenden Fragen beantworten können:

- Haben Sie den Computer gekauft, weil jemand Ihnen versprach, er werde bestimmte Probleme lösen? Haben Sie diesem Versprechen geglaubt?
- Haben Sie den Computer gekauft, weil Sie von Technologie fasziniert sind?
- Können Sie die Aufgabenstellung des Computers genausogut von Hand erledigen?
- Wird der Computer weniger als eine Stunde pro Tag benötigt?
- Nimmt der Computer gemessen an den Resultaten zuviel Zeit in Anspruch?

– Würden Sie den gleichen Handel noch einmal tätigen?

Wenn Sie irgendeine dieser Fragen mit „ja" beantworten mußten, besteht eine Wahrscheinlichkeit, daß Sie noch kein geeignetes System für die Anwendung Ihres Computers entwickelt haben. Das Schlüsselwort ist hier „System". „System" schließt ein, daß der Computer in einer systematischen, logischen Weise benutzt wird.

Betrachten Sie dieses Beispiel und klären Sie, was unter „Systemkriterien" zu verstehen ist. Angenommen, Sie arbeiten für ein Versandhaus für Tennis-zubehör. Sie wollen die Bearbeitung von Bestellungen, Inventar sowie periodische Listen von Post- und Telefonbestellungen für Ihren Chef auf Computer legen. Dafür können Sie einen Computer und Software kaufen, aber ist das alles, was Sie brauchen? Definitiv: nein.

Ihre Firma empfängt Bestellungen per Post und gelegentlich per Telefon. Ihr Computersystem muß bis ins Detail erkennen, wie der Computer Post- und Telefoninformation annimmt (bedenken Sie, daß in den meisten Fällen Post nur einmal täglich auftritt, während das Telefon jederzeit läuten kann). Außerdem erhalten Sie Rechnungen, Verkaufsbelege, Versandbelege und Preisänderungen von Großhandelsunternehmen. Der Computer muß auch das verarbeiten können. Tatsächlich müssen dem Computer die Lager-Daten zur Verfügung stehen, bevor er für den Warenverkauf eingesetzt werden kann. Und wie erfaßt der Computer Reklamationen, Anfragen über den Lieferungsstand, Bestellungsrücknahmen usw.?

Angenommen, Ihre Firma beschäftigt außer dem Geschäftsführer zehn Leute. Brauchen alle zehn den Computer? Hat jedes Firmenmitglied Zugang zu allen darin enthaltenen Informationen? **Braucht** jedes Familienmitglied Zugang zu allen gespeicherten Informationen?

In der Praxis ist ein Computer generell ein kleiner Teil in einem Rundumsystem. Geschriebene Belege existieren immer noch und werden von Person zu Person übergeben. Im Falle des Einzelhandels läßt sich der Verkauf gar nicht wirklich computerisieren; alles muß gelagert, identifiziert und gesondert behandelt werden.

Um sicherzustellen, daß der Computer den anderen Systemkomponenten entspricht, müssen Sie Ihre Konstellation Ihrem Computer- und Softwarehändler nahebringen. Wenn mehr als eine Firma einbezogen ist, veranlassen Sie eine gemeinsame Diskussion. Stellen Sie klar, daß Sie eine komplette Integration des Computers in Ihrem Geschäftsablauf wünschen, so daß jeder Beteiligte dies einsieht. Verhandeln Sie nur mit Leuten, die dieser Anforderung entsprechen. Bedenken Sie, daß ein Computer-Händler, der alles über Wartestatus, NAND-Gitter und Unterbrechungsvektoren weiß, aber nichts über Zahlungsverkehr, Bestandsbewertung und Transportabwicklung, für ein Geschäft wie das Ihre unbrauchbar ist.

Solche Diskussionen werden Sie darüber informieren, welches Computer-

zubehör und welche Software Sie benötigen und auch darüber, wie der Computer in Ihren Arbeitsablauf integriert werden kann. Es wird möglicherweise hilfreich sein, von diesem Arbeitsablauf ein Flowchart (= Datenfluß-Diagramm) zu erstellen, um zu erkennen, wie Ihr Geschäft wirklich abläuft (den Weg jedes physischen Bestandteils in Ihrem Geschäft durch alle Instanzen aufzuzeigen).

Wenn Sie davon überzeugt sind, daß Computer und Software für Ihr System und nicht dagegen arbeiten werden, schreiten Sie im Einkaufsverfahren fort. Sehen Sie sich an, wie Zubehör und Software im Zusammenhang mit Ihrer Umgebung funktionieren werden. Ein Computerladen wird einen Computer für einen Teil der eigenen Geschäftsabwicklung einsetzen. Wenn nicht, interessiert man sich dort nicht für die Belange des Benutzers; es handelt sich dann meist um Liebhaber, die mehr an Computerspielen interessiert sind.

Schulen Sie Ihr Personal und bereiten Sie den Geschäftsablauf auf den Einsatz des Computers vor, bevor er eintrifft, nicht hinterher. Wo wollen Sie ihn aufstellen? Braucht er eine spezielle Umgebung? Muß jeder den Umgang mit ihm lernen? Viele solcher Fragen wollen erwogen sein, **bevor** der Computer eintrifft und Sie damit umzugehen versuchen.

Was ist mit Ihrem alten System? Wenn Sie derzeitig alle Bestellungs-, Lager- und sonstigen Belege per Hand bearbeiten, wollen Sie aufhören, sie auszufüllen? Es gibt verschiedene Wege, ein neues Computersystem einzuführen, insbesondere:

Cold turkey (= Sprung ins kalte Wasser)
Eines Tages stoppen Sie das alte System und starten das neue.

Phased in (= phasenweise)
Sie untergliedern jede Aufgabe in Ihre Komponenten und konvertieren diese eine nach der anderen. Wenn ein Teil des neuen Systems anstandslos arbeitet, konvertieren Sie den nächsten Teil. Es handelt sich um eine Serie von kleinen „cold turkeys".

Parallel
Sie benutzen Ihr altes und das neue System gleichzeitig. Sie vergleichen Stück für Stück die Ergebnisse, um sich zu versichern, daß das neue System die Aufgabe korrekt und vollständig durchführt. Sie können Ihren Angestellten leicht die Unterschiede zwischen den Systemen aufzeigen.

Nun vergleichen Sie den Gebrauch Ihres Computers mit der Zielprojektion. Ist er ein Teil des Systems? Fügt er sich in die sonstigen Komponenten ein oder ist er von den anderen Prozeduren isoliert?

Lassen Sie uns die Lehren dieses Abschnitts zusammenfassen:

- Ein Computer ist ein Werkzeug.
- Ein Computer ist eine komplette Lösung für irgendein Problem.
- Sie kontrollieren den Computer, nicht umgekehrt.

Ein Computer ist keine unbedingt Notwendigkeit. Schätzen Sie ab, inwieweit ein Computer Ihren Geschäftsanforderungen entspricht.

Was nun?

Es ist erstaunlich, daß ein Betriebssystem, das nur eine Handvoll eingebauter Kommandos kennt, über 200 Seiten braucht, um in seiner Fülle beschrieben zu werden. In gewisser Hinsicht zeigt dies die Komplexität der Probleme auf, mit der sich die Anwender von Computer-Software immer noch beschäftigen müssen.

Andererseits hoffen wir, daß dieses Buch mehr Information präsentiert, als Sie je über CP/M-80 oder CP/M-86 brauchen werden. Wenn Sie also Ihren Computer mit CP/M fahren und mehr darüber wissen wollen, wird Ihnen das u. U. mehrmalige Lesen dieses Buches wahrscheinlich genügend Hintergrund geben, um den Details zu begegnen, die der tägliche Gebrauch von CP/M einschließt. Außerdem sollte der Aufbau dieses Buches klar genug sein, um auf besondere Fragen sofort die richtige Antwort zu finden.

Die Antwort für die meisten CP/M-Anwender auf die Frage „Was nun?" lautet also: Benutzen Sie den Computer und halten Sie dieses Buch in der Nähe bereit.

Wenn Sie nur neugierig ein Computer-Hobbyist oder vielleicht sogar ein „Computerprofessional" sind, sollten Sie sich damit begnügen, dieses Buch zu Ende gelesen zu haben. Benutzen Sie auch die diesem Buch angefügte Bibliographie, um weitere interessante Informationen zu finden. Stellen Sie das Kapitel über die technischen Aspekte von CP/M-80 und CP/M-86 an den Beginn Ihrer Experimente. Fragen Sie Ihren Computer-Händler, ob in Ihrer Umgebung Anwendergruppen existieren; wenn ja, treten Sie einer bei. Wenn nein, überlegen Sie, ob Sie eine gründen wollen. Kurzum, betrachten Sie dieses Buch nicht als das letzte Wort über CP/M – es ist es nicht.

Anhang A
CP/M – Zusammenfassung der Kommandos

Dieser Anhang faßt das Format von Kommandozeilen und die Funktion eines jeden eingebauten sowie transienten Kommandos in CP/M-80 und CP/M-86 zusammen. Die Kommandos sind alphabetisch aufgelistet.

Tragen Sie in den auf dieser Seite freigelassenen Platz das Format der Kommandozeile für Ihr Platten-Kopier- und Platten-Formatierungs-Programm (Initialisierung) ein. Nähere Erläuterungen siehe Kap. 5.

ASM – Kommando-Zeilen

ASM filename <cr>

Assembliere die Datei „filename.ASM"; benutze für alle Dateien das derzeitig aktive Standard-Laufwerk.

ASM filename.opt <cr>

Assembliere die Datei „filename.ASM" auf Laufwerk o: (A:, B:, . . ., P:). Schreibe die HEX-Datei auf Laufwerk p: (A:, B:, . . ., P:) oder unterlasse dieses, wenn p: als Z: angegeben ist.

Schreibe die PRN-Datei auf Laufwerk t: (A:, B:, . . ., P:), gib sie auf Konsole aus, wenn p: als X: angegeben ist oder unterlasse die Ausgabe, wenn dafür Z: steht.

ASM-86 – Kommando-Zeilen

ASM-86 filename.typ $options

Assembliere die Datei „filename.A86" (oder „filename.typ", wenn ein Typ angegeben ist) mit den spezifizierten Optionen:

Ad Ursprungs-Laufwerk (filename.A86)
Hd Hex-Code-Laufwerk (filename.H86)
Pd Druck-Laufwerk (filename.LST)
Sd Symboltabellen-Laufwerk (filename.SYM)
Fx Format einer Hex-Datei (D = Digital Research, I = Label)

DDT oder DDT-86 – Kommando-Zeilen

DDT <cr> oder DDT-86 <cr>

Lädt DDT und wartet auf dessen Kommandos.

DDT: filename.typ <cr> oder DDT-86:filename.typ <cr>

Lädt DDT in den Kernspeicher und auch „filename.typ" von Laufwerk x: zur Prüfung, Modifikation oder Ausführung.

Zusammenfassung der DDT-Kommandos

Assss

Füge von der hexdezimalen Adresse ssss an Statements in Assembler-Sprache ein.

Bssss,ffff,cccc

Vergleiche Kernspeicher-Blöcke (DDT-86).

D

Zeige den Inhalt der nächsten 192 Bytes im Kernspeicher an.

Dssss,ffff

Zeige den Kernspeicher-Inhalt von ssss bis ffff an.

Efilename.typ

Lade Programm zur Ausführung (DDT-86).

Fssss,ffff,cc

Fülle den Kernspeicher von Adresse ssss bis ffff mit der hexadezimalen 8-bit-Konstante cc.

Fwssss,ffff,cccc

Fülle den Kernspeicher mit der hexadezimalen 16-bit-Konstante (DDT-86).

G

Beginne die Ausführung von der im Programmzähler angegebenen Adresse an.

G,bbbb

Setze einen Checkpoint an der hexadezimalen Adresse bbb, beginne danach mit der Ausführung von der im Programmzähler enthaltenen Adresse an.

G,bbbb,cccc

Setze Checkpoints (= Prüfpunkte) an den hexadezimalen Adressen bbbb und cccc, danach beginne mit der Ausführung von der im Programmzähler enthaltenen Adresse an.

Gssss

Beginne mit der Ausführung ab ssss.

Gssss,bbbb

Setze einen Checkpoint (= Programmunterbrechung) auf bbbb und beginne mit der Ausführung ab ssss.

Hx,Y

Hexadezimale Summe und Differenz von x und y.

Ifilename.typ

Richte den Standard-FCB (file control block) mit dem Namen „filename.typ" ein.

L

Liste die nächsten elf Zeilen eines aus dem Kernspeicher reassemblierten Programmes.

