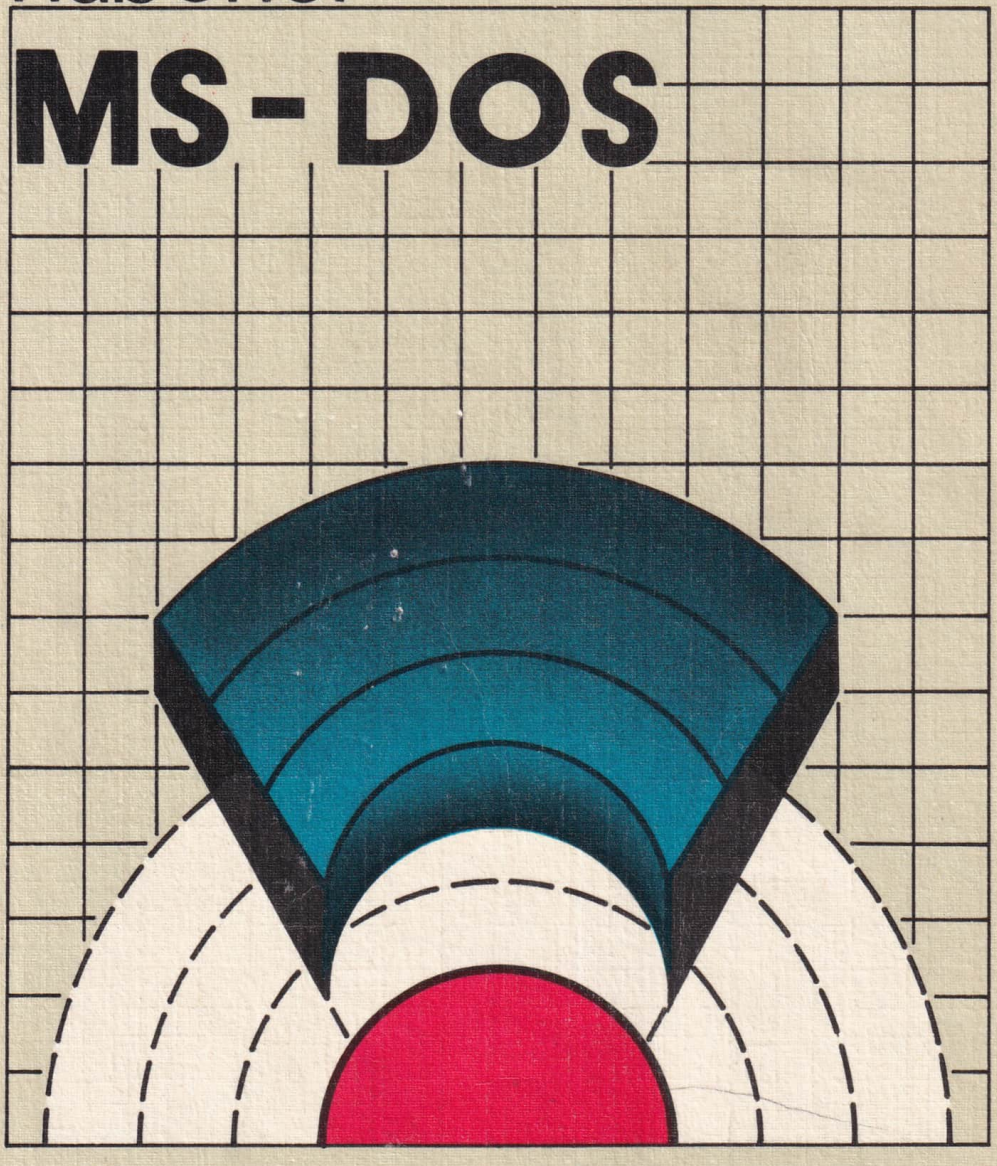
Technische Informatik

Hübener

# MS-DOS



VEB Verlag Technik Berlin

Hübener · MS-DOS

# Technische Informatik

Herausgegeben von
Dr.-Ing. Ludwig Claßen
Dr.-Ing. Dieter Nedo
Dr. rer. nat. Gerhard Paulin

# MS-DOS

Dr. rer. nat. Joachim Hübener

2., unveränderte Auflage

VEB VERLAG TECHNIK BERLIN

Hübener, Joachim:
MS-DOS / Joachim Hübener. – 2., unveränd. Aufl.-
Berlin : Verl. Technik, 1989. – 240 S. : 2 Bilder,
67 Taf.
ISBN 3-341-00707-5

ISBN 3-341-00707-5
ISSN 0863-0860

2., unveränderte Auflage

VT 201 · 3/6011-2 (258)
Printed in the German Democratic Republic
Druck und buchbinderische Verarbeitung: (140) Druckerei Neues Deutschland, Berlin
Lektor: Jürgen Reichenbach
Einbandgestaltung: Gabriele Schwesinger
LSV 3054
Bestellnummer: 554 097 5
02400

# Vorwort

Mit dem Einsatz der PC-Technik hat sich die Informationsverarbeitung grundlegend verändert. Bestimmte Bereiche, wie Büroautomatisierung, Kommunikation, Entwurfsprozesse (CAD/CAM) sowie Planung und Leitung, sind ohne diese Technik nicht mehr vorstellbar. Besonders durch IBM-PC haben Personalcomputer eine sehr große Popularität erreicht. Viele bekannte Computerhersteller bauen sogenannte "kompatible PC" auf 16-Bit-Basis. Das Betriebssystem MS-DOS (Microsoft Disk Operating System), das von IBM als PC-DOS eingeführt wurde und vertrieben wird, hat sich als Standardbetriebssystem durchgesetzt. Ähnlich wie beim Betriebssystem CP/M, das für 8-Bit-Computer als Standard gilt, gibt es eine sehr breite Palette an Anwendungssoftware.

Dieses Buch soll in komprimierter Form eine Einarbeitung in die Möglichkeiten von MS-DOS unterstützen und durch Beschreibung der internen Zusammenhänge das Verständnis für die Arbeitsweise des Systems erleichtern. Bei der Beschreibung wurde der Funktionsumfang der MS-DOS-Version 3.3 berücksichtigt. Ich habe versucht, möglichst viele Informationen für die praktische Arbeit in eine logische Gliederung zu bringen, die von der in den Handbüchern zu MS-DOS gewählten Darstellungsform abweicht. Es ist nicht beabsichtigt, die Systemhandbücher zu ersetzen, sondern sie mit Beispielen und Anwendungslösungen, übersichtlichen Tabellen und anderem Referenzmaterial zu ergänzen. Ich halte es für besonders wichtig, daß der Anwender bei der Lösung seiner Probleme etwas von dem versteht, was im Rechner abläuft. Nur so können evtl. unklare Stellen in der Dokumentation durch eigene Überlegungen geklärt werden.

Nach einer kurzen Einleitung werden im 2. Abschnitt einige Informationen zur PC-Hardware dargestellt, die auch für einen Softwarespezialisten wichtig sind. Abschnitt 3. beschreibt die Möglichkeiten zur Nutzung der Hardwarekomponenten bei der Programmierung. Im Abschnitt 4. wird auf die Struktur von MS-DOS eingegangen und die Nutzung der Dienste des Betriebssytems auf den verschiedenen Programmierebenen (BIOS, DOS) erläutert. Dem Dateisystem ist ein eigener Abschnitt gewidmet, da es eine zentrale Rolle für die Anwendung spielt. Im Abschnitt 6. werden die DOS-Kommandos beschrieben, die von MS-DOS zur Systemnutzung bereitgestellt werden. Der Umfang entspricht der Version 3.3, wobei einige nicht mehr aktuelle Funktionen zur Unterstützung der älteren DOS-Versionen weggelassen wurden. Abschnitt 7. beschreibt die Möglichkeiten zur Stapelverarbeitung, und Abschnitt 8. behandelt Fragen der Entwicklung eigener Programme (Programmierwerkzeuge). Die weiteren Abschnitte erläutern wichtige Ergänzungs- und Anwendungsprogramme für MS-DOS. Neben den Fragen der Bedienung und der internen Funktionen werden damit übersichtsartig auch die gängigen Anwendungssoftwarepakete zur Textverarbeitung, Datenverwal-

tung, Tabellenkalkulation, Rechnerkommunikation und Multi-funktionspakete behandelt. Eine Vollständigkeit ist dabei natürlich nicht möglich und auch nicht angestrebt. Abschnitt 11. enthält viele Programmierbeispiele und praktische Hinweise für die Programmierung. Programmierbeispiele werden in Turbo-Pascal formuliert, das für diese Zwecke als am besten geeignet erscheint. Eine Sammlung von Tabellen und Übersichten im Anhang ist auch für den versierten Programmierer von Interesse.

Das Buch erscheint in der Reihe "Technische Informatik", in der auch weiterführende Titel zur Programmierung und zu wichtigen Anwendungspaketen erscheinen. Mein Dank gilt dem VEB Verlag Technik, insbesondere dem verantwortlichen Lektor, Herrn J. Reichenbach, für die konstruktive Zusammenarbeit und dem Herausgeber, Herrn Dr. D. Nedo, für viele Anregungen und sachliche Kritik. Außerdem danke ich Herrn Dr. Pas und meiner Familie für die Unterstützung bei der Erstellung der Druckvorlage. Ich habe versucht, Fehler nach Möglichkeit zu vermeiden. Sollten sich bei der Fülle des Materials dennoch Fehler eingeschlichen haben, so bin ich für Hinweise über die Verlagsadresse dankbar.

Joachim Hübener

# Inhaltsverzeichnis

# 1. Einleitung

## 1.1. Besonderheiten von PC

Das Betriebssystem MS-DOS der Firma Microsoft wurde 1981 zusammen mit dem IBM-PC eingeführt. Dieser PC war so erfolgreich, daß viele Hersteller sogenannte "kompatible PC" in ihr Fertigungssortiment aufnahmen. MS-DOS war in der Version 1.0 ein relativ einfaches Betriebssystem, das viel Ähnlichkeit mit dem erfolgreichen CP/M hatte. Es wurde ständig weiterentwickelt, wobei auf Kompatibilität besonders geachtet wurde.

Warum hatte der PC mit MS-DOS nun so einen großen Erfolg? Die Grundlage dafür ist sicher die rasante Entwicklung der Mikroelektronik, die eine Produktion in sehr großen Stückzahlen bei kleinem Preis ermöglichte. Dadurch wurde der PC am Arbeitsplatz verfügbar und konnte unmittelbar in den Arbeitsprozeß einbezogen werden. Durch neue Konzepte der Bildschirmsteuerung können auch anspruchsvolle Graphikanwendungen realisiert werden. Viele Softwarehäuser entwickelten leistungsfähige Softwarepakete, die auf dem PC unter MS-DOS lauffähig sind. MS-DOS wurde so zu einem Betriebssystemstandard (Industriestandard) mit einer sehr großen Softwarebibliothek. MS-DOS wird dadurch noch lange eine wichtige Rolle spielen. Der gegenwärtige Trend geht dahin, daß die PC immer leistungsfähiger werden und die Leistung von Graphik-Workstations erreichen. Weiterentwicklungen der Betriebssysteme (OS/2, FlexOS u.a.) gestatten dann sogar die Nutzung des Rechners für mehrere Anwendungen gleichzeitig.

## 1.2. Anwendungsklassen

Das Grundmodell eines PC ist ein allgemein verwendbarer Rechner. Durch das offene Systemkonzept ergeben sich viele Erweiterungsmöglichkeiten und die Anpassungsfähigkeit an spezielle Aufgaben. Nun ist der PC sicher nicht in allen Gebieten gleich gut einsetzbar. Für die Realisierung von Echtzeitsystemen gibt es z.B. besser angepaßte Lösungen in Form von OEM-Rechnern (single board computer) mit Spezialprozessoren. Für die direkte Nutzerkommunikation ist der PC auch in solchen Anwendungen sehr gut geeignet auf Grund ausgebauter Windowtechnik und guter Softwareunterstützung. Die Entwicklung von Industrie-PC geht in diese Richtung. Große Informationssysteme erfordern riesige Datenspeicher mit sehr schnellen Datenkanälen, die z.B. besser durch Groß-

rechner realisiert werden. Einige Anwendungsklassen lassen sich aber sehr gut und besser mit PC als mit anderen Rechnern realisieren. Ohne den Anspruch auf Vollständigkeit seien hier einige Anwendungsgebiete genannt:

- Büroautomatisierung
- Arbeitsplatzcomputer für wissenschaftliche Berechnungen
- Produktionsvorbereitung (CAD/CAM)
- intelligentes Terminal in Informationsnetzen
- Bildverarbeitung
- Textverarbeitung (text processing, desktop publishing)
- Experimenteautomatisierung
- Informationssysteme mit begrenztem Umfang.

Durch ständiges Anwachsen der Leistungsfähigkeit von PC werden die Anwendungsbereiche sicher noch zunehmen.