Lssss

Liste die nächsten elf Zeilen eines aus dem Kernspeicher reassemblierten Programmes von ssss an.

Lssss,ffff

Liste die nächsten elf Zeilen eines aus dem Kernspeicher reassemblierten Programmes von ssss bis ffff.

Mssss,ffff,dddd

Übertrage den Inhalt des Kernspeicher-Blocks von ssss bis ffff nach dddd.

R

Lies eine Datei von Platte auf Kernspeicher (zuerst Kommando „I" benutzen).

Rnnnn

Lies eine Datei von Platte um nnnn Stellen versetzt in den Kernspeicher (zuerst Kommando „I" benutzen).

Rfilename (DDT-86)

Lies eine Datei zu Testzwecken in den Kernspeicher.

Sssss

Zeige den Kernspeicher-Inhalt einer hexadezimalen Adresse ssss an und verändere ihn bei Bedarf.

Tnnnn

Trace (= schrittweise Anzeige) der Ausführung von nnnn Programm-Instruktionen.

TSnnn

Trace (= schrittweise Anzeige) und Aufführung aller Register (DDT-86).

Unnnn

Führe nnnn (hexadezimal) Programm-Instruktionen aus, danach setze einen Stop und zeige die Registerinhalte der CPU an.

USnnn

Führe nnnn Programm-Instruktionen aus und zeige danach alle Registerinhalte an (DDT-86).

V

Zeige das Kernspeicher-Layout nach dem Lesen einer Platte (DDT-86)

Wfilename.typ,ssss,ffff

Schreibe den Kernspeicher-Inhalt von ssss bis ffff nach „filename.typ" (DDT-86).

X

Zeige die Registerinhalte der CPU an.

Xr

Zeige die Registerinhalte an, setze ein Kennzeichen (Flag) in Register r und ändere den Inhalt wahlweise.

DIR – Kommando-Zeilen

DIR x:<cr>

Listet das Inhaltsverzeichnis mit allen Dateien von Laufwerk x:. Bei Nichtangabe von x: wird das derzeitig aktive Standard-Laufwerk angenommen.

DIRx:filename.typ<cr>

Listet das Inhaltsverzeichnis mit allen Dateien von Laufwerk x:, die dem angegebenen Dateinamen und -Typ entsprechen (mehrdeutige Dateinamen (Wildcard-Referenzen sind möglich)). Wenn x: nicht angegeben ist, wird das derzeitige Standard-Laufwerk angenommen.

DIRS (CP/M-86)

Zeigt die Namen der System-Dateien an. Ansonsten funktioniert es wie DIR.

DUMP – Kommandozeile (CP/M-80)

DUMP x:filename.typ<cr>

Listet die hexadezimale Repräsentation jedes Bytes der Datei „filename.typ" auf Laufwerk x:. Wenn der Dateiname mehrdeutig ist, wird die erste passende Datei gelistet.

ED – Kommandozeile

ED:filename,typ<cr>

Ruft den Editor auf, der dann die angegebene Datei auf Laufwerk x: sucht und für den ausgegebenen Text eine temporäre Datei „x:filename.$$$" erstellt. Die Datei-Angabe darf nicht mehrdeutig sein, und x: ist wahlfrei, ansonsten wird das Standard-Laufwerk angenommen.

ED – Zusammenfassung der Kommandos

Anmerkung: Dem Kommando „Z" folgen u. U. nicht-alphabetische Kommandos.

nA

Hänge Zeilen an. Überträgt n Zeilen von der Original-Datei zum Ausgabe-Puffer.
0A überträgt Zeilen, bis der Puffer mindestens zur Hälfte voll ist.

+/−B

Begin/Bottom (Beginn/Ende). Bewegt CP (Cursor-Pointer).
+B bewegt CP an den Anfang des Edit-Uffers.
−B bewegt CP an das Ende des Edit-Puffers.

+/−nC

Bewegt CP um n Stellen.
+ vorwärts
− rückwärts

+/−nD

Vernichtet n Zeichen vor oder nach dem Anzeiger CP im Edit-Puffer.
+ vor
− nach

E

Ende. Schließt die Ausgabe und die Dateien ab und übergibt die Steuerung an CP/M; normales Ende.

nFstring^Z

Finde Zeichenkette. Beginne die Suche nach dem nten Auftreten hinter CP.

H

Gehe zum Anfang der ausgegebenen Datei. Beendet ein Edit, benennt die Datei neu und gibt dann eine frühere temporäre Datei aus.

I<cr>

Insert-Modus (= Einfügung). Der Text gelangt von der Tastatur in den Edit-Puffer (hinter CP); Abschluß durch CONTROL-Z.

Istring^Z

Füge Zeichenkette nach CP in den Edit-Puffer ein.

Istring<cr>

Fügt Zeichenkette und CRLF nach CP in den Edit-Puffer ein.

nJfindstring^Zinsertstring^Zendstring^Z

Gegenüberstellung. Nachdem „findstring" hinter CP gefunden wurde, wird „insertstring" eingefügt und danach alle folgenden Zeichen bis, aber nicht einschließlich „endstring" gelöscht; wird n-mal ausgeführt.

+/−nK

Vernichtet n Zeilen.
+ vor CP
− nach CP

+/−nL

Bewegt CP vom Anfang der Zeile um n Zeilen vorwärts oder zurück.
+ vorwärts
− rückwärts

nMcommandstring^Z

Makro-Kommando. Wiederholt die Ausführung des ED-Kommandos in der durch „commandstring" angegebenen Form nmal. Bei n = 0 oder n = 1 oder nicht angegebenen n wird die Ausführung wiederholt, bis eine Fehlermeldung auftritt.

nNstring^Z

Findet Zeichenkette durch Autoscan (= Selbstüberprüfung). Findet das n.te Auftreten der Zeichenkette automatisch von der Original-Datei ausgehend und schreibt es, wenn nötig, in eine temporäre Datei.

O

Rückkehr zur Original-Datei. Leert den Edit-Puffer und die temporäre Datei, kehrt zum Anfang der Original-Datei zurück und ignoriert vorherige ED-Kommandos.

+/−nP

Bewegt CP und druckt seitenweise. Bewegt CP um jeweils eine Seite vorwärts oder rückwärts, danach wird die auf CP folgende Seite angezeigt. „nP" gibt n Seiten aus und bewirkt nach jeder Seite einen Stop.

Q

Quit Edit (= beende die Ausgabe). Löscht temporäre Dateien und blockiert mögliche Datei-Bewegungen, danach übergibt es die Kontrolle an CP/M; die Original-Datei wird nicht verändert.

R<cr>

Lies die Blockbewegungsdatei. Kopiert die vollständige Blockbewegungsdatei X$$$$$$$.LIB von Platte und fügt sie nach CP in den Edit-Puffer ein.

Rfilename<cr>

Lies Bibliotheksdatei. Kopiert die gesamte Datei „filename.LIB" von Platte und fügt sie nach CP in den Edit-Puffer ein.

nSfindstring^Zreplacestring^Z

Unterstütze Zeichenkette. Von CP an wird der Vorgang nmal wiederholt: „findstring" wird gefunden und durch „replacestring" ersetzt.

+/−nT

Tippe n Zeilen.
+ gibt n Zeilen nach CP aus
− gibt n Zeilen vor CP aus.
Wenn CP nicht an einem Zeilenanfang steht, bewirkt
0T die Anzeige vom Zeilenanfang bis CP
T die Anzeige von CP bis zum Ende der Zeile
OTT die Anzeige der gesamten Zeile ohne Bewegung von CP.

+/−U

Übersetzung in Großbuchstaben. Bei „+" findet die Übersetzung statt, bei „−" nicht.

0V

Edit-Puffer: freier Platz/Gesamtumfang; ausgegeben als Dezimalzahl.

+/−V

Verifiziere Zeilennumerierung. Nach „+V" wird für jede Zeile eine Nummer ausgegeben; diese wird dann als ED-Anforderung für die von CP angezeigte Zeile vorangestellt. Bei „−V" findet dieser Prozeß nicht statt, und die Anforderung von ED lautet „*".

nW

Schreibt n Zeilen vom Edit-Puffer in die temporäre Datei und löscht sie dann im Edit-Puffer.

nX

Block-Übertragung (Xfer). Kopiert n Zeilen hinter CP nach X$$$$$$$.LIB und erweitert somit den Inhalt dieser Datei.

nZ

Schlaf. Zögert die Exekution des folgenden Kommandos um die von n angegebene Zeitspanne hinaus.

n:

Bewegt CP an den Anfang von Zeile n (s. „+/−V").

:m

Fahre fort bis zur Zeile Nummer m. Ein Präfix für das Ende des folgenden Kommandos. Der Anfang wird durch CP angezeigt (s. „+/−V").

+/−n

Bewege CP und gib eine Zeile aus. Abgekürzte Form von +/−nLT.

ERA – Kommando-Zeilen

ERAx:filename.typ<cr>

Lösche die Datei „filename.typ" in Laufwerk x:. Dateiname und -Typ können mehrdeutig sein. Laufwerk x: wird wahlweise angegeben, sonst gilt das z. Zt. aktive Standard-Laufwerk.

GENCMD – Kommandozeile (CP/M-86)

GENCMD filename options

Konvertiere eine hexadezimale Objekt-Datei (filename.H86) in ein ausführbares Programm (filename.CMD). Die Optionen schließen das Modell 8080 und das Setzen der Segment-Register CODE, DATA, STACK und EXTRA ein.

Zeilenausgabe-Kommandos

CONTROL-C

Startet CP/M neu, wenn es das erste Zeichen in einer Kommandozeile ist. Sogenannter **Warmstart**.

CONTROL-E

Rückt zum Beginn der nächsten Zeile vor. Wird für die Eingabe langer Kommandos gebraucht.

CONTROL-H oder BACKSPACE

Löscht ein Zeichen vom Bildschirm (Version 2.0 und neuere).

CONTROL-J oder LINEFEED

Genau wie CARRIAGE RETURN (Version 2.0 und neuere).

CONTROL-M

Wie CARRIAGE RETURN (<cr>).

CONTROL-P

Schaltet die List-Einheit (normalerweise den Drucker) ein oder aus.

CONTROL-R

Wiederholt die laufende Kommandozeile (besonders sinnvoll für Version 1.4; verifiziert nach Löschung von Zeichen die Korrektur (Version 1.4 und neuere).

CONTROL-S

Stoppt zeitweilig die Datenwiedergabe auf der Konsole. Sie wird durch das Drücken irgendeiner Taste fortgesetzt.

CONTROL-U oder CONTROL-X

Löscht die laufende Kommandozeile (Version 1.4 und neuere).

RUBOUT (RUB) oder DELETE (DEL)

Löscht ein Zeichen und zeigt es verdoppelt an.

LOAD – Kommandozeile (CP/M-80)

LOAD x:filename <cr>

Liest die Datei filename.HEX von Laufwerk x: und erstellt das ausführbare Programm filename.COM auf dem selben Laufwerk.

MOVCPM – Kommandozeile (CP/M-80)

MOVCPM <cr>

Bereitet eine neue Kopie des CP/M unter Ausnutzung des gesamten Kernspeichers; übergibt die Kontrolle an das neue CP/M, sichert es aber nicht auf Platte.

MOVCPM nn <cr>

Bereitet eine neue Kopie des CP/M unter Ausnutzung von nn K; übergibt die Kontrolle an das neue CP/M, sichert es aber nicht auf Platte.

*MOVCPM ** <cr>*

Bereitet eine neue Kopie des CP/M unter Ausnutzung des gesamten Kernspeichers, die mit SYSGEN oder SAVE gesichert werden kann.

*MOVCPM nn * <cr>*

Bereitet eine neue Kopie des CP/M unter Ausnutzung von nn K, die mit SYSGEN oder SAVE gesichert werden kann.
nn ist eine dezimale Ganzzahl, die bei den Versionen 1.3 und 1.4 zwischen 16 und 64, von der Version 2.0 an zwischen 20 und 64 rangiert.

PIP – Kommandozeilen

PIP <cr>

Lädt PIP in den Kernspeicher. PIP fordert Kommandos an, führt sie aus und fordert neue an.

PIP pipcommandline <cr>

Lädt PIP in den Kernspeicher. PIP führt das Kommando „pipcommandline" aus und kehrt dann nach CP/M zurück.

PIP – Zusammenfassung der Kommandos

x:new.typ = y:old.typ[p] < cr >

Kopiert die Datei „old.typ" von Laufwerk y: auf die Datei „new.typ" auf Laufwerk x: unter Beachtung der Parameter „p".

x:new.typ = y:old1.typ[p],z:old2.typ[q] < cr >

Erstellt eine Datei „new.typ" auf Laufwerk x:, die aus den Inhalten der Datei „old1.typ" auf Laufwerk y: mit den Parametern „p" gefolgt von den Inhalten der Datei „old2.typ" auf Laufwerk z: mit den Parametern „q" besteht.

x:filename.typ = dev:[p] < cr >

Kopiert Daten von der Einheit „dev:" in die Datei „filename.typ" auf Laufwerk x:.

dev: = x:filename.typ[p] < cr >

Kopiert Daten von der Datei „filename.typ" in Laufwerk x: auf die Einheit „dev:".

dst: = src:[p] < cr >

Kopiert Daten von der Einheit „src:" auf die Einheit „dst:".

PIP – Zusammenfassung der Parameter

B	Spezifiziert Übertragungen im Blockmodus.
Dn	Löscht alle Zeichen nach der nten Spalte.
E	Echot Kopien während der Durchführung auf Konsole.
F	Entfernt während der Übertragung Formularvorschub-Zeichen.
Gn	Weist PIP an, eine Datei vom Benutzerbereich n zu kopieren.
H	Prüft auf Intel-Hex-Dateiformat.
I	Ignoriert :00-Sätze bei der Übertragung von Intel-Hex-Dateien.
L	Übersetzt Groß- in Kleinbuchstaben.
N	Addiert eine Zeilennummer zu jeder übertragenen Zeile.
O	Übertragung von Objekt-Dateien (ignoriert EOF [Dateiende-Merker]).
Pn	Bewirkt einen Seitenvorschub nach jeweils n Zeilen.
Qa^Z	Spezifiziert das Ende eines Kopiervorgangs nach Auftreten der Zeichenkette „s".
R	Weist PIP an, von einer Systemdatei zu kopieren.
Ss^Z	Spezifiziert den Beginn des Kopiervorgangs nach Auftreten der Zeichenkette „s".
Tn	Setzt einen Tabulatorstop nach jeder nten Spalte.

U	Übersetzt Klein- in Großbuchstaben.
V	Verifiziert Kopien nach Beendigung durch Vergleich.
W	Weist PIP an, auf eine R/O-Datei zu kopieren.
Z	Setzt das Prüfbit bei ASCII-Zeichen auf 0.

PIP – Empfangseinheiten

CON:	PUN:	LST:	Logische Einheiten
TTY:	PIP:	LPT:	
CRT:	UP1:	UL1:	
UC1:	UP2:		Physische Einheiten
OUT:	PRN:		Spezielle PIP-Einheiten

PIP – Sendeeinheiten

CON:	RDR:		Logische Einheiten
TTY:	PTR:		
CRT:	UR1:		
UC1:	UR2:		Physische Einheiten
NUL:	EOF:	INP:	Spezielle PIP-Einheiten

REN – Kommandozeile

REN newname.typ = oldname.tpy < cr >

Finde die Datei „oldname.typ" und benenne sie um in „newname.typ".

SAVE – Kommandozeile (CP/M-80)

SAVE nnn x:filename.typ < cr >

Sichert einen Teil der Transient Program Area (= Bereich für transiente Programme), wobei nnn die Anzahl von Kernspeicher-Pages (je 256 Bytes) als Dezimalzahl angibt. Die Laufwerksangabe ist wahlfrei.

STAT – Kommandozeilen

STAT < cr >

Zeigt Attribute und freien Speicherplatz aller Diskettenlaufwerke an, auf die seit dem letzten Warm- oder Kaltstart zugegriffen wurde.

STAT x: < cr >

Zeigt den Freiraum auf der Diskette im Laufwerk x: an.

STAT x:filename.typ < cr > (CP/M 2.0 und neuere)

Zeigt Umfang und Attribute der Datei(en) „filename.typ" auf Laufwerk x: an. Der Dateiname kann mehrdeutig sein, x: ist wahlfrei.

STAT x:filename.typ $atr<cr>

Ordnet die Attribute „atr" „filename.typ" auf Laufwerk x: zu.

STAT DEV:<cr>

Listet die den vier logischen derzeit zugeordneten physischen Einheiten.

STAT VAL:<cr>

Listet die möglichen Datei-Zuordnungen und einen partiellen Überblick über mögliche STAT-Kommandozeilen.

STAT log:=phy:<cr>

Ordnet die physische Einheit „phy:" der logischen Einheit „log:" zu (pro Zeile ist mehr als eine Zuordnung möglich; jede sollte durch Komma getrennt sein).

STAT USR:<cr> (CP/M 2.0 und neuere)

Listet die laufende Benutzer-Nummer sowie alle Benutzer-Bereiche, für die Dateien auf derzeit angeschlossenen Platten existieren.

STAT x:DSK:<cr> (CP/M 2.0 und neuere)

Listet die Charakteristika der Platte auf Laufwerk x:.

STAT x:=R/O<cr> (CP/M 1.4 und neuere)

Ordnet Laufwerk x: einen temporären Schreibschutz-Status zu.

SUBMIT – Kommandozeilen

SUBMIT filename<cr>

Erstellt eine Datei „$$$.SUB" mit den in „filename.SUB" enthaltenen Kommandos. CP/M führt diese Kommandos dann gemäß der Datei anstatt der Tastatur aus.

SUBMIT filename parameters<cr>

Erstellt eine Datei „$$$.SUB" mit den in „filename.SUB" enthaltenen Kommandos; dabei werden gewisse Teile der Kommandozeilen aus „filename.SUB" in „$$$.SUB" durch Parameter ersetzt. CP/M führt diese Kommandos dann gemäß der Datei anstatt der Tastatur aus.

SYSGEN – Kommandozeile

SYSGEN<cr>

Lädt das Programm SYSGEN, um CP/M auf eine andere Diskette zu übertragen.

TOD (CP/M-86)

Möglichkeiten:

Setzt Uhrzeit oder zeigt sie an.

TYPE – Kommandozeile

TYPE x:filename.typ<cr>

Listet den Inhalt der angegebenen Datei auf der Konsole.

USER – Kommandozeile

USER n<cr>

Setzt die angegebene Benutzer-Nummer (0 bis 15 dec).

x: – Kommandozeile

x:<cr>

Schaltet das derzeit angeschlossene Standard-Laufwerk auf x: („A:“ bis „P:“).

Anhang B
ASCII-Zeichencodes

ASCII (American Standard Code for Information Interchange = Amerikanischer Standardcode für Informationsaustausch) besteht aus einem Satz von 96 druckbaren und 32 nichtdruckbaren Zeichen. Die meisten CP/M-Systeme benutzen zumindest eine Untermenge dieses Satzes. Wenn CP/M Zeichen in Form eines Textes auf Diskette speichert, wird von den ASCII-Definitionen Gebrauch gemacht.

Auch verschiedene CP/M-Dienstprogramme gehen von ASCII aus, so zum Beispiel ED, DDT und DDT-86 in der Wiedergabe eines „dump" (= Kernspeicher-Auszug) als Zeichenkette.

ASCII benötigt für die Wiedergabe eines Zeichens nur die unteren sieben der acht vorhandenen Bits; das oberste wird häufig zur **Parity**-Prüfung verwendet. „Parity" (= Parität) ist eine Prüfmethode für die Richtigkeit übertragener Zeichen. Das **Parity-Bit** wird von vielen Mikrocomputern und ihren Einheiten ignoriert, während andere eine der beiden nachfolgend beschriebenen Formen von Parität erfordern:

Even Parity (= gerade Anzahl)
Die Anzahl binärer Einsen in einem Byte muß gerade sein. Wenn also das Zeichen eine ungerade Anzahl vin Einsen enthält, wird das Parity-Bit 1 sein, sonst 0.

Odd Parity (= ungerade Anzahl)
Wie oben, nur ungerade.

Zu den alternativen Wegen der Codierung von Zeichen gehören der von IBM benutzte 8-bit-EBCDIC (Extended Binary Coded Decimal Interchange Code = Erweiterter binär kodierter dezimaler Austausch-Code) sowie eine Anzahl gepackter Binär-Schemata (hauptsächlich für numerische Information).

Tabelle B–1. ASCII-Zeichencode

				b7 ⟶ b6 ⟶ b5 ⟶	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b4	**b3**	**b2**	**b1**	Kol. / Reihe	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	'	<	L	\	l	|
1	1	0	1	13	CR	GS	-	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	–	o	DEL

NUL	Null	DC1	Geräte-Funktionskontrolle 1
SOH	Beginn Kopfzeile	DC2	Geräte-Funktionskontrolle 2
STX	Beginn Text	DC3	Geräte-Funktionskontrolle 3
ETX	Ende Text	DC4	Geräte-Funktionskontrolle 4
EOT	Ende Übertragung	NAK	Bestätigungs-Umkehrung
ENQ	Anfrage	SYN	Synchronizität nicht in Betrieb
ACK	Bestätigung	ETB	Ende Übertragungsblock
BEL	Akustisches Signal	CAN	Vernichte
BS	Rücktaste	EM	Ende Datenträger
HT	Horizontale Tabulation	SUB	Unterstütze
LF	Zeilenvorschub	ESC	Flucht
VT	Vertikale Tabulation	FS	Dateitrennung
FF	Formularvorschub	GS	Gruppentrennung
CR	Zeilenende	RS	Satztrennung
SO	Schiebe hinaus	US	Einheitentrennung
SI	Schiebe herein	SP	Leerzeichen
DLE	Datentrennung	DEL	Lösche

Tabelle B–2. ASCII-Zeichendarstellung in aufsteigender Reihenfolge

Hexadezimal	Binär	ASCII	Hexadezimal	Binär	ASCII
00	000 0000	NUL	30	011 0000	0
01	000 0001	SOH	31	011 0001	1
02	000 0010	STX	32	011 0010	2
03	000 0011	ETX	33	011 0011	3
04	000 0100	EOT	34	011 0100	4
05	000 0101	ENQ	35	011 0101	5
06	000 0110	ACK	36	011 0110	6
07	000 0111	BEL	37	011 0111	7
08	000 1000	BS	38	011 1000	8
09	000 1001	HT	39	011 1001	9
0A	000 1010	LF	3A	011 1010	:
0B	000 1011	VT	3B	011 1011	;
0C	000 1100	FF	3C	011 1100	<
0D	000 1101	CR	3D	011 1101	=
0E	000 1110	SO	3E	011 1110	>
0F	000 1111	SI	3F	011 1111	?
10	001 0000	DLE	40	100 0000	
11	001 0001	DC1	41	100 0001	A
12	001 0010	DC2	42	100 0010	B
13	001 0011	DC3	43	100 0011	C
14	001 0100	DC4	44	100 0100	D
15	001 0101	NAK	45	100 0101	E
16	001 0110	SYN	46	100 0110	F
17	001 0111	ETB	47	100 0111	G
18	001 1000	CAN	48	100 1000	H
19	001 1001	EM	49	100 1001	I
1A	001 1010	SUB	4A	100 1010	J
1B	001 1011	ESC	4B	100 1011	K
1C	001 1100	FS	4C	100 1100	L
1D	001 1101	GS	4D	100 1101	M
1E	001 1110	RS	4E	100 1110	N
1F	001 1111	US	4F	100 1111	O
20	010 0000	SP	50	101 0000	P
21	010 0001	!	51	101 0001	Q
22	010 0010	"	52	101 0010	R
23	010 0011	#	53	101 0011	S
24	010 0100	$	54	101 0100	T
25	010 0101	%	55	101 0101	U
26	010 0110	&	56	101 0110	V
27	010 0111	'	57	101 0111	W
28	010 1000	(	58	101 1000	X
29	010 1001	)	59	101 1001	Y
2A	010 1010	*	5A	101 1010	Z
2B	010 1011	+	5B	101 1011	[
2C	010 1100	´	5C	101 1100	\
2D	010 1101	-	5D	101 1101	]
2E	010 1110	.	5E	101 1110	^
2F	010 1111	/	5F	101 1111	–

Tabelle B–2. ASCII-Zeichendarstellung in aufsteigender Reihenfolge (Fortsetzung)

Hexadezimal	Binär	ASCII	Hexadezimal	Binär	ASCII
60	110 0000		70	111 0000	p
61	110 0001	a	71	111 0001	q
62	110 0010	b	72	111 0010	r
63	110 0011	c	73	111 0011	s
64	110 0100	d	74	111 0100	t
65	110 0101	e	75	111 0101	u
66	110 0110	f	76	111 0110	v
67	110 0111	g	77	111 0111	w
68	110 1000	h	78	111 1000	x
69	110 1001	i	79	111 1001	y
6A	110 1010	j	7A	111 1010	z
6B	110 1011	k	7B	111 1011	{
6C	110 1100	l	7C	111 1100	|
6D	110 1101	m	7D	111 1101	}
6E	110 1110	n	7E	111 1110	~
6F	110 1111	o	7F	111 1111	DEL