## 1.3. Logische Gliederung des Buches

Für die Beschreibung in diesem Buch wurde ein Schichtenmodell als Grundlage gewählt (Tafel 1.1). Die Zuordnung der Ebenen zu den einzelnen Abschnitten ist im Bild angegeben. Die Beschreibung beginnt aus didaktischen Gründen bei der untersten Ebene (Hardware). Die weiteren Ebenen sind aus der Struktur von MS-DOS abgeleitet. Unterste Programmierebene ist die direkte Hardwareansteuerung, die durch das Betriebssystem nicht weiter unterstützt wird (Abschn. 3.). Unterste Ebene des Betriebssystems, dessen Struktur im Abschn. 4. dargestellt wird, ist das BIOS (basic input output system), das wie eine Schale um die Hardware liegt und die Geräteansteuerung über festgelegte Schnittstellen gestattet. Die eigentlichen Funktionen des Betriebssystems werden durch das DOS bereitgestellt, das oberhalb des BIOS liegt. Die Funktionen des BIOS und DOS sind in den Abschnitten 4. und 5. beschrieben. Diese Ebenen sind wiederum Basis des Kommandointerpreters COMMAND, auf dessen Funktionen (Kommandos) im Abschn. 6. eingegangen wird. Weitere Ebenen werden durch Kommandofolgen (Abschn. 7.) und eigene Programmentwicklungen (Abschn. 8.) gebildet. Eine Integration verschiedener Programme unter einer einheitlichen Benutzeroberfläche ist durch Zusatzpakete (Abschn. 9.) möglich. Die wichtigsten Anwendungsgebiete werden durch Programmkomplexe unterstützt, die nicht mehr Bestandteil von MS-DOS sind, jedoch den eigentlichen Gewinn für den PC-Einsatz bringen und nur unter MS-DOS bzw. einem kompatiblen System laufen. Einige Pakete werden im Abschn. 10. übersichtsartig dargestellt. Bei der Funktionsbeschreibung in den einzelnen Abschnitten werden nach einer Übersicht die Funktionen in logischen Gruppen dargestellt, die sich an den Erfordernissen der Programmierung für einen Datenkanal

(Gerät, Verzeichnis, Dateiname) orientieren. Im Abschn. 11. sind Beispiele zur Programmierung unter MS-DOS zusammengefaßt. Zum Vergleich von MS-DOS mit weiteren Betriebssystemen, die auf PC nutzbar sind, enthält Abschn. 12. einige Angaben. Am Schluß des Buches sind wichtige Informationen zur Nutzung von MS-DOS in Tabellenform angefügt. Außerdem erleichtert ein ausführliches Sachwörterverzeichnis den Zugang zu einzelnen Funktionen.

**Tafel 1.1. Ebenen der Programmierung unter MS-DOS**

| Programmierebenen | |
|---|---|
| Integrierte Anwendungspakete | Abschnitt 10. |
| Alternative Benutzeroberflächen | Abschnitt 9. |
| Programmierwerkzeuge (eigene Programme) | Abschnitt 8. |
| Stapelverarbeitung (BATCH) | Abschnitt 7. |
| Nutzung des Kommandointerpreters COMMAND | Abschnitt 6. |
| Programmierung mit DOS-Funktionen | Abschnitt 4.,5. |
| Programmierung mit BIOS-Funktionen | Abschnitt 4. |
| Direkte Programmierung der Hardware | Abschnitt 3. |
| Hardwarekomponenten | Abschnitt 2. |

Die Darstellung der Funktionen geschieht nach pragmatischen Gesichtspunkten; die Syntax ist dadurch nicht immer ganz einheitlich. Durch Erläuterungen und Beispiele wird die Anwendungsform verdeutlicht. Sollte etwas unklar bleiben, so findet man sicher durch Experimentieren eine Lösung.

# 2. PC-Hardware

## 2.1. Überblick, PC-Komponenten

Der PC ist für den Anwender/Programmierer eine abgeschlossene Einheit mit bestimmten Leistungsmerkmalen. FÜr ein genaues Verständnis spezieller Funktionen ist es jedoch nicht zu vermeiden, daß man sich auch als Softwareentwickler mit den Bestandteilen des Rechners im Detail beschäftigt. Grundelemente eines PC sind Prozessor, Speicher und spezielle Ergänzungsbausteine, die auf der Systemplatine zu einem Rechnerkern zusammengeschaltet sind. Die Systemplatine enthält Steckplätze (slots), in die Erweiterungskarten eingesetzt werden können, um den Rechner an spezielle Bedürfnisse anzupassen. Einige der Erweiterungskarten, z.B. Bildschirmanschlußkarte, Steuereinheit (controller) für externe Geräte (Diskettenlaufwerk, Festplatte), sind bei fast allen PC vorhanden, werden jedoch mit zunehmendem Integrationsgrad mit auf der Systemplatine untergebracht. Wesenliche Bestandteile des PC sind

- Rechnergrundgerät (Systemplatine, Steckplätze, Netzteil)
- Bildschirm (Monitor)
- Tastatur
- interne oder externe Speichereinheiten (Disketten, Festplatten)
- Graphikperipherie (Tablett, Maus, Plotter usw.)
- Drucker
- Erweiterungskarten.

Die Bestandteile Grundgerät, Monitor und Tastatur sind bei jedem PC vorhanden. Das Grundgerät besteht aus einem Gehäuse, in dem die Systemplatine, die Stromversorgung (Netzteil), zwei Diskettenlaufwerke bzw. ein Diskettenlaufwerk und ein Festplattenlaufwerk untergebracht sind. Im Gehäuse ist bei den meisten PC Platz für insgesamt 8 Erweiterungskarten. Bei größeren PC (z.B. IBM-PC/AT) können weitere Laufwerke untergebracht werden. An der Gehäusevorderseite ist ein kleiner Lautsprecher angebracht, der über einen Tongenerator vom Programm aus angesteuert werden kann. Der Ausbau mit weiteren Baugruppen ist von der Anwendung abhängig. Zum Starten des Systems ist mindestens ein externes Speichermedium (Diskettenlaufwerk oder Festplatte) erforderlich. Der Anschluß der peripheren Baugruppen erfolgt über Erweiterungskarten am Systembus. Durch steigenden Integrationsgrad werden auch bei Erweiterungskarten mehrere I/O-Kanäle zusammengefaßt (z.B. Multifunktionskarte mit Floppy-Controller, serieller/paralleler

Schnittstelle, Uhr usw.). Alle Baugruppen werden über spezielle Port-Adressen und Interrupts durch Software im BIOS (basic input output system) angesprochen. Die Programmierung wird in den entsprechenden Abschnitten beschrieben.

Durch die offene Systemstruktur ist problemlos der Anschluß spezieller Geräte und Baugruppen über Erweiterungskarten möglich. Durch Erweiterungsadapter kann der PC auf mehrere Gehäuse ausgedehnt werden. Beim Ausbau ist natürlich die Leistungsfähigkeit des internen Netzteils und die Busbelastbarkeit zu berücksichtigen.

## 2.2. Systemplatine

Das Konzept der Systemplatine mit genormten Schnittstellen für Erweiterungskarten hat wesentlich zum großen Erfolg des IBM-PC und seiner zahlreichen Kompatiblen beigetragen. Sehr viele Firmen konnten darauf aufbauend spezielle Erweiterungskarten herstellen, so daß es für fast alle PC-Anwendungen angepaßte Lösungen gibt. In diesem Kapitel sollen die wesentlichen Hardwarebausteine kurz dargestellt werden. Für eine ausführliche Behandlung muß auf die Spezialliteratur verwiesen werden.

### 2.2.1. Mikroprozessor

Kernstück des PC ist ein 16-Bit-Mikroprozessor der Firma Intel. Besondere Bedeutung haben der 8088 im IBM-PC/XT und der 80286 im IBM-PC/AT erlangt. Diese Prozessoren sind Vertreter einer ganzen Familie von weitgehend kompatiblen Bausteinen, die sich durch ihre speziellen Leistungsparameter unterscheiden (8088, 8086, 80188, 80186, 80286, 80386). Der 8088 ist intern vollständig softwarekompatibel zum 8086, hat jedoch einen 8-Bit-Datenbus, an den die Bausteine der 8080-Familie angeschlossen werden können. Der Befehlssatz des **8088/8086** weist gegenüber den in Mikrorechnern weit verbreiteten 8-Bit-Prozessoren 8080 und Z80 einige grundlegende Unterschiede auf. Bewährte Konzepte wurden bei der Weiterentwicklung beibehalten. Neu ist die Adressierung von Segmenten mit einer Größe von 64 KByte über spezielle 16-Bit-Segmentregister. Die Anfangsadressen der Segmente sind durch 16 teilbar. Man nennt die dadurch entstehende Adressierungseinheit auch Paragraph (Speichereinheit mit einer Länge von 16 Byte). Der physische (reale) Adreßraum ist auf 1 MByte begrenzt. Die für die physische Adresse notwendigen 20 Adreßleitungen sind aus der CPU herausgeführt und bilden den Adreßbus. Für den Datentransport sind beim 8088 8 Leitungen (8 Bit) und beim 8086 16 Leitungen (16 Bit) vorhanden. Adreß- und Datenleitungen werden z.T. gemeinsam

(zeitmultiplex) genutzt.

Die Prozessoren **80186** und **80188** sind Weiterentwicklungen der Grundtypen, wobei durch höheren Integrationsgrad eine Geschwindigkeitssteigerung und die Unterbringung von sonst zusätzlich erforderlichen Bauelementen auf dem Prozessorchip möglich wurden. Ein qualitativer Sprung erfolgte mit dem im IBM-PC/AT eingesetzten Baustein **80286**. Wesentlich neue Eigenschaften sind

- 3- bis 5fache Leistungssteigerung gegenüber 8086
- aufwärtskompatibler Befehlssatz des 8086
- Unterstützung von Multi-Tasking
- Speicherschutzmöglichkeiten
- 1 GByte virtueller Adreßraum
- 16 MByte physischer Adreßraum (24 Adreßleitungen).

Der 80286 besitzt einen 8086-Modus mit realer Adressierung. In diesem Modus kann das MS-DOS problemlos eingesetzt werden. Der virtuelle Modus (protected virtual address mode) mit allen erweiterten Möglichkeiten des 80286 kann im MS-DOS (Version 3.3) nicht ausgenutzt werden. Dazu wäre eine vollständige Neukonzeption von MS-DOS erforderlich. Mit der Einführung der neuen IBM-Rechnerfamilie Personal Systems/2 wird darum ein neues Betriebssystem MS-OS/2 bzw. OS/2 eingesetzt. Der alte Betriebssystemstandard MS-DOS wird als eine spezielle Variante weiterhin unterstützt.

Eine konsequente Weiterentwicklung der Prozessorlinie ist der **80386** der Firma Intel. Hier gelten die gleichen Einschränkungen für das Standardsystem MS-DOS. Zusätzliche Neuerungen sind vor allem durch die interne 32-Bit-Struktur bedingt. Für Anwender von MS-DOS ist der 32-Bit-PC lediglich ein sehr schneller AT. Auf die Architektur des 80386 wird in diesem Buch nicht eingegangen.

### 2.2.2. Arithmetik-Coprozessor

Alle genannten Prozessoren können wahlweise mit einem Arithmetikprozessor (8087, 80287, 80387) zusammenarbeiten. Durch den Einsatz dieser Coprozessoren wird die Ausführung arithmetischer Operationen mit hoher Genauigkeit wesentlich beschleunigt (Faktor 10 bis 50). Das Zusammenspiel der Prozessoren wird durch spezielle Befehle weitgehend automatisch abgewickelt. Compiler für höhere Programmiersprachen gestatten zumeist die Generierung dieser Befehle, so daß die direkte Programmierung des Coprozessors nur sehr selten erfolgen muß. Die Befehlsliste ist im technischen Handbuch zu finden. Die Verarbeitung der speziellen Formate erfolgt in 8 Registern mit einer Breite von 80 Bit. Neben einfachen arithmetischen Operationen sind auch kompliziertere Funktionen (z.B. trigonometrische Funktionen) realisiert. Für Anwendungen

mit hohem Rechenaufwand (z.B. Graphikarbeitsplätze) ist der Einbau des Coprozessors unbedingt zu empfehlen.

### 2.2.3. Bus

Der Mikrorechner ist zwar der wichtigste Baustein des PC; ohne zusätzliche Spezialbausteine ist er jedoch nicht funktionsfähig. Die Verbindung der Bausteine erfolgt über den **Bus**, der im wesentlichen durch die Anschlußleitungen der CPU gebildet wird. Der Bus stellt damit einen zentralen Teil der Systemplatine dar. Die Anschlußbedingungen der einzelnen Bausteine sind aufeinander abgestimmt bzw. werden durch spezielle Schaltungen an den Bus angepaßt. Beim Datentransfer werden alle Daten über den Bus geleitet. Eine spezielle Bussteuerung sorgt dafür, daß der Ablauf dabei korrekt ist. Nach der Funktion lassen sich die Busleitungen in 4 Gruppen einteilen:

- Adreßbus
- Datenbus
- Kontrollbus
- Versorgungsleitungen.

Der **Adreßbus** dient zur Auswahl des Zieles für einen Datentransfer (z.B. Ausgabe-Port, Speicheradresse). Beim PC/XT sind 20 Adreßleitungen und beim PC/AT 24 Adreßleitungen vorhanden. Damit lassen sich 1 MByte bzw. 16 MByte Adressen ansteuern.

Der **Datenbus** ist der eigentliche Datenweg. Beim Prozessor 8088 sind 8 Datenleitungen, beim 8086 und 80286 16 Datenleitungen vorhanden. Damit lassen sich parallel 8 Bit (1 Byte) bzw. 16 Bit (1 Wort) übertragen. Da der 8088 intern ebenfalls ein 16-Bit-Prozessor ist, müssen bei vielen Operationen zwei Datenübertragungen nacheinander stattfinden. Dadurch ist der 8088 entsprechend langsam gegenüber den echten 16-Bit-Prozessoren.

Der **Kontrollbus** vereinigt Unterbrechungs- und Bussteuerleitungen und realisiert damit das zeitlich korrekte Verhalten des Rechners.

Die **Versorgungsleitungen** versorgen die Bausteine mit den erforderlichen Arbeitsspannungen.