Anhang C
Vergleich der Versionen 1.3, 1.4 und 2.2 des CP/M-80 sowie des CP/M-86

Tabelle C–1. Kommandos für die Zeilenausgabe

Kommando	Version 1.3	Version 1.4	Version 2.2	CP/M-86
CONTROL-C	Ja	Ja	Ja	Ja
CONTROL-E	Nein	Ja	Ja	Ja
CONTROL-H oder BACKSPACE	Nein	Nein	Ja	Ja
CONTROL-J oder LINEFEED	Nein	Nein	Ja	Ja
CONTROL-M oder CARRIAGE RETURN	Ja	Ja	Ja	Ja
CONTROL-P	Ja	Ja	Ja	Ja
CONTROL-R	Nein	Ja	Verbessert	Verbessert
CONTROL-S	Ja	Ja	Ja	Ja
CONTROL-U	Ja	Ja	Verbessert	Verbessert
CONTROL-X	Nein	wie CONTROL-U	BACKSPACE und ERASE	Ja
DELETE oder RUBOUT	Ja	Ja	Ja	Ja

Tabelle C–2. Neue oder geänderte Kommandos

Kommando	Version 1.3	Version 1.4	Version 2.2	CP/M-86
DIR	Ja	Ja	Neu: Zeigt pro Zeile 4 Dateinamen an. Zeigt nur DIR-Dateinamen an. Zeigt Dateinamen nur für den derzeit aktiven Benutzer an.	Ja
DIRS	Nein	Nein	Nein	Zeigt alle Dateien an, auch Systemdateien
ED	Ja	Neu: +/–V 0X 0V nnnn: nX :mmmm Einzigartig: E, H, Q, O Tabs immer als Leerzeichen geechot.	Neu: +V ist Standardannahme. Kann R/O-Datei nicht ändern. Kann auf SYS-Datei nicht zugreifen.	Ja
ERA	Ja	Neu Fragt ALL? nach ERA *.*	Neu Löscht nur Dateien des derzeit aktiven Benutzers.	Ja
PIP	Ja	Neu: Options-Parameter. Physische Gerätenamen. Mehrdeutige Dateinamen.	Neue Parameter: Gn R W	Ja
SAVE	Ja	Ja	Neu Ändert Benutzer-Kernspeicher nicht (TPA).	Nein (s. DDT-86)
STAT	Ja Zeigt nur freie Plattenkapazität an.	Neu: Zeigt den Umfang jeder Datei an. log: = phy: x: = R/O Erkennt gekoppelte Platten. Zeigt die freie Kapazität aller aktiven Platten an.	Neu: VAL: Menu DSK: x:DSK: USR: $S Dateiattribute: R/W, R/O DIR, SYS	Ja
SUBMIT	Ja	Ja	Neu: XSUB	Ja (Kein XSUB)
USER	Nein	Nein	Ja	Ja

Tabelle C–3. Unterschiede bei Platten

Item	Version 1.3	Version 1.4	Version 2.2	CP/M-86
Maximale Anzahl von Laufwerken	2	4	16	16
Maximale Speicherkapazität pro Laufwerk	1 MB*	1 MB	16 MB	16 MB
Maximale Anzahl von Dateien	64	64	Ausweitbar	Ausweitbar
Zugriffsmethode	Sequentiell	Sequentiell	Sequentiell oder direkt	Sequentiell oder direkt
Lage der Plattencharakteristika	BDOS	Platten-Parameterblock	BIOS	BIOS

* MB = Megabyte

Tabelle C–4. Neue oder geänderte BDOS-Funktionen

Funktion	Version 1.3	Version 1.4	Version 2.2	CP/M-86
6 Steuere Konsolen-I/O	Nein	Nein	Ja	Ja
10 Lies Konsolen-Puffer	Ja	Ja	Verbesserte Zeilenausgabe	Verbesserte Zeilenausgabe
12 (siehe rechts)	Hebe Plattenkopf	Hebe Plattenkopf	Melde Versionsnummer	Melde Versionsnummer
15 Datei-Eröffnung	Ja	Ja	Verbessert	Verbessert
17 Suche erste	Ja	Ja	Verbessert	Verbessert
18 Suche nächste	Ja	Ja	Verbessert	Verbessert
19 Lösche Datei	Ja	Ja	Verbessert	Verbessert
22 Lege Datei an	Ja	Ja	Verbessert	Verbessert
23 Benenne Datei um	Ja	Ja	Verbessert	Verbessert
24 Melde Login-Vector	Ja	Ja	Verbessert	Verbessert
28 bis 36	Nein	Nein	Ja	Ja

Anhang D
CP/M-Anforderungen

X>	CP/M wartet auf Ihr Kommando; x: ist das derzeit aktive Standard-Laufwerk.
nx>	MP/M wartet auf Ihr Kommando; x: ist das derzeit aktive Standard-Laufwerk; n ist die derzeit aktive Benutzer-Nummer.
*	PIP wartet auf Ihr Kommando.
*	ED wartet auf Ihr Kommando.
nnnn:*	ED wartet auf Ihr Kommando; der Zeichenvektor steht auf Zeilennummer nnnn.
*	Zeigt auch bei Microsoft BASIC, EDIT, FORTRAN, COBOL und Pascal eine Kommando-Anforderung an.
–	DDT wartet auf Ihr Kommando.

Tragen Sie ab hier die Kommando-Anforderungen anderer von Ihnen benutzter Programme ein:

Anhang E
Diskettenauswahl

In jeder der angegebenen Kategorien gibt es Disketten von einfacher oder doppelter Dichte. Stellen Sie sicher, daß Sie die richtige Dichte für Ihr System wählen. Gewöhnlich können Systeme, die doppelte Dichte zulassen, auch mit einfacher Dichte fahren, so daß Sie hier beide Möglichkeiten nutzen können.

Systeme mit softsektorierten 8-Zoll-Disketten

Altos
Cromemco FDC Controller
Cromemco System 3
Delta
Digital Microsystems
Discus
Dynabyte DB8/4
Godbout
iCOM 3712, 3812
IMS 8000
IMSAI FDC2
IMSAI VDP-80
Intecolor (ISC) 8063, 8360, 8963
Intel MDS
Micromation
Morrow Discus
Mostek
Ohio Scientific C3
Pertec PCC 2000
Processor Technology Helios
Radio Shack TRS-80 Model II
Radio Shack TRS-80 Model I/Micromation
Radio Shack TRS-80 Model I/Omikron
Radio Shack TRS-80 Model I/Shuffleboard
Research Machines
SD Systems
Spacebyte
Tarbell
TEI
Thinker Toys
TRS-80. Vgl. Radio Shack
Vector Graphic System 2800

Systeme mit hartsektorierten 8-Zoll-Disketten

MITS 3200, 3202
Processor Technology Helios II

Systeme mit softsektorierten 5¼-Zoll-Disketten

Apple/SoftCard
AVL Eagle
BASF System 7100
Cromemco Z2D
DEC VT180
Digi-Log Microterm II
Durango F-85
Gnat
IBM Personal Computer
Hewlett-Packard HP-80 Series
Heath/Zenith double-density
iCOM 2411 Micro Floppy
IMS 5000
IMSAI VDP-40, -42, -44
Intertec SuperBrain
Kontron PSI-80
MSD
Osborne 1
Polymorphic 8813
Quay 500, 520
Radio Shack TRS-80 Model I
Radio Shack TRS-80 Model
I/FEC Freedom
Radio Shack TRS-80 Model
I/Omikron
RAIR
Research Machines
Sanco 7000
SD Systems
SuperBrain. Vgl. Intertec
TEI
Quay
Vector Graphic (gegenwärtige Systeme)
Xerox 820

Systeme mit hartsektorierten 5¼-Zoll-10-Sektor-Disketten

Heath H8 and H17, H27
Heath H89

Horizon
Meca
North Star Horizon
Vista V80
Vista V200
Zenith Z89

Systeme mit hartsektorierten 5¼-Zoll-16-Sektor-Disketten

Blackhawk (40 TPI)
CDS Versatile 3B (40 TPI)
CDS Versatile 4 (100 TPI)
COMPAL-80 (100 TPI)
Dynabyte (einige Modelle)
Exidy Sorcerer (100 TPI)
Micropolis Mod I (40 TPI)
Micropolis Mod II (100 TPI)
Nylac (40 TPI, 100 TPI)
REX (40 TPI)
Sorcerer. Vgl. Exidy
Vector Graphic (100 TPI)
Vector MZ (100 TPI)
Versatile. Vgl. CDS

Anhang F
Kommentierende Bibliographie

CP/M allgemein

Ballinger, Charles. „HDOS or CP/M?". **Interface Age**, September 1980, pp. 88–91.
Vergleicht zwei Plattenbetriebssysteme für Heath-Computer.

Brigham, Bruce, ed. **CP/M Summary Guide**. Glastonbury. Ct.: Rainbow Associates, 1980.
Das Buch liefert die Kommandos für CP/M, DESPOOL, MAC, TEX, CBASIC und BASIC-80 von Microsoft in übersichtlicher Weise. Eine nützliche computerbezogene Hilfe.

Epstein, Jake, and Terry, Chris. „Introduction to CP/M; The CP/M Connection". **S100 Microsystems**, September/Oktober 1980, pp. 10–32.
Fortsetzung einer Serie über CP/M-Artikel.

Fernandez, Judi, and Ashley, Ruth. **Using CP/M**. New York: John Wiley Sons, 1980.
Ein Lehrbuch für Autodidakten, geschrieben in einem Frage-und-Antwort-Format, bezogen auf technische Information. Eine Schnell-Fibel für CP/M-Kommandos und ihre Syntax.

Fritzon, Richard. „The New CP/M: Is It Worth It?". **Kilobaud Microcomputing**, Juli 1980, p. 66.
Eine kurze Zusammenfassung der Unterschiede zwischen den Versionen 1.4 und 2.0 von CP/M.

Miller, Alan. „Diagnostics Package for CP/M". **Interface Age**, Oktober 1980, p. 104.
Überprüft das Supersoft-CP/M-Diagnostik-Paket; diese Programme testen Kernspeicher, CPU, Platte, Drucker und Konsole.

North, Steve. „The CP/M Disk Operating System". **Creative Computing**, November/Dezember 1978, pp. 52–53.
Eine kurze Zusammenfassung des CP/M-Betriebssystems und der begleitenden Programme.

Stewart, John. „CP/M Primer; A Most Sophisticated Operating System". **Kilobaud Microcomputing**, April 1978, pp. 30–34.
Eine frühe detaillierte Beschreibung des CP/M-Betriebssystems.

„Upgraded CP/M Floppy Disc Operating System". **Dr. Dobbs**, November 1976, p. 51.
Detaillierter Überblick über die Merkmale der Neufassung 1.4 von CP/M. Schließt eine Beschreibung der begleitenden 1.4-Handbücher ein.

Warren, Jim. „First Word on a Floppy-Disc Operating System". **Dr. Dobbs**, April 1976, p. 5.
Die erste veröffentlichte Information über CP/M.

——. „The Time for Floppy's Is Just About Now!". **Dr. Dobbs**, August 1976, p. 5.
Betrachtet die zu diesem Zeitpunkt erhältlichen Floppy-Disk-Systeme und gibt einen Überblick über CP/M-Merkmale.

Zaks, Rodnay. **The CP/M Handbook with MP/M**. Berkeley: Sybex, 1980.
Diskutiert CP/M und MP/M – Kommandos, Programme und Einrichtungen.

CP/M-kompatible Software

Collins, Rosann; Hines, Theodore; and Rowan, George. „Manipulating Pencil Files; Convert Them to BASIC". **Creative Computing**, August 1979, pp. 98–99.
Illustriert, wie mit Electric Pencil durch Processor Technology's BASIC erstellte Dateien zu benutzen sind; für viele verschiedene CP/M-kompatible BASICs erhältlich.

Craig, John. „A New Kind of Pencil!". **Creative Computing**, Februar 1979, pp. 30–33.
Gründliche Nachforschungen über das Textverarbeitungssystem von Electric Pencil.

Didday, Rich. „Universal Data Entry System; In a Car Pooling Application". **Creative Computing**, Mai 1980, pp. 102–110.
Befaßt sich mit UDE von Software Store und demonstriert dieses Programm.

Eubanks, Gordon. „Notes on CP/M's BASIC-E". **Dr. Dobbs** 19:35.
Brief an den Herausgeber über die Versionen des BASIC-E, seinen öffentlich-rechtlichen Status, und wie man revidierte Versionen erhält.

(**Anmerkung:** Gordon Eubanks entwickelte BASIC-E für eine MA-Dissertation; es ist der Vorfahre von CBASIC und CBASIC2.)

Fitzgerald, Jim. „Off-the-Shelf Word-Processing System". **Kilobaud Microcomputing**, September 1980, pp. 92–94.
Beschreibt ein CP/M-System mit Electric Pencil.

Foster, Charlie. „Pascal with a Z80". **Interface Age**, November 1980, pp. 60–62.
Gibt einen Überblick über Pascal/Z von Ithaca Intersystems, unter besonderer Berücksichtigung der Unterschiede gegenüber UCSD Pascal.

Hallen, Rod. „Super Word Processors". **Kilobaud Microcomputing**, Juni 1980, pp. 214–217.

Überblick über die Merkmale von ED, EDIT, Electric Pencil, WordStar und The Magic Wand ohne Kommentar – eine kurze Bibliographie von Artikeln über Textverarbeitung.

——. „Tarbell Disk BASIC". **Kilobaud**, Mai 1980, pp. 168–170.
Überblick über die Merkmale von Tarbell Disk BASIC im Vergleich zu anderen Interpretern.

——. „The Battle of the Word Processors". **Creative Computing**, November 1979, pp. 48–53.
Überblick über ED, TEX, EDIT und Electric Pencil.

Hamilton, R. W. „WPDaisy Word PRocessing System". **Creative Computing**, Mai 1979, pp. 36–41.
Beschreibt ein wenig bekanntes, aber leistungsfähiges Textverarbeitungssystem.

Hart, Glenn. „New BASIC from Tarbell". **Creative Computing**, Januar 1980, pp. 20–23.
Beschreibt ein von CP/M-Systemen etwas unterschiedliches BASIC.

——. „Magic Wand Word Processor". **Creative Computing**, August 1980, pp. 38–45.
Beschreibt den Aufbau des Magic Wand-Textverarbeitungssystems.

Heintz, Carl. „Analyst: Another Data Base Manager". **Interface Age**, Dezember 1980, pp. 42–44.
Beschreibt das Analyst-Datenbank-Management-System von Structured Systems.

——. „Maxiledger". **Interface Age**, September 1980, pp. 42–44.
Beschreibt das Maxiledger-„genral ledger accounting system" (= generelles Hauptbuch-Führungssystem) von Compumax Associates.

——. „A Peach of a General Ledger Program". **Interface Age**, Oktober 1980, pp. 46–48.
Überblick über das Peachtree General Ledger-System.

——. „Pearl – A Novel Programming Gem". **Interface Age**, November 1980, p. 47.
Ein Programm, das Programme „schreibt"; Pearl schreibt geschäftsorientierte Anwender-Pakete gemäß der Benutzer-Eingabe. Hier wird erklärt, wie mächtig solch ein Konzept sein kann.

Heyman, Victor. „IDSWORD – The Comprehensive Processing System for Home and Business". **Creative Computing**, pp. 43–44.
Beschreibt ein anderes wenig bekanntes CP/M-kompatibles Textverarbeitungssystem.

Hogg, Douglas. „How Good Is Microsoft's FORTRAN-80?". **Creative Computing**, Januar 1979, pp. 62–67.
Beschreibt FORTRAN-80 von Microsoft und untersucht allgemein die Implementation von FORTRAN auf einem Mikrocomputer.

Johnson, Bob. „Business Software Review". **Interface Age**, August 1979, pp. 38–39.
Gibt einen Überblick über das „Graham Dorian Apartment Management and Payroll"-Software-Paket.

Kendall, Wallace. „Prettyprinting with Microsoft BASIC". **Kilobaud Microcomputing**, Mai 1979, p. 80.
Präsentiert einen einfachen Weg, Programmauflistungen mit Microsoft BASIC zu formatisieren.

Knecht, Ken. „CBASIC Review". **80 Microcomputing**, April 1980, pp. 130–132.
Vergleicht CBASIC-Einrichtungen mit Microsoft-BASIC und führt dabei gleichzeitig das CP/M-Betriebssystem von FMG für den TRS-80 ein.

Lindsay, John. „New Version of BASIC". **Kilobaud**, Mai 1980, pp. 72–74.
Faßt die Merkmale der Version 5 des Microsoft BASIC zusammen und diskutiert die Unterschiede zwischen den Versionen.

Lutz, Dick. „Sharpening Your Pencil". **Creative Computing**, März 1980, pp. 30–35.
Ein Programm zur Erweiterung des Designs von Electric Pencil.

Magruder, Bob. „The New WPDaisy: Word-Processing Software". **onComputing**, Herbst 1980, pp. 68-74.
Eine ausgiebige Darstellung von WPDaisy.

McClure, James. „CBASIC – A Review". **Creative Computing**, September 1979, pp. 48–51.
Beschreibt eine häufig benutzte höhere Programmiersprache. Bietet Timing-Vergleiche für eine Reihe standardisierter Maßstäbe mit dem Beweis, daß sich CBASIC in vielen Tests als relativ langsam erwiesen hat.

——. „Microsoft vs. MicroFocus COBOL". **Creative Computing**, März 1980, pp. 20–29.
Vergleicht die Eigenschaften zweier Erst-Implementationen von COBOL für CP/M-Benutzer.

——. „A Personal Finance System". **Kilobaud Microcomputing**, Juni 1979, pp. 74–78; Juli 1979, pp. 50–56; August 1979, pp. 66–75.
Ein ausführliches CBASIC2-Programm zum Management persönlicher Finanzen.

Miller, Alan. „BASCOM: Microsoft's BASIC Compiler for the 8080/Z80". **Interface Age**, Juli 1980, pp. 124–126.
Diskutiert den Unterschied zwischen Compilern und Interpretern unter

besonderer Berücksichtigung des BASIC-Compilers von Microsoft mit Beispielprogrammen.

——. „CBASIC: A Business-Oriented Language for CP/M". **Interface Age**, August 1979, pp. 116–119.
Überblick und Diskussion über die Hochsprache CBASIC für CP/M.

——. „CP/M for the TRS-80 Model II: Lifeboat and FMG Corp. Versions". **Interface Age**, November 1980, pp. 94–98.
Ein detaillierter Vergleich zweier unterschiedlicher Implementationen von CP/M für Modell II.

——. „The Electric Pencil for CP/M". **Interface Age**, August 1978, pp. 148–149.
Ein Review für ein populäres Textverarbeitungssystem.

——. „Pascal for CP/M: Digital Marketing's Pascal/M". **Interface Age**, September 1980, pp. 96–103.
Betrachtet Pascal/M und bestätigt den Unterschied zu anderen Hochsprachen.

North, Steve. „Creative Computing Reviews Five Software Packages – Tiny C, Microsoft BASIC 5.0, Research Machines Z80 Algol, Structural Analysis SP80 Macros, Digital Research CP/M 2.0 and MP/M". **Creative Computing**, März 1980, pp. 40–44.
Überblick über die Titel.

Pournelle, Jerry. „Omikron TRS-80 Boards, NEWDOS+, and Sundry Other Matters". **Byte**, Juli 1980, pp. 198–208.
Ein überwältigender Einblick in den Erfahrungsbereich von Pournelle mit Mikrocomputern – er ist ein populärer Science Fiction-Autor (**Mote in God's Eye, Lucifer's Hammer** und andere) und bietet eine persönliche und faszinierende Schau.

Press, Larry. „A Review of Four Text-Formatting Programs". **onComputing**, Herbst 1980, pp. 48–54.
Vergleicht S-80, TEX, TPS und Textwriter.

——. „Word Processors: A Look at Four Popular Programs". **onComputing**, Sommer 1980, pp. 38–52.
Ein detaillierter Vergleich von Auto Scribe, Electric Pencil, Magic Wand und WordStar.

Sanger, Joseph. „The Electronic Librarian". **Kilobaud Microcomputing**, November 1979, pp. 44–62.
Ein BASIC-E-Programm für die Pflege von Informations-Datenbanken.

Sjowall, Tor. „CP/M to UCSD Pascal File Conversion". **Dr. Dobbs**, Oktober 1980, pp. 16–19.
Pascal-Quellcode zur Konvertierung von CP/M-Dateien in das UCSD-Pascal-Format.

VanHorn, Eric. „Super-Sort by Micro-Pro International". **Creative Computing**, Juli 1979, pp. 34–37.
Super-Sort.

CP/M und Assembler-Programmierung

Barbier, Ken. „CP/M for Single-Drive Systems". **Kilobaud Microcomputing**, September 1980, pp. 94–98.
Erklärt die Probleme bei einem CP/M-System mit einfachem Laufwerk; schließt „assembly language single-drive file copy program source listing" (= Assemble Einfachlaufwerk-Kopierprogramm-Quellauflistung) ein.

Barker, Lee. „Help with OSI's CP/M". **Dr. Dobbs**, Mai 1980, pp. 36–37.
Eine Modifikation des BIOS für OSI-Computer, wobei CP/M erfaßte Probleme ausradiert.

Biese, Leo and Iannuccillo, Emilio. „MASTHEAD: Why Not Title Your Printouts?". **Interface Age**, August 1980, pp. 122–127.
Eine Methode, Titel für Programmlisten groß zu drucken, wobei der Quellcode in Assembler oder Microsoft BASIC angegeben werden kann.

Cecil, Alex. „ACT: An 8080 Macroprocessor". **Dr. Dobbs** 23:20–45.
Ein „TRAC"-ähnlicher Text-Interpreter; ACT ist in PL/M geschrieben. Sollte für Studenten interessant sein.

Christensen, Ward. „An 8080 Disassembler". **Dr. Dobbs**, Februar 1977, pp. 30–43.
Eine vollständige Auflistung eines Assembler-Programms für einen 8080-Disassembler. Die Dokumentation erklärt das Problem der Disassemblierung allgemein.

Cotton, Gene. „How to Solve Your Damaged Disk Dilemma". **Interface Age**, September 1980, pp. 80–86, 130–131.
Ein Assembler-Programm zur Rettung von Fehlerstellen auf einer Diskette namens .bad. Die Beschreibung enthält eine ausführliche Information, wie CP/M Information auf Diskette gliedert.

Epstein, Jake. „An Introduction to CP/M". **S-100 Microsystems**, Januar/Februar 1980, pp. 6–10, März/April 1980, pp. 28–33, Mai/Juni 1980, pp. 12–17.
Eine ausgezeichnete Einführung in CP/M für Programmierer mit ausführlicher technischer Information.

Foster, Charlie and Meador, Richard. „8080 Dynatrace". **S-100 Microsystems**, Juli/August 1980, pp. 22–31.
Ein dynamisches bildschirmorientiertes Assembler-Testsystem im 8080-Assembler (der Artikel beinhaltet eine Expertise zur Auflösung der Ausgabe-Routinen).

Frantz, James. „Turn-Key CP/M Systems". **Creative Computing**, Dezember 1979, pp. 104–107.
Schritt-für-Schritt-Direktiven, um CP/M automatisch nach einem Kaltstart ausführbar zu machen.

Friedman, David. „Las Vegas Super Slot: A CP/M Game Machine Program Using Flashwriter I Graphics". **Dr. Dobbs**, November/Dezember 1980, pp. 10–22.
Ein Spielprogramm für CP/M-Systeme mit „Vector Graphic's Flashwriter I board" in Assembler.

Gagne, Jim. „Vice Versa – Pencil to CP/M and Reverse". **Dr. Dobbs**, März 1979, pp. 26–29.
Wie man Electric Pencil-Dateien ins CP/M-Format reformatisiert und umgekehrt; Assembler-Code vorausgesetzt.

Haanstra, Bruce. „Optional Printing with CP/M and Microsoft BASIC". **Interface Age**, November 1980. pp. 84–86.
Illustriert, wie die Abfangvorrichtung des ^P-Zeichens im Microsoft BASIC umgangen werden kann, wenn man den Drucker in CP/M ein- und ausschalten will.

Hallen, Rod. „Battle of the Assemblers". **Creative Computing**, Dezember 1979, pp. 42–45.
Vergleicht CP/M-gestütztes ASM mit ASMB, einem von Technical Systems Consultants erhältlichen Assembler. Beschreibt auch kurz einen Assembler namens „SASSY".