### 2.2.4. Speicher

Zur Speicherung von Daten im Rechner werden RAM- und ROM-Speicherbausteine eingesetzt. Jede Speicherstelle von 1 Byte (8 Bit) wird eindeutig mit einer Adresse angesprochen. Der verwendete Bausteintyp bestimmt die Anzahl der notwendigen Speicherschaltkreise und den Platzbedarf auf der Systemplatine. Die

Adressenaufteilung im Adreßraum der CPU ist festgelegt (vgl. Tafel 2.1). Die Adressen sind zu Blöcken von 64 KByte zusammengefaßt und werden durch das erste Zeichen der Adresse in hexadezimaler Form (0...9, A...F) gekennzeichnet. Die ersten 10 Blöcke (640 KByte) sind für RAM-Speicher vorgesehen. Im 0-Block sind Tabellen des Betriebssystems gespeichert (vgl. Abschn. 4.). Die Speicherblöcke im Adreßbereich oberhalb 640 Kbyte sind meistens nur teilweise installiert. Es existieren Varianten zur Erweiterung des nutzbaren Speichers über die 640-KByte-Grenze hinaus (z.B. EMS von Lotus/Intel/Microsoft), deren Erläuterung hier jedoch zu weit führen würde. Die Möglichkeit zur Einbindung in MS-DOS wird in der entsprechenden Dokumentation zum Erweiterungsspeicher beschrieben.

**Tafel 2.1. Verwendung der Speicherblöcke im PC**

| Segment-adresse | Verwendung |
|---|---|
| F000 | ROM (ROM-BIOS, ROM-BASIC, Diagnose) |
| E000 | Zusatz-ROM |
| D000 | Zusatz-ROM |
| C000 | BIOS-Erweiterungen (z.B. Festplatte) |
| B000 | Bildschirmspeicher |
| A000 | Bildschirmspeicher (Erweiterung) |
| 9000 | RAM (640 KByte) |
| 8000 | RAM (576 KByte) |
| 7000 | RAM (512 KByte) |
| 6000 | RAM (448 Kbyte) |
| 5000 | RAM (384 KByte) |
| 4000 | RAM (320 KByte) |
| 3000 | RAM (256 KByte) |
| 2000 | RAM (192 KByte) |
| 1000 | RAM (128 KByte) |
| 0000 | RAM (64 KByte: Tabellen,BIOS,DOS,COMMAND) |

### 2.2.5. Zusatzbausteine

Auf der Systemplatine sind noch weitere wichtige Bausteine vorhanden, die aber für die Programmierung nur von geringer Bedeutung sind. Sie werden hier nur aufgezählt (Tafel 2.2). Eine genaue Beschreibung ist in den technischen Handbüchern zu finden.

Einige der Bausteine können je nach PC-Typ auch auf Erweiterungskarten untergebracht sein. Die Programmierung dieser Bausteine erfolgt normalerweise durch das BIOS. Direkte Programmierung sollte nur in Ausnahmefällen erfolgen, da keine Kompatibilität über alle PC-Typen gesichert ist. Außerdem besteht die Gefahr, die Arbeit von MS-DOS zu stören.

**Tafel 2.2. Spezielle Bausteine und deren Verwendung**

| Baustein | Bezeichnung | Verwendung |
|---|---|---|
| 8259A (PIC) | Interrupt-Controller | Überwachung ext. Interrupts |
| 8284A | Takt-generator | Erzeugung des CPU-Taktes |
| 8253 (PIT) | Zeitgeber Timer | Zeitsignale, Tonerzeugung |
| 8255A (PPI) | Peripherie-Interface | Steuerung spezieller Ports, z.B. Lautsprecher, Kassette |
| 8237A (DMA) | DMA-Controller | direkter Speicherzugriff zur Ein- und Ausgabe |
| 6845 (CRT) | CRT-Controller | Bildschirmansteuerung |
| PD765 (FDC) | Floppy-Controller | Diskettenansteuerung |
| 8250 | COM-Port | serielles Interface |

## 2.3. Bildschirm und Bildschirmadapter

Der **Bildschirm oder Monitor** ist ein monochromes oder farbiges Ausgabemedium für Texte oder Graphiken mit unterschiedlicher Darstellungsqualität. Es werden vier Grundtypen unterschieden:

**Monochromer Direktsignalmonitor:** Zeichen werden direkt an der programmierten Position ausgegeben. Einfache Graphiken lassen sich mit vorgegebenen speziellen Zeichen generieren. Die Ansteuerung erfolgt durch den Monochromadapter. Die Textdarstellung hat eine sehr gute Qualität. Die Ansteuerung von Pixeln ist nur über Spezialkarten (z.B. MGA- oder Hercules-Karte) möglich.

**Monochromer Mischsignalmonitor:** Die Ansteuerung erfolgt durch den Farbgraphikadapter (CGA) am Mischsignalausgang (composite video). Dieser Monitortyp hat eine sehr weite Verbreitung gefunden. Farbdarstellungen sind nicht möglich. Oftmals ist die Darstellung der Farbinformation in verschiedenen Graustufen realisiert.

**Farbiger Mischsignalmonitor:** Dieser Monitor wird wie der monochrome Monitortyp am CGA-Mischsignalausgang angeschlossen. Der Monitor entspricht einem Farbfernsehgerät mit HF-Adapter. Die Qualität von Texten ist nicht besonders gut.

**RGB-Farbmonitor:** Für jede Farbe (RGB Rot-Grün-Blau) existiert eine eigene Leitung. Der Anschluß erfolgt am RGB-Ausgang des CGA. Einige Monitortypen haben eine zusätzliche

Leitung für das Intensitätssignal (IRGB). Die Text- und Graphikdarstellung ist sehr gut. Die erreichbare Auflösung (Anzahl der Pixel horizontal/vertikal) ist von der Frequenz abhängig, mit der die Strahlsteuerung erfolgt. Für die EGA-Bildschirmkarte sind nur spezielle Monitore mit einer höheren Frequenz (z.B. NEC-MultiSync) geeignet.

Die Ansteuerung des Monitors erfolgt über einen **Bildschirmadapter**, der als Karte auf die Systemplatine gesteckt wird oder direkt in die Systemplatine integriert ist. Die in Tafel 2.3 genannten Adapter haben sich als Standard herausgebildet. Spezielle Adapter existieren z.B. für besonders hohe Graphikauflösung.

**Tafel 2.3. Bildschirmadapter**

| Kennzeichen | Name (engl.) | max. Auflösung Text | Graphik |
|---|---|---|---|
| MDA | monochrom display adapter | 80*25 | - |
| MGA | monochrom graphic adapter (Hercules) | 80*25 | 720*348 |
| CGA | color graphic adapter | 80*25 | 320*200<br>640*200 |
| EGA | enhanced graphic adapter | 80*25<br>80*43 | 640*350 |
| MCGA | multi color graphic adapter (PS/2-30) | 80*25 | 640*480 |
| VGA | video-graphic-array (PS/2 Standard) | 80*25 | 640*480 |

Die Angaben zur Auflösung betreffen nur typische Werte. Viele Adapter kennen verschiedene Modi, die durch Programme eingestellt werden können.

Die Bildschirmausgabe erfolgt im Text- oder Graphikmodus. Im Textmodus werden komplette Zeichen des aktuellen Alphabets (Zahlen, Buchstaben, Sonderzeichen, Graphiksymbole) in einer Zeichenposition (Zeile, Spalte) dargestellt. Im Graphikmodus wird der Bildschirm in Form von Punkten (Pixeln) angesteuert.

Die Bildschirmansteuerung erfolgt auf Hardwareebene über Speicherzugriffe und Ports. In der Regel ist eine direkte Hardwareansteuerung nicht sinnvoll, weil damit keine Programmportabilität gewährleistet ist. Zur Programmierung sollte man auf die

BIOS- bzw. DOS-Funktionen zurückgreifen. Genaue Informationen für die direkte Bildschirmprogrammierung findet man im technischen Handbuch des PC bzw. in den Unterlagen zum Adapter.

## 2.4. Tastatur

Die Tastatur ist das Haupteingabegerät des PC. Das Tastaturfeld ist nicht bei allen PC gleich; es gibt aber eine weitgehende Ähnlichkeit, so daß hier auf die Unterschiede nicht besonders eingegangen wird. Die PC-Tastatur wird über ein flexibles Kabel an das Grundgerät mit einem Diodenstecker angeschlossen. Die Elektronik erhält die Versorgungsspannung vom Grundgerät. Der eingebaute Prozessor (z.B. Intel 8048 oder 8082) stellt die logische Verbindung zwischen den Tasten und dem BIOS her. Jeder Tastendruck und jedes Loslassen einer Taste erzeugt ein Interruptsignal, auf das im BIOS eine Routine reagiert (Tastaturinterrupt INT 9). Als Information wird ein Tastencode (Scan-Code) am Port 96 (hex 60) bereitgestellt, aus dem die betätigte Taste eindeutig ermittelt werden kann. Vom Betriebssystem werden die Unterschiede der einzelnen Tastaturmodelle berücksichtigt.

Die für den Original-PC gültige Zuordnung zwischen Taste und Scan-Code ist in Tafel 2.4 enthalten.

**Tafel 2.4. Zuordnung zwischen Tasten und Scan-Code**

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reihe 1 | Esc | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | - | = | <- |
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| Reihe 2 | Tab | Q | W | E | R | T | Y | U | I | O | P | [ | ] | CR |
| | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 |
| Reihe 3 | Ctrl | A | S | D | F | G | H | J | K | L | ; | ' | ` | |
| | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | |
| Reihe 4 | Sh-L | \ | Z | X | C | V | B | N | M | , | . | / | Sh-R | * |
| | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 51 | 53 | 54 | 55 |
| Reihe 5 | Alt | | | | Leertaste | | | | | | | | Caps-Lock | |
| | 56 | | | | | | 57 | | | | | | 58 | |
| Funkt.- | F1 | F2 | F3 | F4 | F5 | F6 | F7 | F8 | F9 | F10 | Ins | Del | Num | Scr |
| Tasten | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 82 | 83 | 69 | 70 |
| Num.-Block | | | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | - | + |
| (Cursor) | | | | | 79 | 80 | 81 | 75 | 76 | 77 | 71 | 72 | 73 | 74 | 78 |

## 2.5. Disketten

Die Informationsspeicherung erfolgt in der magnetisierbaren Schicht einer rotierenden Diskette, die als Minidiskette (5 1/4 Zoll) oder Mikrodiskette (3 1/2 Zoll) bezeichnet wird. Beim PC werden standardmäßig die Minidisketten benutzt. Zunehmend finden auch die Mikrodisketten Anwendung. Die Oberfläche wird durch den Schreib-Lese-Kopf abgetastet, der durch den Zugriffsmechanismus in definierte Positionen gebracht werden kann. Durch die Anzahl der Positionen werden konzentrische Kreise auf der Oberfläche adressiert, die Spuren (Tracks) genannt werden. Spur 0 ist die außenliegende Spur. Es haben sich Laufwerke mit 40 und 80 Spuren durchgesetzt. Disketten lassen sich 1- oder 2seitig beschreiben, je nachdem, ob das Laufwerk mit 1 oder 2 Schreib-Lese-Köpfen ausgestattet ist.

In jeder Spur sind mehrere Sektoren untergebracht, die jeweils mit einem Zugriff gelesen bzw. geschrieben werden. Als Sektorlänge wird bei MS-DOS standardmäßig 512 Byte gewählt. Gewöhnlich sind beim PC/XT 9 Sektoren auf einer Spur gespeichert. Beim PC/AT werden spezielle HD-Disketten (HD high-density) eingesetzt, die eine höhere Speicherdichte erlauben und mit einer höheren Umdrehung laufen. HD-Disketten sind in normalen Laufwerken nicht verwendbar. Die HD-Laufwerke lassen sich auf den Normalmodus einstellen. Dadurch sind auch Standarddisketten in HD-Laufwerken verwendbar.

In Tafel 2.5 sind die verschiedenen Formate zusammengestellt.

**Tafel 2.5. Diskettenformate**

| Format | Dichte | Seiten | Spuren | Sektoren | Kapazität |
|---|---|---|---|---|---|
| S8 | 48 tpi | 1 | 40 | 8 | 160 KByte |
| S9 | 48 tpi | 1 | 40 | 9 | 180 KByte |
| D8 | 48 tpi | 2 | 40 | 8 | 320 KByte |
| D9 | 48 tpi | 2 | 40 | 9 | 360 KByte |
| QD9 | 96 tpi | 2 | 80 | 9 | 720 KByte |
| QD15 | HD | 2 | 80 | 15 | 1200 KByte |

Der Anschluß der Diskettenlaufwerke an den Bus erfolgt über den Diskettenkontroller, der im wesentlichen einen DMA-Betrieb mit dem Speicher realisiert und durch das BIOS gesteuert wird. Die Programmierung sollte nicht unterhalb des BIOS-Niveau erfolgen.

Die Einteilung der Diskettenoberfläche erfolgt durch das FORMAT-Kommando unter Nutzung des BIOS. Die dazu erforderlichen Programmierhinweise sind im Abschn. 4.3.2. zu finden.

## 2.6. Festplatte

Die Festplatte ist ein Speichermedium, das im Prinzip ähnlich wie die Diskette arbeitet. Zur Erhöhung der Speicherkapazität sind mehrere Scheiben übereinander angeordnet und der Positioniermechanismus mit einer entsprechenden Anzahl Köpfen ausgestattet. Die Scheiben können nicht ausgewechselt werden. Durch den dadurch stabileren mechanischen Aufbau wird eine wesentlich größere Umdrehungsgeschwindigkeit möglich und die Zugriffszeit sehr klein. Die in einer Position des Zugriffsarms übereinander liegenden Spuren nennt man Zylinder.

Die physische Struktur der Festplatte wird durch ein spezielles Formatierprogramm eingerichtet. Normalerweise werden die Festplatten vom Hersteller formatiert, der dabei auch die Überprüfung auf defekte Spuren vornimmt und eine Kennzeichnung der Spuren auf der Platte als unbrauchbar einträgt. Eine erneute Formatierung der Platte ist möglich, sollte aber mit großer Vorsicht erfolgen, da sonst alle gespeicherten Daten verlorengehen. Auf Grund der großen Speicherkapazität wurde eine Unterteilung der Platte in bis zu 4 Bereiche (partitions) vorgesehen. In den Bereichen können z.B. verschiedene Betriebssysteme untergebracht sein oder auch eine physische Platte als aus mehreren logischen Laufwerken bestehend angesehen werden. Weitere Angaben dazu in den Abschnitten 4.3.2. und 6.3. In Tafel 2.6 sind einige charakteristische Kenngrößen für Festplatten aufgeführt. Aus den technischen Unterlagen zur Festplatte kann man die notwendigen Angaben entnehmen. Bei einigen Festplatten-Verwaltungsprogrammen ist die Angabe einer Kennung (type) möglich, mit der eine bestimmte Parameterkombination ausgewählt wird. Die Vielfalt ist bei Festplatten sehr groß.

Der Anschluß der Festplatte erfolgt über einen speziellen Festplattenkontroller, der im DMA-Betrieb einen sehr schnellen Datentransfer realisiert. Die Programmierung erfolgt über das BIOS.

**Tafel 2.6. Festplattenformate (Beispiele)**

| Format | Seiten | Sektoren | Zylinder | Kapazität |
|---|---|---|---|---|
| XT | 4 | 17 | 306 | 10 MByte |
| XT | 8 | 17 | 306 | 20 MByte |
| AT | 4 | 17 | 615 | 20 MByte |

## 2.7. Drucker

Ein sehr wichtiger Bestandteil des PC ist der Drucker. Es würde den Rahmen des Buches sprengen, auf die verschiedenen Typen einzugehen. Einige Hinweise müssen hier genügen.

Es werden hauptsächlich zeichenweise arbeitende Drucker eingesetzt, die über eine parallele oder serielle Schnittstelle an den Rechner angeschlossen werden. Je nach Art der Zeichengenerierung unterscheidet man Typenraddrucker mit einem festen Zeichenvorrat und Matrixdrucker, die beliebige Zeichen generieren können und dadurch für graphische Ausgaben geeignet sind. Ebenfalls für Graphikausgaben geeignet sind die relativ teuren Laserdrucker mit ausgezeichneter Ausgabequalität. Die Programmierung des Druckers erfolgt über das BIOS.

An einen PC können mehrere Drucker angeschlossen werden, die z.B. mit verschiedenen Papiersorten arbeiten.

## 2.8. Erweiterungen

Das offene Konzept des PC läßt sehr viele Erweiterungen zu. Die Anzahl von Standardschnittstellen ist zwar beschränkt durch die Anzahl von Erweiterungsplätzen; in einem gekoppelten Zusatzgefäß oder über spezielle Schnittstellen lassen sich jedoch weitere Baugruppen unterbringen und damit anspruchsvolle Anwendungslösungen realisieren. Einige allgemeine Erweiterungen sollen hier genannt werden:

- Adapter für spezielle hochauflösende Graphik
- Maus, Tablett, Joystick für Cursorpositionierung
- Scanner für Bildaufnahme
- Netzadapter für LAN und WAN
- Prozeßinterfaces
- Adapter für externe Speicher, z.B. Streamer-Tape
- EPROM-Programmiereinheit.

Die Programmierunterstützung wird meistens durch den Produzenten der Erweiterung mitgeliefert. Die Einbindung in das Betriebssystem erfolgt über spezielle Gerätetreiber oder durch eigene Schnittstellengestaltung. Die durch das Betriebssystem gebotenen Möglichkeiten dazu werden im Abschn. 4.2. beschrieben. Die Erstellung eigener Treiberprogramme setzt eine genaue Kenntnis der Gerätefunktionen und der Treiberschnittstelle voraus.

## 2.9. Zusammenspiel der Komponenten

Die in den vorhergehenden Abschnitten genannten Komponenten werden zu einem Rechner zusammengeschaltet und ergeben eine funktionsfähige Einheit. Dazu ist ein Informationsaustausch erforderlich, der durch spezielle Hardwarefunktionen und durch ein Programm gesteuert wird. Die für die Programmierung wichtigen Aspekte der Hardware sind im Abschn. 3.1. (Programmiermodell) kurz zusammengestellt. Hier sollen nur einige Hardwareaspekte genannt werden. Detailierte Angaben dazu findet man in den Datensammlungen zu Prozessorschaltkreisen bzw. in Büchern zur Hardware des 8086.

Der Mikroprozessor kann ein in Form von **Prozessorbefehlen** im Speicher stehende Programm abarbeiten. Die interne Darstellung des Programms wird durch einen Assembler oder Compiler erzeugt. Die im Speicher abgelegten Daten werden durch Angaben zur **Speicheradressierung** ausgewählt. Beim Zugriff ist das **Datenformat** eine wichtige Kenngröβe. Es gibt an, wie lang das Datenfeld ist und wie es zu interpretieren ist. Zur Adressierung werden Prozessorregister verwendet. Die **Adressierung der Geräte** erfolgt über Portadressen, die nicht mit Speicheradressen identisch sind. Eine wichtige Eigenschaft des Prozessors für die Realisierung von Betriebssystemfunktionen ist die Möglichkeit der **Unterbrechung** des Programmablaufs durch Software oder Hardware.

Die Kenntnis der internen Formate des Programms ist nur in Sonderfällen nötig. Zur Unterstützung der Fehlersuche auf der Hardwareebene sollte man sich eines Debuggers (Fehlersuchprogramm) bedienen. Einige PC (z.B. von ZENITH) enthalten auf EPROM einen Monitor, um die Hardware direkt anzusprechen.

## 2.10. Inbetriebnahme

Zu jedem Rechner wird eine Vorschrift zur Inbetriebnahme mitgeliefert, die aufmerksam gelesen werden sollte. Falls der Rechner aus separaten Baugruppen selbst konfiguriert wird, ist auf den Rechnergrundtyp (XT oder AT), durch den der Bus festgelegt wird, und auf spezielle Parameter (z.B. Taktfrequenz) zu achten. Komplettgeräte lassen sich problemlos zusammenstecken und laufen meistens sofort. Beim mehrfachen Ein- und Ausschalten sollte immer der Stillstand der evtl. vorhandenen Festplatte abgewartet werden und der Verschluβ des Diskettenlaufwerks geöffnet sein. Keinesfalls sollte man Steckverbindungen im eingeschalteten Zustand verändern. Wenn man den Rechner transportieren will, so ist es angebracht, den Zugriffsarm der Festplatte in eine Parkposition zu stellen. Für diesen Zweck gibt es kleine Zusatzprogramme (z.B. PARK, SHUTDOWN o.ä.), die auch evtl. andere "Aufräumarbeiten" vor dem Abschalten vornehmen können.

# 3. PC-Software

In diesem Abschnitt soll eine Übersicht über die PC-Software gegeben werden. Im Abschn. 3.1. wird das Programmiermodell erläutert, das alle für die Programmierung wichtigen Aspekte des PC zusammenfaßt. Abschn. 3.2. gibt eine kurze Einführung in die Komponenten der grundlegenden Software des PC. In den Abschnitten 3.3. und 3.4. beginnen dann konkrete Schritte in die Welt von MS-DOS.

## 3.1. Programmiermodell

Der Programmierer sieht die Hardware mit anderen Augen an als der Elektroniker. Die Struktur des Rechners ist dabei weniger von Interesse. Für das Verständnis der Arbeitsweise sind folgende Aspekte wichtig:

- Prozessorbefehle (Befehlsliste)
- Speicheradressierung
- Datenformate
- Geräteadressierung (Ports)
- Unterbrechungen (Interrupts).

### 3.1.1. Befehlsliste

Der Mikroprozessor ist ein programmierbarer Baustein, der das in Form von Prozessorbefehlen im Speicher stehende Programm abarbeiten kann. Die interne Darstellung des Programms wird durch einen Assembler oder Compiler erzeugt. Die Befehle des 8086 sind in Tafel 3.1 enthalten. Die im 80286 zusätzlich vorhandenen, jedoch im MS-DOS (Version 3.3) nicht verwendeten Befehle sind nicht mit aufgeführt. Die Tabelle ist nur als Übersicht gedacht, um z.B. beim Programm DEBUG den Sinn der Assemblerbefehle zu verstehen. Zur Assemblerprogrammierung muß auf die Spezialliteratur verwiesen werden. Moderne Softwareentwicklung geht zunehmend dazu über, auch Systemsoftware in einer höheren Sprache (z.B. C, Modula-2, Pascal) zu programmieren. Neue Compiler gestatten auch die Lösung hardwarenaher Probleme.

Die interne Darstellung von Befehlen und Daten erfolgt in binärer Form. Die Anzeige des Speicherinhalts auf dem Bildschirm und beim Drucken wird in hexadezimaler Form durchgeführt. Bei der Anwendung von Fehlersuchprogrammen wird meist auch die Anzeige in Befehlsform (Assemblerbefehle) und als Zahlen und Text unterstützt.

**Tafel 3.1. Befehlsliste des 8086/8088**

| GENERAL PURPOSE | |
|---|---|
| **MOV** | move |
| **PUSH** | push word onto stack |
| **POP** | pop word off stack |
| **XCHG** | exchange |
| **XLAT** | translate |
| **INPUT/OUTPUT** | |
| **IN** | input |
| **OUT** | output |
| **ADDRESS OBJECT** | |
| **LEA** | load effektive address |
| **LDS** | load pointer using DS |
| **LES** | load pointer using ES |
| **FLAG TRANSFER** | |
| **LAHF** | load AH from flags |
| **SAHF** | store AH in flags |
| **PUSHF** | push flags onto stack |
| **POPF** | pop flags off stack |

| STRINGS | |
|---|---|
| **MOVS** | move string |
| **CMPS** | compare string |
| **SCAS** | scan string |
| **LODS** | load string |
| **STOS** | store string |
| **REP** | repeat |

| FLAG OPERATIONS | |
|---|---|
| **STC** | set carry flag |
| **CLC** | clear carry flag |
| **CMC** | complement carry flag |
| **STD** | set direction flag |
| **CLD** | clear directin flag |
| **STI** | set interrupt enable |
| **CLI** | clear interrupt enable |
| **EXTERNAL SYNCHRONISATION** | |
| **HLT** | halt until interrupt |
| **WAIT** | wait for busy not activ |
| **ESC** | escape to extension |
| **LOCK** | lock bus during next |
| **NOP** | no operation |

| ADDITION | |
|---|---|
| **ADD** | add |
| **ADC** | add with carry |
| **INC** | increment by 1 |
| **AAA** | ASCII adjust for ADD |
| **DAA** | decimal adjust for ADD |
| **SUBTRACTION** | |
| **SUB** | subtract |
| **SBB** | subtract with borrow |
| **DEC** | decrement by 1 |
| **NEG** | negate |
| **CMP** | compare |
| **AAS** | ASCII adjust for SUB |
| **DAS** | decimal adjust for SUB |
| **MULTIPLICATION** | |
| **MUL** | multiply unsigned |
| **IMUL** | integer multiply |
| **AAM** | ASCII adjust for MUL |
| **DIVISION** | |
| **DIV** | divide |
| **IDIV** | integer divide |
| **AAD** | ASCII adjust for DIV |
| **CBW** | convert byte to word |
| **CWD** | convert word to double |

| LOGICALS | |
|---|---|
| **NOT** | not |
| **AND** | and |
| **OR** | inclusiv OR |
| **XOR** | exclusive OR |
| **TEST** | test |
| **SHIFTS** | |
| **SHL** | shift logical left |
| **SAL** | shift arithm. left |
| **SHR** | shift logical right |
| **SAR** | shift arithm. right |
| **ROTATES** | |
| **ROL** | rotate left |
| **RCL** | rotate carry left |
| **ROR** | rotate right |
| **RCR** | rotate carry right |

**Tafel 3.1. Befehlsliste des 8086/8088 (Fortsetzung)**

| CONDITIONAL TRANSFER | |
|---|---|
| **JA/JNBE** | jump if above |
| **JAE/JNB** | jump if above or equ |
| **JB/JNAE** | jump if below |
| **JBE/JNA** | jump if below or equ |
| **JC** | jump if carry |
| **JE/JZ** | jump if equal/zero |
| **JG/JNLE** | jump if greater |
| **JGE/JNL** | jump if greater equ |
| **JL/JNGE** | jump if less |
| **JLE/JNG** | jump if less or equal |
| **JNC** | jump if not carry |
| **JNE/JNZ** | jump if not equal |
| **JNO** | jump if not overflow |
| **JNP/JPO** | jump if not parity |
| **JNS** | jump if not sign |
| **JO** | jump if overflow |
| **JP/JPE** | jump if parity |
| **JS** | jump if sign |

| UNCONDITIONAL TRANSFERS | |
|---|---|
| **CALL** | call procedure |
| **RET** | return procedure |
| **JMP** | jump |
| ITERATION CONTROL | |
| **LOOP** | loop |
| **LOOPE/ LOOPZ** | loop if equal |
| **LOOPNE/ LOOPNZ** | loop if not equal |
| **JCXZ** | jump if CX = 0 |
| INTERRUPTS | |
| **INT** | interrupt |
| **INTO** | interrupt if overflw |
| **IRET** | interrupt return |

### 3.1.2. Speicheradressierung

Der 8086/8088 verfügt über insgesamt 14 Register mit einer Breite von 16 Bit (Tafel 3.2). Speziell sind das:

- 12 Daten- und Adreßregister
  1 Befehlsregister (instruction pointer)
  1 Anzeigeregister (flag register).