Hoffer, W. C. „Data and Time for the CP/M Operating Systems". **Interface Age**, August 1978, pp. 152–156.
Assembler-Routine, die eine CompuTime-Anlage befähigt, automatisch Zeit und Datum von Programmauflistungen zu erfassen.

Kildall, Gary. „Simple Technique for Static Relocation". **Dr. Dobbs**, 22:10–13.
Der Autor des CP/M beschreibt dessen Technik der Verschieblichkeit.

Miller, Alan. „CP/M Part 2 – A Macro Assembler and other Goodies". **Interface Age**, Dezember 1978, pp. 130–135.
Beschreibt den MAC-Assembler von Digital Research, die SID-Testhilfe für symbolische Instruktionen von Digital Research sowie ein Beispiele-BIOS für ein CP/M-System von North Star.

——. „An Interrupt-Driven Keyboard Buffer". **Interface Age**, Oktober 1980, pp. 106–107, 137–141.
Eine alternative Weise der Implementation einer Erkennungsroutine der Tastenauslösung für das CP/M-BIOS von North Star.

——. „Structured Assembly-Language Programming for the 8080". **Interface Age**, November 1979, pp. 153–155.

Beschreibt SP80, einen Satz von Makro-Routinen in strukturierter Assemblersprache für CP/M-Systeme. Vergleicht den MAC-Assembler von Digital Research mit SP80.

——. „ZSID Z80 Debugger for CP/M". **Interface Age**, August 1980, pp. 88–90.
Beschreibt die Fehlererkennung ZSID von Digital Research für symbolische Assemblerprogramme und geht auch auf das DESPOOL-Programm von Digital Research ein.

Parsons, Ronald. „UCSD Pascal to CP/M File Transfer Program". **Dr. Dobbs**, August 1979, pp. 12–16.
Assembler-Routine für Programmübertragungen vom UCSD-Pascal-Format in CP/M-Dateistruktur.

Pugh, Tim. „Intelligent Terminal Implementation on S100 bus". **Dr. Dobbs**, 26:4–16.
Ein wohlkommentiertes Assemblerprogramm, das die Kommunikation zwischen jedem CP/M-System mit einem IDS-88 – Modem und irgendeinem anderen Computer-System erlaubt.

Terry, Chris. „The CP/M Connection". S-100 **Microsystems**, Juli/August 1980, pp. 32–35.
Diskutiert die Verschieblichkeits-Methodik des CP/M unter besonderer Berücksichtigung des Anschlusses von CBIOS und anderen Modifikationen eines verschobenen Systems.

Van Buer, Darrel. „A Table-Driven Assembler on CP/M". **Dr. Dobbs**, Februar 1980, pp. 18–25.
Ein PL/M-Quellcode für einen Makro-Assembler.

Willoghby, Steve. „Hardcopy Device Driver Programs for CP/M". **Dr. Dobbs** 48:34–37.
Assembler-Routinen zum Einschluß in die CP/M-CBIOS-Sektion für Diablo 1650- oder Fernschreib-Drucker.

Anhang G
CP/M-Bezugsquellenverzeichnis

Hersteller CP/M-kompatibler Sprachen, Programme und Computer

Textverarbeitungsprogramme

Magic Wand	Small Business Applications, Inc.
Microsoft Edit	Microsoft, Inc.
Electric Pencil	Michael Shrayer Software, Inc.
SCOPE	Vector Graphic, Inc.
WordStar	MicroPro International

Sprachen

Assemblers	Digital Research, Inc.
	Microsoft, Inc.
	Sorcim Corp.
ADA	Supersoft, Inc.
BASIC	Microsoft, Inc.
	Supersoft, Inc.
C	Supersoft, Inc.
	Whitesmiths, Ltd.
CBASIC	Digital Research, Inc.
COBOL	Microfocus, Inc.
	Micro Soft, Inc.
FORTRAN	Microsoft, Inc.
	Supersoft, Inc.
FORTH	Forth, Inc.
	Supersoft, Inc.
LISP	Microsoft, Inc.
	Supersoft, Inc.
Pascal	Digital Research, Inc.
	Ithaca Intersystems, Inc.
	Sorcim Corp.
PL/I	Digital Research, Inc.

Datenmanagement-Systeme

SELECTOR-IV	Micro.AP, Inc.
Pearl	Computer Pathways Unlimited
HDMS	Micro Data Base Systems

Von CP/M abgeleitete Betriebssysteme

CDOS	Cromemco, Inc.
SDOS	SD Systems
TPM	Computer Design Labs
TSA/OS	TSA Software

Anhang H
Index der Hersteller

ADDS
Applied Digital Data Systems
100 Marcus Boulevard
Hauppauge, NY 11787
(516) 231-5400

Altos Computer Systems
2360 Bering Drive
San Jose, CA 95131
(408) 946-6700

ANSI
American National Standards Institute
1430 Broadway
New York, NY 10018
(212) 354-3300

Apple Computer
10260 Bandley Drive
Cupertino, CA 95014
(800) 538-9696 (except California)
(800) 662-9238 (from California)

BDS
BD Software
See Lifeboat Associates

Bell Laboratories–C Language
Patent License Organization
Western Electric Company
P.O. Box 25000
Greensboro, NC 27420
(919) 697-6530

Byte Publications, Inc.
70 Main Street
Peterborough, NH 03458
(603) 924-9281

Compiler Systems, Inc.
P.O. Box 145
Sierra Madre, CA 91024
(213) 355-4211

Computer Design Labs
342 Columbus Avenue
Trenton, NJ 08629
(609) 599-2146

Computer Pathways Unlimited
2151 Davcor Street S.E.
Salem. OR 97302
(503) 363-8929

CP/M Users Group
1651 Third Avenue
New York, NY 10028

Cromemco, Inc.
280 Bernard Avenue
Mountain View, CA 94043
(415) 964-7400

Digital Microsystems
1840 Embarcadero
Oakland, CA 94606
(415) 582-3686

Digital Research
P.O. Box 579
801 Lighthouse Avenue
Pacific Grove, CA 93950
(408) 649-3896

DynaByte, Inc.
115 Independence Drive
Menlo Park, CA 94025
(415) 329-8021

Exidy
1234 Elko Drive
Sunnyvale, CA 94086
(408) 734-9410

FORTH, Inc.
2309 Pacific Coast Highway
Hermosa Beach, CA 90254
(213) 372-8493

Heath Data Systems
Hilltop Road
St. Joseph, MI 49085
(616) 982-3200

Industrial Micro Systems
628 N. Eckhoff Street
Orange, CA 92668
(714) 978-6966

Intel Corporation
3065 Bowers Avenue
Santa Clara, CA 95051
(408) 987-8080

Ithaca Intersystems, Inc.
1650 Hanshaw Road
P.O. Box 91
Ithaca, NY 14850
(607) 257-0190

Lifeboat Associates
1651 Third Avenue
New York, NY 10028
(212) 860-0300

Lifelines Publishing Corp.
1651 Third Avenue
New York, NY 10028
(212) 722-1700

Michael Shrayer Software, Inc.
1198 Los Robles Drive
Palm Springs, CA 92262
(714) 323-1400

Micro Data Base Systems
P.O. Box 248
Lafayette, IN 47902
(317) 448-1616

MICRO. AP, Inc.
9807 Davona Drive
San Ramon, CA 94583
(415) 828-6697

Micro Focus Inc.
1620 Civic Center Drive
Santa Clara, CA 95050
(408) 984-6961

Micromation, Inc.
1620 Montgomery Street
San Francisco, CA 94111
(415) 398-0289

MicroPro International Corp.
1299 4th Street
San Rafael, CA 94901
(415) 457-8990

Microsoft, Inc.
10800 N.E. Eighth, Suite 819
Bellevue, WA 98004
(206) 455-8080

MITS
See Pertec Computer Corp.

MMS
Miller Microcomputer Services
61 Lake Shore Road
Natick, MA 01760
(617) 653-6136

Morrow Designs
5221 Central Avenue
Richmond, CA 94804
(415) 524-2101

Ohio Scientific
1333 South Chillicothe Road
Aurora, OH 44202
(216) 831-5600
(800) 321-6850

Onyx Systems, Inc.
10375 Bandley Drive
Cupertino, CA 95014
(408) 257-8022

Pertec Computer Corp.
20630 Nordhoff Street
Chatsworth, CA 91311
(213) 998-1800

Radio Shack
1300 One Tandy Center
Fort Worth, TX 76102
(817) 390-3700

SD Systems
P.O. Box 28810
Dallas, TX 75228
(214) 271-4667

Small Business Applications, Inc.
3220 Louisiana, Suite 205
Houston, TX 77006
(713) 528-5158

Sorcim Corp.
405 Aldo Avenue
Santa Clara, CA 95050
(408) 727-7634

Supersoft, Inc.
P.O. Box 1628
Champaign, IL 61820
(217) 359-2112

Tarbell Electronics
950 Dovlen Place, Suite B
Carson, CA 90746
(213) 538-4251

Texas Instruments, Inc.
P.O. Box 1444
Houston, TX 77001
(800) 257-7850
(except NJ)
(800) 322-8650
(from NJ)

Thinkertoys
See Morrow Designs

TSA Software
5 North Salem Road
Ridgefield, CT 06877
(203) 438-3954

Vector Graphic, Inc.
31364 Via Colinas
Westlake Village, CA 91362
(213) 991-2302

Whitesmiths, Ltd.
P.O. Box 1132
Ansonia Station, NY 10023
(212) 799-1200

Zenith
See Heath Data Systems

Glossarium

Abschließen = Close.
Eine Datei abzuschließen bedeutet, (zumindest zeitweise) jede damit verknüpfte Verarbeitung zu beenden. Ein „Close" macht Informationsspeicherung und Suchvorgänge auf Dateien schneller und effizienter. CP/M aktualisiert (Update) eine Datei nicht völlig, bevor sie abgeschlossen ist.

Anweisung = Statement.
Eine Anweisung ist eine komplette Instruktion (oder eine Kombination von Instruktionen) an einen Computer. Die Komplexität einer Anweisung (man denke sie sich als einen Computer-Satz) hängt ganz von der benutzten Sprache und dem Stil des Programmierers ab.

ASCII.
American Standard Code for Information Interchange (= amerikanischer Standard-Code für Informationsaustausch) bezieht sich auf die interne Wiedererkennung von Buchstaben, Zahlen und Symbolen der englischen Sprache durch den Computer. Anhang B liefert eine vollständige Auflistung dieser Darstellungen.

Assemblersprache = Assembly Language.
Assemblersprache besteht aus einer Reihe von Namen (mnemonische Symbole genannt), die die verschiedenen Kombinationen von „1" und „0" darstellen, die den Computer instruieren. Anstatt daß B„11001001" (Maschineninstruktion) eingegeben wird, bewirkt das mnemonische Symbol „RET" dasselbe: der Computer verzweigt (RETURN) zu einer vorher gespeicherten Rücksprungadresse.

Aufruf = Call.
Ein „Call" weist den Computer an, die Kontrolle an eine andere Sektion des Kernspeichers abzugeben, an der sich die aufgerufene Routine befindet. Aufrufe werden in der Programmierung oft benutzt, denn viele Prozesse teilen sich in Unterroutinen auf. Anstatt die gleiche Routine auf verschiedene Kernspeicherplätze zu duplizieren, wird sie an einer Adresse gespeichert und falls benötigt aufgerufen. Wenn der Aufruf beendet ist, übergibt er die Kontrolle der dem „Call" direkt folgenden Instruktion.

Ausführung = Execute.
Ein Programm auszuführen bedeutet, daß der Computer alle Instruktionen so ausführt, wie das Programm sie enthält. Wenn die CPU ein Stück Information als Befehl interpretiert, führt sie ihn aus.

Band = Tape.
Ein Computerband gleicht einem Tonband. Es besteht gewöhnlich aus einem ½- oder 1-Zoll-Magnetband, das auf eine Spule gewickelt ist. Computer speichern Informationen in einer höheren Dichte auf Band als Geräte für den Hausgebrauch. Die Information wird unterschiedlich gespeichert:

digitale Impulse gegen analoge Darstellungen. Ein Band ist ein einfach zu handhabendes und kostengünstiges Medium zur Speicherung von Daten oder Programmen. Jedoch sind Bänder nicht für direkten Zugriff geeignet. Je länger das Band, desto länger ist auch die Wartezeit für die Plazierung eines Satzes. Bandverarbeitung bleibt eine Hauptstütze für Klein- und Großrechenanlagen, und Bänder werden als kostengünstiges Speichermedium für Heimcomputer benutzt. Dennoch wird die Bandverarbeitung bei CP/M-Systemen selten angewandt.