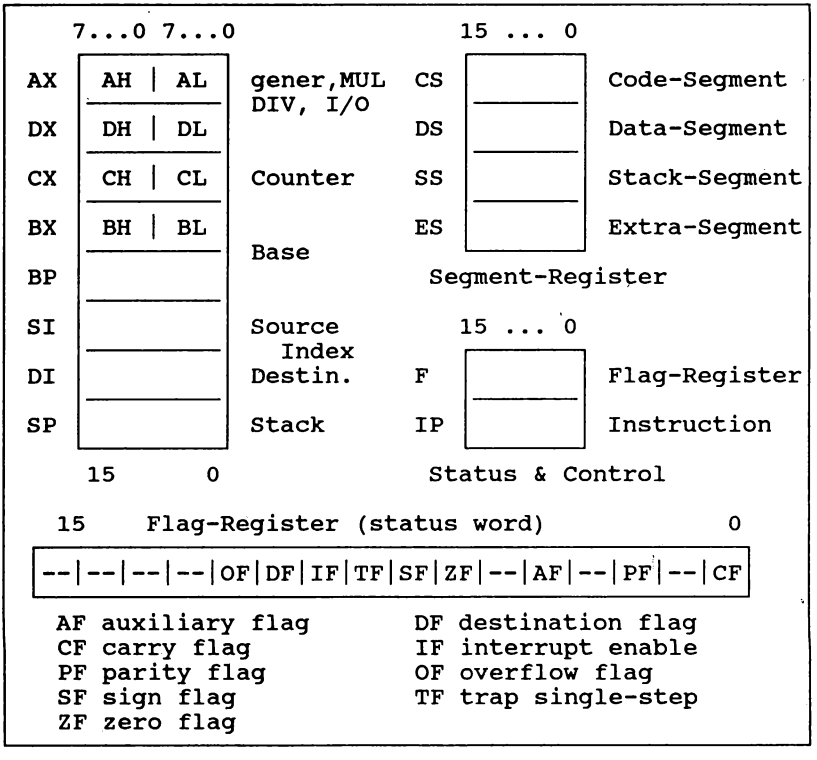
Die effektive Adresse im Adreßraum des Prozessors wird immer durch ein Segmentregister zusammen mit relativer Offsetadresse innerhalb des Segments gebildet.

```
Beispiel:  Segmentadresse      1A7C  = 1A7C0 (hex.)
           Offsetadresse       0385  =  0385 (hex.)
           ---------------            -----
           effektive Adresse:         1AB45 (hex.)
```

Das Speichermodell des 8080 entspricht hier einem Segment von 64 KByte. Durch die Segmentadressierung lassen sich Segmente relativ leicht im Speicher verschieben; die Offsetadressen innerhalb des Segments verändern sich dabei nicht. Die Aufteilung eines Programms durch einen Compiler in Codesegment (CS), Datensegment (DS), Stacksegment (SS) und evtl. ein Extrasegment (ES) wird durch die Prozessorarchitektur effektiv unterstützt. Den meisten Registern sind bestimmte Aufgaben zugewiesen; sie lassen

sich daher nicht alle gleichwertig für die Adressierung verwenden.

**Tafel 3.2. Registermodell 8086/8088**




Der Prozessor kennt verschiedene Modi der Speicheradressierung. Bei der Verwendung der Register ist immer genau zu überlegen, welcher Modus genutzt wird, da das eine häufige Fehlerursache ist, die zu langen Fehlersuchen führt.

### 3.1.3. Datenformate

Kleinste Adressierungseinheit im Speicher ist eine Speicherstelle von einem Byte (8 Bit). Der 8086 ist ein 16-Bit-Prozessor, dessen Register und Zugriffsbreite zum Speicher Verarbeitungseinheiten von 16 Bit zulassen. Die damit realisierten Datenformate sind in Tafel 3.3 aufgezählt. Weitere Formate lassen sich natürlich durch spezielle Programme schaffen; die Verarbeitungsgeschwindigkeit für grundlegende Operationen wird dadurch aber we-

sentlich reduziert. Bei einigen Befehlen wird das adressierte Datenformat durch den Befehlstyp festgelegt, bei anderen durch die angegebenen Register.

**Tafel 3.3. Datenformate des 8086/8088**

| Format | Bit | Wert (hex.) | Wert (dez.) |
|---|---|---|---|
| Byte | 8 | 00 ... FF | 0... 255 |
| Byte (+/-) | 8 | 80...0...7F | -127... +128 |
| Wort | 16 | 0000...FFFF | 0... 65535 |
| Wort (+/-) | 16 | 8000...7FFF | -32768...+32767 |
| BCD | N Byte | (binary coded decimals) | |
| BCD packed | N Byte | (packed binary coded decimals) | |
| ASCII | N Byte | | |
| String | N Byte/Wort | | |
| Pointer | 32 | (Segment,Offset) | |

Ein Byte kann man als numerische Einheit oder als Zeichen interpretieren. Die Zuordnung zwischen den Buchstaben, Zahlen und Sonderzeichen des Alphabets und dem Bytewert wird im MS-DOS standardmäßig nach dem ASCII-Code durchgeführt. Die Zurdnungstabelle ist im Anhang zu finden.

**Tafel 3.4. Portadressen für Geräte**

| Baustein bzw. Gerät | Adr. XT | Adr. AT |
|---|---|---|
| DMA-Controller | 000-00F | 000-01F |
| Interrupt-Controller | 020-021 | 020-03F |
| Zeitgeber | 040-043 | 040-05F |
| PPI | 060-063 | - |
| Tastatur | - | 060-06F |
| DMA-Seitenregister | 080-083 | 080-09F |
| NMI-Maskenregister | | 070-07F |
| Interrupt-Controller 2 | - | 0A0-0BF |
| DMA-Controller 2 | - | 0C0-0DF |
| Reset 80x87 | - | 0F0-0F1 |
| 80x87 | - | 0F8-0FF |
| Game-Controller (Joystick) | 200-20F | 200-207 |
| Erweiterungseinheit | 210-217 | - |
| Parallelport 2 (LPT2) | - | 278-27F |
| Serielles Port 1 (COM1) | 3F8-3FF | 3F8-3FF |
| Serielles Port 2 (COM2) | 2F8-2FF | 2F8-2FF |
| Prototypkarte | 300-31F | 300-31F |
| Festplatte | 320-32F | 1F0-1F8 |
| Parallelport 1 (LPT1) | 378-37F | 378-37F |
| SDLC-Port | 380-38F | 380-38F |
| Synchr. Kommunikation 1 | - | 3A0-3AF |
| Monochromadapter/Drucker | 3B0-3BF | 3B0-3BF |
| Farbgraphikadapter | 3D0-3DF | 3D0-3DF |

### 3.1.4. Geräteadressierung (Ports)

Die Adressierung der Geräte erfolgt über Portadressen, die nicht mit Speicheradressen identisch sind. Die bei den Befehlen IN und OUT angegebenen Adressen liegen im Bereich von 0 bis 65535. Die Zuordnung der Geräte zu den Portadressen ist in Tafel 3.4 zu sehen. Es ist zu beachten, daß Unterschiede bei den Modellen XT und AT bestehen. Der Zugriff auf Ports gestattet zwar den schnellsten Zugriff auf die Geräte, verhindert aber meistens die Portabilität. Einige Programme, bei denen es auf sehr schnelle Gerätereaktionen ankommt, machen trotzdem von dieser Möglichkeit Gebrauch. Meistens läßt sich die verwendete Portadresse dann in einem Konfigurierungsprogramm einstellen.

### 3.1.5. Unterbrechungen (Interrupts)

Eine wichtige Eigenschaft des Prozessors für die Realisierung von Betriebssystemfunktionen ist die Möglichkeit der Unterbrechung des Programmablaufs durch Software oder Hardware. Der Prozessor 8086 läßt 256 Unterbrechungsarten zu. Für diesen Zweck liegt am RAM-Beginn eine Interruptvektortabelle, wo für jede Unterbrechungsart eine Adresse (Segment, Offset) gespeichert ist. Diese Adresse wird angesprungen, wenn ein INT-Befehl mit der entsprechenden Nummer abgearbeitet bzw. ein externes Unterbrechungssignal über den PIC-Baustein erzeugt wird. Die unter dieser Adresse erreichbaren Routinen sind entweder Teil des ROM-BIOS, des BIOS, des DOS oder der Anwenderprogramme. Es werden die folgenden Unterbrechungsarten unterschieden, die im Abschn. 4. weiter präzisiert werden:

- CPU-Interrupts
- Hardwareinterrupts
- BIOS-Interrupts
- DOS-Interrupts
- BASIC-Interrupts
- Adreßinterrupts
- allgemeine Interrupts.

Da die Interruptvektoren im RAM stehen, können sie beliebig durch Programme verändert werden. Von dieser Möglichkeit machen viele Ergänzungsprogramme zum Betriebssystem Gebrauch.

## 3.2. Kommandos, Dateien, Betriebssystem

Die PC-Hardware ist ohne PC-Software nicht zu benutzen. Der Rechner muß zur Ausführung der gewünschten Aktionen in der Lage sein, auf entsprechende **Kommandos** zu reagieren. Die dazu erforderlichen Programme sind in Form von **Dateien** auf einem Datenträger (und teilweise im EPROM) abgelegt. Alle grundlegenden Funktionen (Programme) bilden in ihrer Gesamtheit das PC-**Betriebssystem**, das die Möglichkeiten der Hardware effektiv ausnutzt und die Anwendung erleichtert. Beim MS-DOS handelt es sich um ein plattenorientiertes System (Diskette, Festplatte). Die Reaktionszeiten der externen Speichermedien bestimmen wesentlich das Gesamtverhalten des Systems.

Eine Beschreibung verfügbarer DOS-Kommandos ist im Abschn. 6. zu finden. Im Abschn. 4. wird die interne DOS-Struktur näher behandelt, und der Abschn. 5. ist dem Dateisystem gewidmet.

Das Betriebssytem wird als notwendiges Zubehör zum PC mitgeliefert, oder es wird separat gekauft. Für den PC existieren eine Reihe weiterer Betriebssyteme (z.B. CP/M, XENIX, DOS-Plus, Concurrent DOS), die jedoch alle nicht die Verbreitung von MS-DOS bzw. PC-DOS gefunden haben.

Die ersten IBM-PC kamen 1981 mit dem Betriebssystem PC-DOS auf den Markt. PC-DOS kann als eine spezielle Realisierung von MS-DOS angesehen werden. Die Firma Microsoft liefert das Betriebssystem MS-DOS für kompatible PC mit den Prozessoren 80x86 bzw.80x88. Zwischen den beiden Varianten MS-DOS und PC-DOS gibt es einige kleine Unterschiede, die jedoch für das Verständnis der Arbeitsweise nicht von Bedeutung sind. PC-DOS berücksichtigt wesentlich stärker die von IBM gelieferten peripheren Geräte.

Im Laufe der Zeit wurde MS-DOS ständig weiterentwickelt. In der **Version 1.0** war die interne Schnittstellengestaltung sehr stark an dem erfolgreichen 8-Bit-System CP/M orientiert, um den Übergang zu MS-DOS zu erleichtern. Es gab einige Beschränkungen (Speichergröße 64 KByte, einseitige Disketten u.ä.), die in der **Version 2.0** im wesentlichen behoben waren. Als wichtigste Neuerung ist das an UNIX bzw. XENIX orientierte hierarchische Dateisystem zu nennen, mit dem auch die Verwaltung umfangreicher Dateisammlungen auf Diskette und Festplatte möglich wurde. In der **Version 3.0** kamen einige Verbesserungen des Kommandointerpreters hinzu und die Erweiterung mit Netzwerkfunktionen. Ein Übergang auf Fähigkeiten zum Multi-user- bzw. Multi-tasking-Betrieb und die Aufhebung der 640-KByte-Grenze erfolgte jedoch nicht. Eine Version 4 hat es bisher nicht gegeben; die Version 5 ist als Vorarbeit zum neuen Betriebssystem OS/2 anzusehen. Die Version 3.3 wird auf den kleinen Modellen der PS/2-Serie auch weiterhin genutzt, da die für OS/2 nötigen Befehle des 80286 im 8086-Befehlsvorrat nicht enthalten sind.

Im folgenden Abschnitt werden noch einige Anmerkungen zur

Benutzung von MS-DOS gegeben. Die weiteren Abschnitte enthalten dann detaillierte Informationen zur Arbeitsweise und zur Struktur von MS-DOS.

Der einfachste Weg, mit dem PC und dem Betriebssystem vertraut zu werden, ist die Benutzung. Durch die relativ einfache Struktur ist die Bedienung leicht zu erlernen.

## 3.3. Start des Betriebssystems

Bevor das Betriebssystem aktiv wird, sind folgende Bedienungsschritte erforderlich:

- Inbetriebnahme (vgl. Abschn. 2.10.)
- Betriebssystem bereitstellen
  MSDOS auf Diskette - Diskette in Laufwerk A: einlegen
  MSDOS auf Festplatte - Laufwerk A: öffnen
- Einschalten oder Neustart mit
- Reset bzw. gleichzeitig <Ctrl-Alt-Del>
- Datum/Uhrzeit eingeben, wenn gefordert.

Der Ladevorgang ist ein stufenweiser Prozeß (boot-strap). Die Anfangsladeroutine steht im ROM des PC und lädt das eigentliche Ladeprogramm in den RAM. Dieses erweiterte Ladeprogramm lädt das DOS von der Diskette oder Festplatte mit seinen Bestandteilen und startet die Abarbeitung.

## 3.4. Die Welt der Programme unter MS-DOS

Wenn es gelungen ist, das Betriebssystem MS-DOS zu starten, so ist der Zugang zur wohl größten Softwarebibliothek der Welt eröffnet. Neben den Standardprogrammen von MS-DOS gibt es für alle wichtigen Anwendungen eines PC meist mehrere gute kleinere oder größere Programme bzw. Programmpakete, die den nach einem brauchbaren Programm Suchenden vor eine schwierige Wahl stellen. Die großen Softwarehäuser konkurrieren mit immer neuen Versionen ihrer Programme auf dem Softwaremarkt. Die weiter wachsende Leistungsfähigkeit der PC-Hardware läßt auch noch weiterhin viele Verbesserungen zu. Die folgende Aufzählung von Anwendungsgebieten zeigt die Vielfalt der Einsatzmöglichkeiten. Einige wichtige Programmsysteme werden in diesem Buch vorgestellt bzw. erwähnt.

- DOS-Kommandos (neue oder alternative Programme)
- Werkzeuge zur Softwareentwicklung
- Textsysteme
- Datenbanksysteme
- Kommunikationsprogramme
- Kalkulationsprogramme
- Graphiksysteme
- integrierte Pakete für spezielle Anwendungen
- Datenerfassungssysteme.

Einen Eindruck von der Angebotspalette kann man sich durch das Blättern in einer der vielen Fachzeitschriften zur PC-Technik bzw. zu MS-DOS verschaffen (z.B. BYTE, mc, DOS, Computer Persönlich, PC-Welt, MP). Hier soll eine kleine Auswahl weitverbreiteter Softwarepakete einmal nur genannt werden. Im Abschn. 10. wird etwas näher auf Anwendungspakete eingegangen. Der im Literaturverzeichnis genannte Softwareführer erleichtert die Suche nach geeigneten Programmen für die eigene Arbeit. Die Entscheidung für ein bestimmtes Produkt muß letztlich jeder selbst treffen.

**Tafel 3.5. Softwareliste (Auswahl)**

**Benutzerschnittstellen**

| Programm | Funktion | Firma |
|---|---|---|
| MS-Windows | Windowsystem | Microsoft |
| DOS-Manager | Menüsystem, Maus | Microsoft |
| TopView | Windowsystem | IBM |
| Desqview | Window, multitask. | Quarterdeck |
| GEM-Desktop | Windowsystem | Digital Research |
| Norton Commander | Menüsystem | Norton |
| XTREE | Menüsystem | Executive Systems |
| QDOS | Menüsystem | Gazelle Systems |
| MultiLink | multitask. DOS | Softwarelink |

**Geräteverwaltung/speicherresidente Dienstprogramme**

| Programm | Funktion | Firma |
|---|---|---|
| FASTBACK | Datensicherung | UTIMACO |
| TurboBACKUP | Datensicherung | UTIMACO |
| PCTOOLS | diverse Funktionen | Central Point |
| Copy II PC | Disketten kopieren | Central Point |
| Explorer | Diskettenanalyse | McQuaid Software |
| CopyWrit | Disketten kopieren | McQuaid Software |
| Norton Utilities | diverse Funktionen | Peter Norton |
| MMIBM | Diskettenanalyse | Photon Software |
| SpeedStor | Festplattenverw. | Storage Dimensions |
| SideKick | diverse Tools | Borland |
| SuperKey | Tastaturmakros | Borland |

**Textverarbeitung**

| Programm | Funktion | Firma |
|---|---|---|
| MS-Word | Textsystem | Microsoft |
| WordStar | Textsystem | MicroPro |
| WordStar 2000 | Textsystem | MicroPro |
| WordPerfect | Textsystem | Wordperfect Corp. |
| 1stWordPlus | Textsystem | GST |
| StarWriterPC | Text, DFÜ, Graphik | Star Division |
| Norton Editor | Texteditor | Peter Norton |
| TexAssWindowsPlus | Textsystem | bongartz&schmidt |
| TurboEditor | Texteditor | Borland |

**Tabellenkalkulation**

| Programm | Funktion | Firma |
|---|---|---|
| MS-Excel | Kalk.,Datenbank | Microsoft |
| MS-Multiplan | Kalkulation | Microsoft |
| Quattro | Kalkulation | Borland |
| Javelin | Kalkulation | Ashton Tate |
| 1-2-3 | Kalkulation | Lotus |
| Supercalc | Kalkulation | Comp.-Associates |

**Datenbank**

| Programm | Funktion | Firma |
|---|---|---|
| MS-RBase | relat. Datenbank | Microsoft |
| dBase III Plus | Datenbank | Ashton Tate |
| Oracle | relat. Datenbank | Kettler Consult. |
| GBase | Datenbank | SPI |
| Reflex | Datenbank, Graphik | Borland |

**Graphik**

| Programm | Funktion | Firma |
|---|---|---|
| MS-Chart | Präsentationsgr. | Microsoft |
| DR-Wordchart | Präsentationsgr. | Digital Research |
| AutoCAD | CAD-System | Autodésk |
| GEM-Draw Plus | Zeichenprogramm | Digital Research |

**Integrierte Pakete**

| Programm | Funktion | Firma |
|---|---|---|
| MS-Works | diverse Funktionen | Microsoft |
| FrameWork II | diverse Funktionen | Ashton Tate |
| OpenAccess II | diverse Funktionen | SPI |
| Symphony | diverse Funktionen | Lotus |
| Smart | diverse Funktionen | Innovative Softw. |

# 4. Struktur von MS-DOS

In diesem Abschnitt werden die Komponenten von MS-DOS und deren Funktionen behandelt. Die Kenntnis der internen Zusammenhänge ist vor allem für den Softwareentwickler erforderlich.

## 4.1. DOS-Komponenten

In den Abschnitten 2. und 3. sind die wesentlichen Hardwarekomponenten genannt worden, die programmgesteuert durch den Prozessor koordiniert werden können. Diese Basishardware wird durch ein **ROM-BIOS** (basic input output system software) angesteuert, das die wesentlichen Grundfunktionen des PC realisiert. Diese Software stellt die Grundlage für die Nutzung von MS-DOS/PC-DOS dar und ist Bedingung für die Kompatibilität zum IBM-PC. Das ROM-BIOS liegt in unterschiedlichen Varianten von verschiedenen Produzenten vor. Neben der "Originalversion" von IBM gibt es z.B. ein ROM-BIOS von PHOENIX, das intern anders realisiert ist, da Copyrightbestimmungen zu beachten waren. Das ROM-BIOS ist fest in den ROM des PC eingeschrieben. Änderungen sind nur durch Auswechseln des ROM bzw. das erneute Beschreiben möglich. Da im Laufe der Zeit doch noch einige Fehler zu beheben waren und auch Neuerungen aufgenommen wurden, gibt es mehrere Versionen. Diese Tatsache sollte man berücksichtigen, wenn z.B. ein Programm auf einem PC nicht korrekt funktioniert, obwohl auf einem anderen PC mit dem gleichen DOS keine Fehler festgestellt wurden. Die **Version des ROM-BIOS** ist in den ROM-Speicherstellen F000:FFF5-F000:FFFC abgelegt. Außderdem ist in der letzten ROM-Adresse eine **Modellidentifikation** eingetragen. Die Zählung läuft von FF (hex.) rückwärts mit den Modellen PC, XT, PCjr, AT. Diese Kennung läßt jedoch keine eindeutige Erkennung des PC-Typs zu, da mit den verschiedenen Ausführungen diese Zählung nicht eingehalten wurde.

Im ROM-BIOS ist der Anfangslader zum Einlesen der Komponenten von DOS untergebracht. Nach dem Einschalten des PC wird ein Startprogramm abgearbeitet, das folgende Aktionen durchführt:

- Einschalttest (POST power on self test)
- Feststellen der Hardwareausstattung
- Initialisierung der Bausteine
- Setzen der Interruptvektoren auf Anfangswerte
- Aufruf des Urladers (boot-strap-loader).

Der Einschalttest kann bei großem Speicher relativ lange

dauern; er wird darum beim wiederholten Start (reboot) mit <Reset> oder mit <Ctrl-Alt-Del> übersprungen. Die Speicheraufteilung im PC sieht im Bereich über 640 KByte Speicher für Erweiterungskarten vor. Auf solchen Karten ist im Normalfall zusätzlicher ROM untergebracht, der die auf der Karte installierten Funktionen unterstützt. Das POST-Programm überprüft den zusätlichen Speicher und sucht nach einer Kennung, ob im ROM BIOS-Erweiterungen enthalten sind. Wird solche Kennung gefunden, so ruft es die speziellen Initialisierungsroutinen der Karten auf, die alle notwendigen Schritte zur Einbindung der Hardware in das ROM-BIOS unternehmen. Eine wichtige Karte ist z.B. der Festplattenkontroller. Die Interruptvektoren zeigen nach der Initialisierung auf BIOS-Routinen; sie können später durch andere Werte ersetzt werden. Die bei der Initialisierung ermittelten Statusinformationen werden im unteren RAM-Bereich abgelegt und stehen dort für das Betriebssystem und Anwendungsprogramme zur Verfügung.

Der letzte Schritt des Startprogramms ist der Aufruf des Urladers, der auf dem Diskettenlaufwerk A: nach einem Boot-Eintrag sucht. Wird nach spätestens etwa 2 Sekunden kein Eintrag gefunden, so ist entweder keine Systemdiskette eingelegt oder das Laufwerk offen. Wenn eine Festplatte angeschlossen ist, so wird der Ladevorgang dort wiederholt; ansonsten wird ein evtl. vorhandenes ROM-BASIC gestartet.

Das Startprogramm ist neutral gegenüber dem zu ladenden Betriebssystem. Dadurch können in einem PC verschiedene Systeme gestartet werden. Hier soll nur auf MS-DOS bzw. PC-DOS eingegangen werden, das aus den 3 folgenden Komponenten zusammengesetzt ist:

- **BIOS** (basic input output system)
- **DOS** (disk operating system)
- **COMMAND** (command interpreter)

Diese Komponenten sind als erste Dateien auf einer Systemdiskette gespeichert und werden durch die im Boot-Eintrag enthaltende Laderoutine geladen. Im Inhaltsverzeichnis erscheint nur die Datei COMMAND.COM. Die beiden anderen Komponenten BIOS (IO.SYS bzw. IBMBIO.SYS) und DOS (MSDOS.SYS bzw. IBMDOS.SYS) sind als versteckt gekennzeichnet, damit ein Kopieren an eine andere Stelle verhindert wird. Man kann die Kennzeichnung "versteckt (hidden)" beseitigen und die Dateien kopieren, riskiert dabei jedoch Probleme beim späteren Boot-Vorgang.

Das **BIOS** ist das DOS-Interface zur Hardware. Es ist als Ergänzung zu den Basisroutinen im ROM anzusehen und enthält Modifikationen und Ergänzungen zur Funktion des ROM-BIOS. Durch die enthaltenen Routinen, die als Gerätetreiber (DRIVER) bezeichnet werden, wird zusammen mit dem ROM-BIOS die Verbindung zwischen der Hardware und dem Betriebssystem hergestellt.

Die wesentlichen Funktionen des BIOS sind

- Steuerung von Bildschirm, Tastatur, Drucker
- Steuerung der seriellen Schnittstelle
- Systemparameter bereitstellen
- Zugriff zu Sektoren von Disketten und Festplatten
- Verwaltung von Datum und Uhrzeit (Timer-Routinen)
- Boot-Strap.

Alle Routinen werden über Softwareinterrupts angesprungen. Die Funktionen lassen sich für spezielle Geräte erweitern. Einige Hinweise dazu sind im Abschn. 4.2. enthalten. Die Programmierung der BIOS-Funktionen wird im Abschn. 4.3. beschrieben.

Das **DOS** enthält die Grundfunktionen zur Verwaltung der Ressourcen (Geräte, Speicher, Dateien, Prozesse) und steuert den Gesamtablauf der Programme. Es stellt somit den eigentlichen Kern des Betriebssystems dar. Die Funktionen des DOS bauen auf den BIOS-Funktionen auf und realisieren durch komplexere Algorithmen ein höheres Niveau der PC-Programmierung. Alle Anwendungsprogramme nutzen im wesentlichen die Schnittstelle der DOS-Funktionen, wenn nicht starke Argumente (z.B. Geschwindigkeit) zu einer hardwarenahen Programmierung zwingen. Auch die Laufzeitbibliotheken der Compiler greifen auf die DOS-Schnittstelle zu. Die wesentlichen Funktionen werden im folgenden aufgezählt (eine genaue Beschreibung der Nutzung und Verwendung in eigenen Programmen erfolgt in den Abschnitten 4.4. und 5.):

- Geräteansteuerung auf höherem Niveau als im BIOS
- Speicherverwaltung (memory management)
- Prozeßverwaltung (process management)
- Dateiverwaltung (file and directory management).