BASIC.
Beginners All-purpose Symbolic Instruction Code (= Allzweck-Instruktionscode für Anfänger) ist eine höhere Programmiersprache. Es ist mehr als eine symbolische Repräsentation der Instruktionen, die ein Computer interpretieren kann. So enthält z. B. die BASIC-Anweisung „PRINT" viele Computer-Instruktionen.

Betriebssystem = Operating System.
Das Betriebssystem (OS) ist das Gehirn des Computers. Es ist darauf ausgelegt, zu „wissen", welche Peripherie verfügbar und wie mit ihr zu kommunizieren ist. Das Betriebssystem kontrolliert die gesamte Ausstattung, führt aber gewöhnlich (wie auch im Falle des CP/M) keine komplizierten Prozeduren durch.

Bibliotheksdatei = Library File.
Sie enthält eine Reihe von Standard-Routinen, die man einem neuen Programm beifügen kann. Theoretisch könnte man aus verschiedenen Bibliotheksdateien ein komplettes Programm zusammenstellen.

COBOL.
Common Business Oriented Language (= allgemeine geschäftsorientierte Sprache) instruiert den Computer in englischähnlichen Sätzen. COBOL-Programme sind leicht lesbar, benötigen jedoch viel Platz. COBOL ist eine standardisierte Sprache; sie ist gleich, egal welcher Computer sie benutzt. Sie wird vorwiegend von Großfirmen und der Regierung angewandt.

CPU.
Central Processing Unit (= zentrale Recheneinheit) ist das Herz eines Computers, durch das alle Informationen fließen.

Datei = File.
Sie besteht aus zusammenhängenden Blöcken von Daten oder Instruktionen auf einem externen Speichermedium. Z. B. könnte ein komplettes Programm in einer Datei namens „PROGRAMM.BAS" gespeichert sein. Eine komplette Reihe zusammenhängender Daten (z. B. Prüfregister, Adreßdatei o. ä.) kann in einer Datei gesichert werden.

Dateinamen und -typen = File Names/File Types.
Der Dateiname in CP/M hat drei getrennte Sektionen: ein Name mit bis zu acht Zeichen, gefolgt von einem eindeutigen Typenidentifikator, genannt Dateityp. CP/M lokalisiert die Datei durch Dateinamen und -typ.

Daten = Data.
Daten sind alle Informationen, die ein Computer verarbeitet oder speichert. Ein wichtiger Unterschied besteht zwischen einer Computer-Instruktion und Verarbeitungsdaten. Ein Computer unterscheidet zwischen Daten und Instruktionen gänzlich aus dem Zusammenhang. Ein fehlendes Stück Information kann seltsame Resultate hervorrufen.

Datenträger = Media.
Datenträger wie z. B. Floppy-Disketten oder Bandkassetten sind das Material, auf dem Information gespeichert ist.

Dienstprogramme = Utilities.
Dienstprogramme werden gewöhnlich von vielen Arten von Anwendern für mehr als einen Zweck angewandt. Anders verhält es sich bei Anwenderprogrammen, die i. a. nur eine spezifische Aufgabe erfüllen.

DMA.
Direct Memory Access (= direkter Speicherzugriff) umgeht die CPU, und Information fließt direkt vom und zum Speicher. Während die Plattenlaufwerke eine DMA-Übertragung vornehmen, wartet die CPU einfach auf die Vollendung des Vorganges. Normalerweise muß Information die CPU passieren, wenn sie von Kassette zum Kernspeicher oder umgekehrt fließt. DMA-Übertragungen sind in der Regel schneller als bei Einschluß der CPU.

Dokumentation = Documentation.
Dokumentation umfaßt alle geschriebene Information über Computer- oder Programmausstattung (Software) wie z. B. Systemhandbücher.

Formatisierung = Formatting.
Vor dem Gebrauch einer Diskette wird diese formatisiert, d. h. mit Leerinformation beschrieben. Dabei wird auch geprüft, ob die Diskette gelesen werden kann.

Grundstellung = Reset.
Grundstellung befiehlt dem Computer den Neustart (Restart). Es schaltet ihn aus und dann wieder ein.

Hartplatte = Hard Disk.
Hartplatten sind Datenträger, die in polierten Metalloberflächen oder mehrlagig verdrahtet montiert sind. Die Mehrlagenplatten sind mit magnetischem Oxid beschichtet. In einer Festplatte sind die Lagen permanent im Laufwerk befestigt (und versiegelt), man kann den Datenträger nicht entfernen, ohne das Laufwerk zu ruinieren. In Laufwerken mit austauschbaren Platten befinden sich die Datenträger in herausnehmbaren Plastikmagazinen.

Hex.
Hex, Kurzform für „hexadezimal", bezieht sich auf ein Zahlensystem zur math. Basis 16 mit den Ziffern 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Die

kleinste Computereinheit von Bedeutung (1 Byte = 8 Bit [0 oder 1]7) besteht aus zwei hex Ziffern. So wird aus B„11110000" X„F0". Die Tabelle in Anhang B zeigt die Äquivalenzen zwischen binären, hexadezimalen und dezimalen Ziffern.

Kompilierer = Compiler.
Compiler werden für höhere Programmiersprachen benötigt, die die Instruktionen nicht in der Form übersetzen, in der sie eingegeben werden. Bei einem BASIC-Compiler könnte man z. B. mit Hilfe des Editors ein Programm eingeben und dann kompilieren. Kompilierung besteht aus den folgenden Schritten: zuerst prüft der Computer das Programm auf offensichtliche Fehler, dann übersetzt er die Instruktionen höherer Ordnung in Maschinenbefehle.

Konfiguration = Configure.
Ein Programm zu konfigurieren bedeutet, einige Programm-Parameter auf die jeweilige Peripherie abzustimmen. Ein Programm kann zusammen mit verschiedenen Kombinationen von Terminals, Druckern und anderen Einheiten benutzt werden. Im CP/M ist der Begriff „Konfiguration" normalerweise mit dem BIOS (Basic Input Output System) assoziiert. Um CP/M zu fahren, muß BIOS auf der Anlage installiert sein. Diese Aufgabe überläßt man am besten einem Systemspezialisten. Der Prozeß ist in Kapitel 7 zusammengefaßt.

Laden mit Urlader = Booting/Bootstrap.
Durch Laden wird ein Computer initialisiert, indem ein Betriebssystem von einer Diskette oder einem Band in den internen Kernspeicher kopiert wird.

Laufzeit = Run-Time.
Einige Module werden Laufzeit-Routinen oder -Codes, auch ladefähige Codes genannt.

Lesen = Read.
Lesen bedeutet, ein Stück Information von einer Datei zu holen, also sie dort zu finden und in den Kernspeicher (oder eine andere Einheit) zu übertragen.

Leser = Reader.
Ein Lochstreifenleser liest die von der Stanzeinheit gestanzten Löcher und interpretiert sie gemäß einem vorgegebenen Code.

Locher/Stanzer = Punch.
Eine Stanzeinheit stanzt Löcher in Lochkarten oder -streifen. Bevor es Disketten gab, wurden dadurch Daten oder Programme eingegeben.

Makrodefinition = Macro.
Makro bezieht sich auf Bibliotheksdateien in Assembler-Sprache; es gibt dafür also Makro-Assembler, die die Aufrufe in ihre Einzelinstruktionen auflösen.

Maschinenabhängig = Machine Dependent.
Wenn ein Programm oder eine Einheit maschinenabhängig ist, kann nur mit der entsprechenden Gerätekonstellation gearbeitet werden.

Maschinencode/-sprache/-instruktionen = Machine Code/-Language/-Instructions.
Alle drei gehören zur untersten Ebene von Instruktionen, die also direkt vom Computer verstanden werden.

Minifloppies.
„Minifloppy", eine Handelsmarke der Shugart Associates, bezieht sich auf ihre 5¼-Zoll-Floppy-Disketten. Diese werden inzwischen auch von anderen Herstellern produziert.

Platten-Inhaltsverzeichnis = Directory.
Eine Directory enthält eine Tabelle der Platteninhalte. Darin sind der Name jeder Datei sowie einige andere Informationen gespeichert, die CP/M benutzt, um den physischen Status der Datei auf der Platte und ihren momentanen Gebrauch aufzuzeigen.

Puffer = Buffer.
Ein Puffer ist ein Kernspeicherbereich, der zeitweilig Eingabe von oder Ausgabe zu einer peripheren Computer-Einheit (z. B. Plattenlaufwerk) speichert. Die CPU kann Informationen im Kernspeicher schneller verarbeiten als auf einer langsameren mechanischen Einheit wie Drucker, Plattenlaufwerk, Stanzer oder Tastatur.

RAM.
Random Access Memory (= Direktzugriffs-Speicher) ist der primäre Speicherbereich eines Computers. Bei Stromunterbrechung ist die Information im RAM verloren. Das ist ein Grund, weshalb CP/M jedesmal neu geladen werden muß, wenn das Computer-System eingeschaltet wird.

Register.
Ein Register ist ein interner temporärer Speicherbereich in der CPU. Die 8080-Familie von CPUs (auf der CP/M läuft), besitzt ein Minimum von acht Registern. 8080-CPU-Register sind das A- und Flag- (= Markierungs-), B- und C-, D- und E- sowie H- und L-Register. Die paarweise Aufzählung dieser Register entspricht den 8080-Instruktionen, die sich auf Registerpaare beziehen. Nur Programmierer der Assembler- und der Maschinensprache sind mit Registerinhalten konfrontiert.

ROM.
Vom Read Only Memory (= Festspeicher) kann nur gelesen werden, die darin gespeicherte Information kann nicht geändert werden. Nichtveränderbare Routinen wie der System-Urlader (Bootstrap) oder ein System-Monitor (der einen Computer auf höherer Ebene kontrolliert als CP/M) befinden sich oft im ROM.

Routinen.
Routinen bestehen aus einer Reihe von Modulen (aufeinander bezogene Instruktionen), die spezifische Aufgaben übernehmen. Routinen können Programme umfassen, müssen aber selbst keine sein. Ein Zeichen einzulesen, zu prüfen, ob eine Taste gedrückt war, ein Zeichen auszugeben oder den Motor des Plattenlaufwerks einzuschalten, sind Beispiele für Routinen.

Schrägversetzen = Skewing.
Der Begriff „Skewing" bezieht sich darauf, wie Information physisch auf eine Diskette gespeichert wird. Information wird nicht-angrenzenden Speicherplätzen zugeordnet – das erste Stück Information zu Sektor 1, das zweite zu Sektor 18, das dritte zu Sektor 25 usw. Dieses Verweben von Information wird vollständig von CP/M gehandhabt.

Schreiben = Write.
Information wird auf Diskette oder andere Datenträger einschließlich RAM geschrieben.

Schreibschutzsperre = Write-Protect.
Man kann bei einer Platte mit Schreibschutzsperre keine weitere Information hinzufügen, sondern nur lesen. Die Schreibschutz-Raste einer 8-Zoll-Diskette befindet sich an ihrer Unterseite, bei einer Minifloppy wird sie durch Überkleben der Raste auf der rechten Seite aktiviert. Ebenso schützt das Kommando STAT Disketten und Dateien vor Überschreibung.

Sektoren und Spuren = Sectors and Tracks.
Wenn Information auf ein Kassettenband gespeichert wird, so geschieht dies in Form einer einzelnen Datenspur über die ganze Länge des Bandes. Die Oberfläche einer Diskette dagegen besteht aus einer Anzahl konzentrischer Spuren, von denen jede wiederum in Sektoren unterteilt ist, so daß CP/M zu jedem Punkt der Diskettenoberfläche über die Sektoren- und Spurnummer zugreifen kann (s. a. Abb. 1–3).

Sequentiell = Sequential.
Auf Elemente einer sequentiellen Datei wird in der Reihenfolge ihrer Speicherung zugegriffen; es müssen also vor Element Nr. 10 die neun vorherigen Elemente gelesen werden. Sequentielle Speicherungsmethoden werden angewandt, wenn der Platzbedarf nicht im voraus geschätzt werden kann. Sequentielle Dateien haben einen Dateiende-Merker (EOF – End of File) und können aus Elementen unterschiedlicher Länge (variable Satzlänge) bestehen.

Speicheradresse = Memory Location.
Speicherplätze eines Computers sind durch Adressen definiert. Auf einen spezifischen Speicherplatz kann jeweils ein Computerwort (Byte) an Information gespeichert werden.