Die Komponente von MS-DOS, mit der der Anwender am meisten direkt zu tun hat, ist der Kommandointerpreter **COMMAND**. Er realisiert die standardmäßig gelieferte Benutzerschnittstelle. In COMMAND sind einige Routinen (Int 22h, 23h, 24h) enthalten, die als Standard angenommen werden, wenn durch den Benutzer keine eigenen bereitgestellt werden. Diese Routinen bleiben ständig im Speicher (residenter Teil von COMMAND). Der eigentliche Kommandointerpreter kann durch ein geladenes Anwendungsprogramm überlagert werden und wird nach Beendigung durch den residenten Teil wieder eingelesen. Die Unterteilung in residenten und transienten Teil ist nicht mehr von Bedeutung, da die PC heute schon mit relativ großem Speicher (Standardwert ist 640 KByte) ausgeliefert werden.

In Tafel 4.1 ist die Speicheraufteilung nach dem Laden von MS-DOS dargestellt. Es ist zu entnehmen, daß das eigentliche Betriebssystem im unteren Teil des Speichers abgelegt wird. Der

Speicher kann dadurch nach oben beliebig ausgebaut werden. Durch die Hardwarestruktur des 8086/8088 sind als obere RAM-Grenze bisher 640 KByte festgelegt. Beim AT werden häufig wesentlich größere Speicher verwendet, die dann jedoch nur durch spezielle Treiber (RAM-Disk, RAM-Driver, EMS-Unterstützung) genutzt werden können. Die vollwertige Nutzung des 80286-Adreßraums ist beim Nachfolgesystem OS/2 vom Konzept her bereits gesichert.

**Tafel 4.1. RAM-Speicherbelegung bei MS-DOS**

| RAM-Ende | Modifikation mit |
|---|---|
| **COMMAND** (transienter Teil) | |
| **Systemstack** 256 Byte | |
| **Anwenderprogramm** ... | Programmstart & Beendigung |
| **COMMAND** (residenter Teil) | SHELL in CONFIG.SYS |
| **Device-DRIVER** Geräteschnittstellen | DEVICE in CONFIG.SYS |
| **BUFFER** Ein-/Ausgabe-Puffer | BUFFERS in CONFIG.SYS |
| **DOS** (Interruptroutinen) | |
| **BIOS** (Interruptroutinen) (Tabellen, Statusinformationen) | |
| **Interruptvektoren** | |

## 4.2. Konfigurierung

Beim Laden von MS-DOS ist das Einbinden spezieller Systemroutinen möglich. Nachdem die Standardgerätetreiber geladen sind, wird eine Konfigurierungsdatei mit dem Namen CONFIG.SYS im Rootverzeichnis (vgl. Abschn. 5.) des Ladegeräts gesucht, die Angaben zum Einstellen von Systemparametern und zu speziellen Geräten enthält. Da das MS-DOS relativ klein ist im Vergleich zum verfügbaren Speicher, wird keine Generierung des Systems aus Modulbibliotheken durchgeführt. Einzelne interne Funktionen lassen sich aktivieren oder deaktivieren. Durch diesen Mechanismus ist die Anpassung des Betriebssystems an spezielle Erfordernisse des

Nutzers möglich. Die zusätzlich eingebundenen Systemroutinen sind bis zum nächsten Ladevorgang (Reboot) aktiv. Die Datei CONFIG.SYS enthält ASCII-Text und wird mit einem Texteditor erstellt und geändert. Folgende Kommandos, die im folgenden näher erläutert werden, sind möglich:

| | |
|---|---|
| **BREAK** | Prüfung auf Ctrl-C |
| **BUFFERS** | Anzahl der Sektorpuffer |
| **COUNTRY** | Landescode einstellen |
| **DEVICE** | Installierung eines Gerätetreibers |
| **FCBS** | Anzahl gleichzeitig offener FCB |
| **FILES** | Anzahl offener Dateien |
| **LASTDRIVE** | max. Anzahl Geräte (A: ... Z:) |
| **SHELL** | primärer Kommandointerpreter (Standard: COMMAND.COM). |

**BREAK** [ON | OFF]

Mit diesem Kommando kann die Reaktion des DOS auf Unterbrechungssignale von der Tastatur (<Ctrl-C> oder <Ctrl-Break>) verändert werden. Normalerweise kontrolliert DOS nur nach jedem Zugriff zu Tastatur, Bildschirm oder Drucker, ob eine Unterbrechung erfolgt ist. Durch die Einstellung 'BREAK on' wird nach jedem DOS-Funktionsaufruf eine Überprüfung durchgeführt und DOS-Interrupt 35 (23h) ausgelöst, falls eine Tastaturunterbrechung vorlag. Programme, die nicht die DOS-Funktionen benutzen, müssen eine eigene Behandlung von <Ctrl-C> durch Bereitstellung einer Routine für Interrupt 27 (1Bh) des BIOS realisieren, wenn Programme unterbrechbar sein sollen.

Die BREAK-Funktion kann auch durch das DOS-Kommando BREAK (vgl. Abschn. 6.) gesteuert werden.

**BUFFERS = n**

Der Datenaustausch mit externen Datenträgern erfolgt über Pufferspeicher. Normalerweise sind zwei Puffer im Speicher realisiert. Umfangreiche Dateioperationen erfordern das Einlesen mehrerer Sektoren zum Auffinden der gesuchten Daten. Durch Vergrößerung der Pufferanzahl kann der Dateizugriff beschleunigt werden. Jeder Puffer belegt 528 Byte des Speichers. Die Anzahl sollte daher nicht unnötig groß gewählt werden. Das Maximum ist 255, eine sinnvolle Größe ist für viele Anwendungsfälle ein Wert von n=20.

**COUNTRY = Landescode**

Mit diesem Kommando lassen sich einige landesspezifische Parameter des DOS einstellen. Dazu gehören z.B. Format für Datum, Uhrzeit und Währungssymbol. Standardeinstellung ist der Code 001 (USA); für den deutschen Sprachraum gilt der Code 049. Die Landeskennungen entsprechen den im internationalen Telefonverkehr gebräuchlichen Werten.

**DEVICE = [d:]Treibername** /Optionen

Dieses Kommando ist vorgesehen, um neue Gerätetreiber in die Behandlung durch DOS einzubeziehen. Nach dem Laden ist der Treiber integraler Bestandteil von DOS. Einige installierbare Treiber gehören zum DOS, andere werden durch den Hersteller der Spezialgeräte mitgeliefert.

Beispiel: - Allgemeiner DRIVER für Blockgeräte

```
DEVICE = DRIVER.SYS /Optionen

     /D:n    Kanalnummer (0=A,1=B,...)
     /F:nn   Format  0 = 360 KByte ; 1 = 1.2 MByte
       (Std. : 2)    2 = 720 KByte ; 3 =   8"SD
                     4 =   8"DD ; 5 = Festplatte
                     6 = Tape   ; 7 = sonstige
     /C      Verschlußinterrupt beachten
     /H:nn   Anzahl Köpfe
     /S:ss   Anzahl Sektoren/Spur
     /T:nnn Anzahl Spuren
     /N      nicht auswechselbar
```

- ANSI-Treiber für Tastatur und Bildschirm

  DEVICE = ANSI.SYS (vgl. Abschnitte 6.3.,6.4.)

- virtuelle RAM-Disk

  DEVICE = VDISK.SYS [kbyte] [/E] [/A]

  /E Erweiterungsspeicher > 1MByte
  /A EMS-Spezifikation (Intel, Lotus, Microsoft)

- VDI-Treiber für Graphikgeräte (Abschn. 10.5.)
- CP/M-Treiber (z.B. SuperCopy, UniForm)
- Netz-Treiber (Abschn. 10.4.)

**FILES = n**

Einstellung der max. Anzahl offener Dateien. Der Wert von n liegt zwischen 8 (Standard) und 20. Für jede Datei werden im Speicher 38 Bytes Dateikontrollinformation reserviert (vgl. DOS-Funktionen zur Dateibehandlung Abschn. 6.).

Das Kommando **FCBS** ist nur für das Dateisystem der Version 1.0 von MS-DOS wichtig. Die Funktionen des Dateisystems ab Version 2.0 arbeiten mit einem Dateiidentifikator (file-handle).

**LASTDRIVE = d**

Für jedes Blockgerät wird von DOS eine Parametertabelle geführt. Mit diesem Parameter wird der nötige Bereich eingerichtet. Für d ist die höchste Gerätekennung anzugeben.

Beispiel: LASTDRIVE = G  Geräte A: B: C: D: E: F: G:

**SHELL = Kommandointerpreter**

Der Standard-Kommandointerpreter ist COMMAND.COM, der vom DOS nach der Installation der Treiber geladen wird. Will man einen eigenen Interpreter verwenden, so ist der entsprechende Name hier anzugeben. Zusätzliche Informationen nach dem Programmnamen werden als Parameter angesehen.

Beispiel für eine Datei CONFIG.SYS

```
BUFFERS = 10
FILES = 20
DEVICE = driver.sys /D:1     2. Laufwerk 720 KByte
DEVICE = vdisk.sys 360 /E    RAM-Disk 360 KByte Ext. Memory
BREAK on
```

Eine weitere Möglichkeit zur Einstellung von Standardwerten ist die Verwendung der Datei AUTOEXEC.BAT, die vom Kommandointerpreter COMMAND.COM nach dem Start gelesen wird (vgl. Abschn. 7.). Der Unterschied zur Datei CONFIG.SYS besteht jedoch darin, daß in der Datei AUTOEXEC.BAT normale DOS-Kommandos stehen und der Aufruf jederzeit wiederholt werden kann. Die mit CONFIG.SYS eingestellten Werte werden in der Regel erst durch ein erneutes Starten geändert. Änderungen in der aktuellen Datei CONFIG.SYS können evtl. einen neuen Start unmöglich machen, wenn Fehler auftreten. Man sollte für solche Fälle immer noch eine Systemdiskette zur Verfügung haben, die anstelle der Festplatte genutzt wird, um die Datei CONFIG.SYS zu korrigieren.

## 4.3. Nutzung der BIOS-Funktionen

### 4.3.1. BIOS-Funktionen (Übersicht)

Die BIOS-Funktionen stellen die Basis dar, auf der MS-DOS aufbaut. Alle PC-Komponenten lassen sich über BIOS-Funktionsaufrufe ansteuern. Beim IBM-PC wurden diese Funktionen Software-Interrupts im Bereich von 10 bis 1F (hex.) zugeordnet. Diese Zuordnung hat sich als ein Standard herausgebildet, der die wichtige Kompatibilität zum IBM-PC garantiert (vgl. Tafel 4.2).

**Tafel 4.2. BIOS-Interrupts**

| Int | Funktion |
|---|---|
| | **CPU-Interrupts** |
| 00 | Division durch 0 |
| 01 | Einzelschritt (single step) |
| 02 | NMI (non maskable interrupt) |
| 03 | Unterbrechungspunkt (breakpoint) |
| 04 | Überlauf (overflow) |
| 05 | Bildschirm ausdrucken (hardcopy) |
| | **IBM-PC Hardware (PIC 8259)** |
| 08 | Zeitgeber (Timer, Hardware-Int 0) |
| 09 | Tastatur (Keyboard, Hardware-Int 1) |
| 0B | serieller Kanal 2 (COM2, Hardware-Int 3) |
| 0C | serieller Kanal 1 (COM1, Hardware-Int 4) |
| 0D | Festplatte (Hard-Disk, Hardware-Int 5) |
| 0E | Diskette (Floppy-Disk, Hardware-Int 6) |
| 0F | Drucker (Printer, Hardware-Int 7) |
| | **BIOS-Funktionen** |
| 10 | Bildschirm |
| 11 | Ausstattung feststellen |
| 12 | Speichergröße ermitteln |
| 13 | Diskette/Festplatte |
| 14 | serielle Schnittstelle |
| 15 | Kassette (falls vorhanden) |
| 16 | Tastatur |
| 17 | Drucker |
| 18 | ROM-Basic (falls vorhanden) |
| 19 | Neustart (Boot-Strap) |
| 1A | Uhrzeit |
| 1B | Tastatur-Break (Ctrl-C) |
| 1C | Zeitgeberbehandlung |
| 1D | Bildschirm-Initialisierung |
| 1E | Adresse Diskettenparameter |
| 1F | Adresse Graphikzeichensatz |

Durch den BIOS-Teil, der zusammen mit DOS geladen wird, wird die Geräteschnittstelle zum DOS realisiert und die evtl. erforderliche Anpassung an die Routinen des ROM-BIOS durchgeführt.

Die in Tafel 4.2 genannten Funktionen (10h - 1Fh) werden vom DOS genutzt und können ebenso von jedem Anwendungsprogramm aus aufgerufen werden. Die im Bereich 00h - 0Fh liegenden Interrupts werden für spezielle Hardwaresteuerungen genutzt. Die bei der Funktionsausführung ermittelten Informationen werden an das rufende Programm in Registern zurückgeliefert. Zwischenergebnisse und aktuelle Statusinformationen werden im unteren RAM-Bereich (Adressen 0:400 - 0:5FF) abgelegt und können von Programmen jederzeit gelesen werden.