Speicherstelle = Byte.
Die Speicherkapazität eines Mikrocomputer-Kernspeichers oder seiner

Platten wird stets als eine Anzahl von Bytes beschrieben. Z. B. kann der Buchstabe „A" oder die Ziffer „1" in einer Speicherstelle dargestellt werden. Zahlen ohne Dezimalpunkt werden gewöhnlich in zwei Bytes gespeichert, während Zahlen mit Dezimalpunkt fünf oder mehr Bytes erfordern können. Die Kernspeichergröße wird normalerweise nicht in Tausenden von Bytes, sondern als eine Anzahl von K-Bytes angegeben. 1 K beträgt 1024 ($= 2^{10}$). Alle Computer sind Binärmaschinen, die in 2er-Einheiten rechnen.

Stapelverarbeitung = Batch Processing.
Bevor Video-Terminals und andere hochentwickelte Einheiten erfunden waren, bekamen die meisten Computer ihre Anweisungen durch Lochkarten. Die Löcher in den Karten repräsentierten Computer-Instruktionen und Daten. Für eine effektive Verarbeitung wurden die Karten in Gruppen (Stapeln) gesammelt. Heutzutage wird der Begriff „Stapelverarbeitung" für jeden Prozeß benutzt, bei dem Instruktionen und Daten (über Terminals, manuelle Locher, optische Leser usw.) gesammelt und die Verarbeitung so lange aufgeschoben wird, bis ein großer „Stapel" akkumuliert ist.

Status.
Computer-Einheiten sind entweder bereit (ready) oder nicht bereit. Wenn der Computer eine Einheit (sagen wir einen Drucker) benutzen will, prüft er ihren Status; er prüft also, ob die Einheit ready ist. Bedauerlicherweise sind die Status-Schemata nicht standardisiert. Ein positiver Stromstoß von einer Einheit kann je nach Hersteller bedeuten, daß sie bereit oder auch nicht bereit ist. CP/M führt automatische Status-Prüfungen durch, ein sauber funktionierendes BIOS vorausgesetzt.

Steuereinheit = Console Device.
Die Steuereinheit sendet und empfängt Eingaben zum und Ausgaben vom Computer. Bei den meisten CP/M-Benutzern ist die Steuereinheit ein Standard-Terminal mit Tastatur und CRT-Datensichtgerät. Bei einigen Systemen können Tastatur und Bildschirm separate Einheiten sein. Einige ältere Systeme benutzen eine Konsolschreibmaschine mit Eingabetastatur, aber diese Tendenz ist seit dem Erscheinen preisgünstiger Datensichtgeräte rückläufig.

Steuerglieder = Drivers.
Ein Driver ist ein kompletter Instruktionssatz, der einen Teil der Gerätschaft kontrolliert. Mit anderen Worten: man benötigt zur Gerätekonfiguration passende Steuerglieder, um ein Plattensystem zu fahren. Die CP/M-Driver sind im BIOS (Basic Input Output System) enthalten.

Übermittlungselement = Modem.
Modulator/Demodulator konvertiert Daten und/oder Computer-Instruktionen in Töne (Frequenzen) und decodiert sie. Telefonleitungen übertragen die modulierten Signale. Verglichen mit Computern sind Modems generell ziemlich langsame Einheiten.

Unterbrechungen = Interrupts.
Eine Unterbrechung ist ein Signal an die CPU, mit der Verarbeitung aufzuhören und eine andere Instruktionsfolge auszuführen.

Ursprung = Source.
Quellcode bezieht sich auf das Originalprogramm in seinem ursprünglichen Zustand. Da Programme gelegentlich kompiliert (in die Maschinen- oder eine effizientere Sprache übersetzt) werden, ist die Unterscheidung zwischen Quell- und Laufzeitprogrammen von Bedeutung. Darüber hinaus beschreibt der Begriff „Source" alle ursprünglichen Daten oder Programme, so daß sich Datensicherung auf die Kopie eines Ursprungs bezieht.

Wahlfrei = Random.
Plattensysteme benutzen zwei Speichermethoden: wahlfreien (direkten) und sequentiellen Zugriff. Bei DA-Dateien (Direct Access) wird durch die Satznummer (Stellung innerhalb einer Gruppe von Sätzen) auf einen Satz zugegriffen. Da ein DA-Satz schnell und akkurat gefunden werden soll, sind Random-Dateien von fester Satzlänge.

Wildcard.
Globale Parameter bleiben während eines Prozesses konstant und können jede beliebige Spezifikation annehmen. Z. B. ist in dem CP/M-Kommando „STATB:*.ASM" „*" ein globaler Parameter enthalten, da jedes beliebige Element diese Position in der Kommandozeile ausfüllen kann.

Z80.
Der Z80-Chip ist eine verbesserte Version des 8080; denn während es volle Kompatibilität aufrechterhält (es kann jede der 8080-Instruktionen genauso ausführen wie der 8080 selbst), besitzt es eine Anzahl zusätzlicher Möglichkeiten. So enthält der Z80 beispielsweise einen doppelten Registersatz gegenüber dem des 8080. Auch ist Indizierung, ein spezieller Typ von Computerinstruktion, erlaubt.

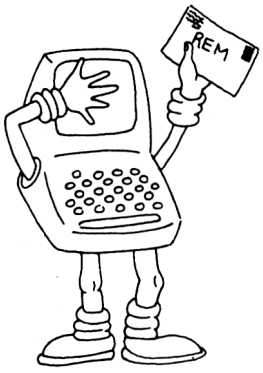
8080.
Der 8080-Mikroprozessor wurde von Intel entworfen. Er gehört zur zweiten Generation von CPU-Chips und damit zur ersten mit einer weitgespannten Konzeption. Er hat acht interne Register: A,B und C,D und E,H und L und Flag (Markierung). Sein Instruktionssatz, d. h. die Instruktionen, die er direkt ausführen kann, ist gegenüber späteren CPU-Versionen etwas eingeschränkt.

Glossarium englisch – deutsch

ASCII
Assembly Language = Assemblersprache
BASIC
Batch Processing = Stapelverarbeitung
Booting/Bootstrap = Laden mit Urlader
Buffer = Puffer
Byte = Speicherstelle
Call = Aufruf
Close = Abschließen
COBOL
Compiler = Kompilierer
Configure = Konfiguration
Console Device = Steuereinheit
CPU
Data = Daten
Directory = Platten-Inhaltsverzeichnis
DMA
Documentation = Dokumentation
Drivers = Steuerglieder
Execute = Ausführung
File = Datei
File Names/Files Types = Dateinamen und -typen
Formatting = Formatierung
Hard Disk = Hartplatte
Hex
Interrupts = Unterbrechungen
Library File = Bibliotheksdatei
Machine Code/- Language/- Instructions = Maschinencode . . .
Machine Dependent = maschinenabhängig
Macro = Makrodefinition
Media = Datenträger
Memory Location = Speicheradresse
Minifloppies = Minidiskette (5¼ Zoll)
Modem = Übermittlungselement
Operating System = Betriebssystem
Punch = Locher/Stanzer
RAM = Random Access Memory
Random = wahlfrei
Read = Lesen
Reader = Leser
Register = interner Speicherbereich
Reset = Grundstellung

ROM = Read Only Memory
Routine = Prozedur, Unterprogramm
Run-Time = Laufzeit
Sectors and Tracks = Sektoren und Spuren
Sequential = Sequentiell
Skewing = Schrägversetzen
Source = Ursprung, Quelle
Statement = Anweisung
Status = Gerätezustand
Tape = Band
Utilities = Dienstprogramme
Wildcard = globaler Parameter
Write = Schreiben
Write-Protect = Schreibschutzsperre

Index



Weitere Bücher zum Thema:

L. Poole

Praktische BASIC-Programme für den IBM Personal Computer

1983. ISBN 3-89028-000-5, 172 Seiten, DM 29,80

Die Anzahl der im Augenblick erhältlichen IBM-PC-Software ist noch sehr begrenzt. Mit diesem Buch stehen nun den Benutzern von IMB-PCs 34 praktische Programme in deutscher Sprache zur Verfügung, die unmittelbar so eingegeben werden können, wie sie im Buch abgebildet sind. Für wenig Geld erhält man damit nützliche Anwendungen für das Büro, für Übungskurse und für zuhause.

A. Osborne/G. Eubanks/M. McNiff

CBASIC Anwenderhandbuch

1984. ISBN 3-89028-006-4, 215 Seiten

Der hier beschriebene populäre BASIC-Compiler für 8080-, Z80- oder 8085-Mikrocomputer ist ebenfalls leicht zu erlernen wie BASIC. Eine einfache und schnelle Programmiersprache, um hochorganisierte, lesbare und modular aufgebaute, strukturierte Programme zu erstellen. Dieses Standardwerk wurde unter Mitarbeit des Erfinders von CBASIC, Gordon Eubanks, verfaßt.

H. Peckham

BASIC für den IBM Personal Computer

1984. ISBN 3-89028-001-3, 348 Seiten

Mit diesem Buch erlernt man leicht und schnell die Programmiersprache BASIC des IBM-PCs. Jedes Kapitel beginnt mit einer kurzen Darstellung der Lernziele. Übungen begleiten die folgenden Lernabschnitte und vermitteln praktische Fähigkeiten im Umgang mit dem Computer. Durch die selbständige Erarbeitung erwirbt der Leser bei einem Minimum an Unterweisung ein solides theoretisches und praktisches Wissen. Das Buch ist für den Kursunterricht ebenso geeignet wie für das Selbststudium.

J. Heilborn/R. Talbott

VC 20 Anwenderhandbuch

1983. ISBN 3-89028-004-8, 388 Seiten

Das Anwenderhandbuch vermittelt alles nötige Wissen für den Umgang mit dem VC20 und seinen Zusatzgeräten. Die VC20-BASIC-Programmiertechnik, der ganze Bereich der Colorgraphik und der Tonerzeugung und sogar Technik und Design eigener elektronischer Unterhaltungsspiele werden genau beschrieben. Das Buch eignet sich sowohl für den Anfänger zur schnellen Einführung als auch für den erfahrenen Anwender als Nachschlagewerk.

A. Fox / D. Franz

BASIC ganz einfach

1984. ISBN 3-89028-002-1, 246 Seiten

Schritt für Schritt wird der Leser durch zahlreiche Übungen und Illustrationen auf humorvolle Art mit BASIC vertraut gemacht. Man benötigt dazu nicht einmal einen Computer und erst recht keine technischen oder mathematischen Vorkenntnisse. Eine unkomplizierte Einführung in die populäre Computersprache für jung und alt.

H. McGilton/R. Morgan

Einführung in das UNIX-System

1984. ISBN 3-89028-003-X, 480 Seiten

Ein Lehrbuch für den Neuling und ein vollständiges Nachschlagewerk für den erfahrenen Anwender. Das Buch stellt die Möglichkeiten des UNIX-Systems anhand aktueller und vollausgetesteter Beispiele des UNIX-Codes vor.

McGraw-Hill
Hamburg

McGraw-Hill Book Co. GmbH · Lademannbogen 136 · D-2000 Hamburg 63 · Tel.: (040) 5382081

CP/M®
Anwenderhandbuch

Hier werden die letzten CP/M-Entwicklungen vorgestellt, einschließlich CP/M™-86, dem Betriebssystem für die auf dem 8086 und 8088 basierenden Mikrocomputer wie z.B. den IBM® Personal Computer. Für Anwender, die die Grundlagen des CP/M®kennenlernen wollen, überbrückt dieser Führer die Kluft zwischen technischen Handbüchern und der eigenen Erfahrung mit Mikrocomputern.

Für Anfänger:

das CP/M® Anwenderhandbuch

- umfaßt alle CP/M®-Kommandos im Detail
- beschreibt Standard-CP/M®-Dienstprogramme
- diskutiert höhere Programmiersprachen und andere Dienstprogramme im CP/M®
- stellt Referenzlisten und Tabellen zur Verfügung
- untersucht, wie das System bei der Anwendung von CP/M® vorgeht.

Für fortgeschrittenere Anwender und Programmierer:

das CP/M® Anwenderhandbuch

- beschreibt detailliert die CP/M™-80- und CP/M™-86-Systemaufrufe und
- stellt CP/M®-Modifikationen für unterschiedliche Computersysteme vor.

Diese Ausgabe schließt eine Darstellung der CP/M®-Ableitungen MP/M II™ und CP/NET® ein.

ISBN 3-89028-005-6

SYBEX

CP/M® Anwenderhandbuch

Thom Hogan

AMSTRAD CPC

MÉMOIRE ÉCRITE
MEMORY ENGRAVED
MEMORIA ESCRITA

https://acpc.me/