**Statusinformationen im unteren RAM-Bereich**

Das ROM-BIOS hinterläßt nach der Initialisierung im unteren RAM-Bereich (400-4FF, 500-5FF) Informationen über die Ausstattung des PC und reserviert Felder für Statusanzeigen, die vom BIOS bei nachfolegenden Aufrufen genutzt werden und den aktuellen Stand anzeigen. Für spezielle hardwarenahe Arbeit kann die Information in diesen Feldern evtl. von Nutzen sein.

| | |
|---|---|
| **400-401** | Portadresse serielle Schnittstelle COM1 |
| **402-403** | Portadresse serielle Schnittstelle COM2 |
| **404-407** | reserviert für COM3 und COM4 |
| **408-409** | Portadresse für parallele Schnittstelle LPT1 |
| **40A-40B** | Portadresse für parallele Schnittstelle LPT2 |
| **40C-40D** | Portadresse für parallele Schnittstelle LPT3 |
| **40E-40F** | reserviert für LPT4 |
| **410-411** | Ausstattungsliste (BIOS-Funktion 11) |
| **413-414** | Speicherkapazität in KByte (BIOS-Funktion 12) |
| **417-418** | Tastaturstatus |
| **419** | aktuelles Zeichen bei Eingabe mit <Alt> |
| **41A-41B** | Anfangsadresse Tastaturpuffer |
| **41C-41D** | Endadresse Tastaturpuffer |
| **41E-43D** | Tastaturpuffer für max. 16 Scan-Codes (16*2) |
| **43E** | Diskettenanzeige (Rekalibrierung) |
| **43F** | Anzeigebits für Diskettenmotor |
| **440** | Motorwartezeit (Takte bis Abschalten) |
| **441** | Diskettenstatusbits<br>7 Time-Out 3 DMA-Fehler<br>6 Spur nicht gefunden 2 Sektor nicht gefunden<br>5 FDC defekt 1 Adreßmarke nicht gef.<br>4 CRC-Fehler 0 ungültiges Kommando |
| **442-448** | Status Bildschirmadapter |
| **449** | aktueller Bildschirmmodus |
| **44A-44B** | Bildschirmbreite (20, 40, 80) |
| **44C-44D** | Länge einer Bildschirmseite |
| **44E-44F** | Offsetadresse der aktuellen Bildschirmseite |
| **450-45F** | Cursorposition (Spalte, Zeile) für 8 Seiten |
| **460-461** | Cursorgröße in Rasterzeilen (Ende, Anfang) |

| | |
|---|---|
| **462** | aktuelle Bildschirmseite |
| **463-464** | Adresse Baustein 6845 (Standard 3D4) |
| **466** | Bitmaske für Farbpaletten |
| **46C-46F** | Hauptzähler für Systemtakt (Mitternacht = 0) |
| **470** | Zählerüberlauf nach 24 Stunden |
| **471** | Anzeige Tastaturunterbrechung (Ctrl-C, Ctrl-Break) |
| **500** | Status Bildschirmdruck (hardcopy)<br>00 kein Fehler FF Druckerfehler<br>01 Druck aktiv |
| **504** | Anzeige, ob einziges Diskettenlaufwerk A:<br>als A: (Wert 00) oder B: (Wert 01) aktiviert |

Das Lesen dieser Werte ist ohne Probleme möglich; das Verändern sollte sehr vorsichtig erfolgen, da die Angaben lebenswichtig für das BIOS sind.

### Aufruf von BIOS-Routinen

Alle BIOS-Routinen werden über Software-Unterbrechungen aufgerufen. Der Aufruf ist für alle Funktionen gleich und in MS-DOS für Assemblerprogramme definiert. Zu jeder BIOS-Funktion gehören mehrere Unterfunktionen, die durch eine Angabe im Register AX (AH, AL) ausgewählt werden. Die BIOS-Funktion wird als Interruptnummer im Befehl INT angegeben. Sind zusätzliche Parameter erforderlich, so werden sie in weiteren Registern angegeben. Das Ergebnis von BIOS-Funktionsaufrufen wird im Register AX bereitgestellt. Im Abschn. 11. sind Beispiele zum Aufruf von BIOS-Funktionen aus den verschiedenen Programmiersprachen enthalten.

**Beispiel:** Aufruf der BIOS-Routinen von Assembler

```
MOV  ...          Register laden
MOV  AH,uf        Unterfunktion
INT  n            BIOS-Funktion
```

Wird der Aufruf einer BIOS-Funktion als separates Unterprogramm realisiert, so sind die Schnittstellenbedingungen des Entwicklungssystems einzuhalten. Die meisten Compiler enthalten Systemroutinen zum Aufruf von Interrupts. Für Turbo-Pascal ist die Prozedur INTR zum Aufruf von Interrupts vorgesehen. Die Register werden in einem Record definiert und auf die entsprechenden Werte gesetzt, bevor der INTR-Aufruf erfolgt. Im Abschn. 11. ist eine Pascal-Unit mit einigen BIOS-Routinen enthalten.

Die in Tafel 4.2 angegeben **CPU-Interrupts** werden nicht vom Programm aus aufgerufen; sie entstehen in bestimmten Situationen beim Abarbeiten eines Programms und gestatten die Programmunterbrechung bei Fehlern (z.B. Division durch 0) und beim Programmtest (z.B. DEBUG). Diese Interrupts werden in der Regel durch die

Fehlerbehandlungsroutinen des Programmiersystems bedient. Die Standardbehandlung durch das BIOS führt zum definierten Programmabbruch.

Die **Hardwareinterrupts** werden durch externe Geräte erzeugt. Beim PC/XT sind 8 Interruptleitungen vorhanden, die durch den Schaltkreis 8259 den entsprechenden Interruptvektoren zugeordnet werden. Beim PC/AT sind 16 solcher Leitungen vorhanden.

### 4.3.2. Bildschirmfunktionen

Für das Verständnis der Bildschirmprogrammierung sind einige Begriffe wichtig, die in diesem Zuasammenhang immer wieder auftreten. Folgende Begriffe sollen hier kurz erläutert werden:

- Controller-Ports
- Bildschirmspeicher, Bildschirmseiten
- Bildschirmmodi
- Attribute, Farbpaletten
- Zeichensatz.

Die direkte Steuerung der Bildschirmausgabe über **Ports** und **Bildschirmspeicher** ist der schnellste Weg zur Darstellung. Bei der Programmierung legt man sich aber damit auf eine ganz bestimmte Hardware fest und ist nicht mehr in der Lage, Programme zwischen unterschiedlich konfigurierten PC auszutauschen. Entweder muß man den Aufwand treiben, durch Schnittstellentreiber mehrere Bildschirmtypen zu berücksichtigen, oder, mehrere Programmversionen anzubieten. Die Hardwareparameter zur direkten Bildschirmsteuerung sind im technischen Handbuch zum Adapter zu finden.

Der **Bildschirmspeicher** enthält das Speicherabbild des Bildes. Sowohl der Prozessor als auch der CRT-Baustein kann zum Speicher zugreifen. Die Darstellung der Daten zu einem Bildpunkt (Pixel) ist vom eingestellten **Bildschirmmodus** abhängig. Jeder Adapter kann in verschiedenen Modi arbeiten. Eine grobe Unterteilung unterscheidet Text- und Graphikmodus. Der Modus wird durch eine Nummer gekennzeichnet. Im Graphikmodus werden Textzeichen aus Bildpunkten generiert. Das Punktmuster wird durch den Textfont bestimmt. Der Standardzeichensatz ist im ROM gespeichert. Auf die Tabelle verweist der Adreßinterrupt 1Fh. Die Tabelle kann durch eine andere Tabelle im RAM ersetzt werden. Das wird z.B. durch Textsysteme angewendet, die im Graphikmodus arbeiten. Bei kleineren Zeichen lassen sich damit z.B. mehr Zeilen auf einem Bildschirm unterbringen. Die wichtigsten Modi sind in Tafel 4.3 zusammengestellt. Es ist zu beachten, das weiterentwickelte Adapter alte Modi meistens mit erweiterten Möglichkeiten unterstützen. Im Speicher auf den Bildschirmadaptern können in Abhängig-

keit von der Größe des Speichers mehrere Bildschirmabbilder untergebracht sein, die durch eine Seitennummer identifiziert werden. Der Aufbau einer Seite ist vom eingestellten Modus abhängig.

**Tafel 4.3. Bildschirmmodi**

| Modus | Typ | Auflösung Text | Auflösung Graph | Farben | Adapter |
|---|---|---|---|---|---|
| 0 | Text | 40*25 | | 16 (Grau) | CGA,EGA (BW40) |
| 1 | Text | 40*25 | | 16/64 | CGA,EGA (CO40) |
| 2 | Text | 80*25 | | 16 (Grau) | CGA,EGA (BW80) |
| 3 | Text | 80*25 | | 16 | CGA,EGA (CO80) |
| 4 | Graph | 40*25 | 320*200 | 4 | CGA,EGA |
| 5 | Graph | 40*25 | 320*200 | 4 (Grau) | CGA,EGA |
| 6 | Graph | 80*25 | 640*200 | 2 | CGA,EGA |
| 7 | Text | 80*25 | 720*350 | 2 | MGA,EGA (MONO) |
| 11 | EGA-intern (Laden Farbzeichensatz) | | | | EGA |
| 12 | EGA-intern (Laden Monozeichensatz) | | | | EGA |
| 13 | Graph | 40*25 | 320*200 | 16 | EGA |
| 14 | Graph | 80*25 | 640*200 | 16 | EGA |
| 15 | Graph | 80*25 | 640*350 | 4 | EGA |
| 16 | Graph | 80*25 | 640*350 | 64 | EGA |
| 17 | Graph | 80*30 | 640*480 | 2 | VGA |
| 18 | Graph | 80*30 | 640*480 | 16 | VGA |
| 19 | Graph | 40*25 | 320*200 | 256 | VGA |

Die Qualität von Textzeichen ist auf dem EGA-Bildschirm durch die höhere Auflösung wesentlich besser. Ebenso stehen mit dem EGA-Adapter und ECD-Bildschirm (enhanced colour display) mehr Farben zur Verfügung. Farben werden aus vier Steuersignalen erzeugt: Intensität, Rotanteil, Grünanteil, Blauanteil. Die Farben lassen sich aus **Farbpaletten** auswählen, in denen jeweils bestimmte Mischungen zusammengestellt sind. Die Anzahl von Farben in jeder Palette wird praktisch durch die Anzahl Bits für jeden Punkt bestimmt. Diese Angaben werden als **Attribute** eines Zeichens oder Pixels bezeichnet. Die Darstellung der Attribute ist in den einzelnen Modi unterschiedlich. Im Textmodus werden im Attribut sowohl Vordergrundfarbe (unteres Halbbyte) als auch Hintergrundfarbe (oberes Halbbyte) dargestellt. Bit 7 wird nicht als Intensität, sondern als Blinkattribut verwendet. Dadurch sind für den Hintergrund nur 8 Farben verfügbar. Im Graphikmodus wird für Pixelausgabe keine Hintergrundfarbe benötigt. Bei Zeichendarstellung ist das Attributbyte wie im Textmodus aufgebaut. In Tafel 4.4 sind die 16 Farben des PC und ihre 4-Bit-Codierung genannt. Auf Grund unterschiedlicher Codierung der Farbinformation in den einzelnen Adaptern ergeben sich Probleme bei der direkten Programmierung des Bildschirms. Das ist ein Grund mehr, sich der BIOS-Schnittstelle zu bedienen. Graphiksysteme stellen verschiedene Treiberprogramme bereit, die auf den verwendeten Adapter

eingestellt werden können. Mit dem neuen System PS/2 von IBM kommen weitere Adapter (MCGA, VGA) hinzu, die eine noch bessere Auflösung als im EGA-Modus gestatten.

**Tafel 4.4. Volle Farbpalette des PC (CGA)**

| I | R | G | B | Nummer | Farbe |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | Schwarz |
| 0 | 0 | 0 | 1 | 1 | Blau |
| 0 | 0 | 1 | 0 | 2 | Grün |
| 0 | 0 | 1 | 1 | 3 | Türkis |
| 0 | 1 | 0 | 0 | 4 | Rot |
| 0 | 1 | 0 | 1 | 5 | Magenta |
| 0 | 1 | 1 | 0 | 6 | Braun (Dunkelgelb) |
| 0 | 1 | 1 | 1 | 7 | Hellgrau (Normal-Weiß) |
| 1 | 0 | 0 | 0 | 8 | Dunkelgrau (Schwarz) |
| 1 | 0 | 0 | 1 | 9 | Hellblau |
| 1 | 0 | 1 | 0 | 10 | Hellgrün |
| 1 | 0 | 1 | 1 | 11 | Helltürkis |
| 1 | 1 | 0 | 0 | 12 | Hellrot |
| 1 | 1 | 0 | 1 | 13 | Hellmagenta |
| 1 | 1 | 1 | 0 | 14 | Gelb (Hellgelb) |
| 1 | 1 | 1 | 1 | 15 | helles Weiß |

Die Steuerung der Attribute und die Auswahl besonderer Eigenschaften des Bildschirmadapters sind nur auf der BIOS-Ebene definiert. Compiler für höhere Programmiersprachen (z.B. Turbo-Pascal) stellen die benötigten Prozeduren direkt zur Verfügung.

Der verwendete **Zeichensatz** ist für die Textmodi als Tabelle vordefiniert. In den Graphikmodi werden alle Zeichen aus Pixeln in einer Box (Breite*Höhe) dargestellt. Die Ausgabe von Text im Graphikmodus ist dadurch wesentlich langsamer. Der Zeichensatz läßt sich im Graphikmodus jederzeit verändern.

**Tafel 4.5. Farbpaletten im Modus 4 (CGA)**

| Farbwert | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| Palette 0 | Hintergrund | Grün | Rot | Braun |
| Palette 1 | Hintergrund | Türkis | Magenta | Hellgrau |
| Palette 2 | Hintergrund | Hellgrün | Hellrot | Gelb |
| Palette 3 | Hintergrund | Helltürkis | Hellmagenta | Weiß |

Alle BIOS-Funktionen für den Bildschirm sind unter Interrupt 16 (10h) zusammengefaßt. Der Adreßinterrupt 1Fh (vgl. Abschn. 4.3.7.) stellt die Adresse der Zeichensatztabelle bereit. Es sind die in Tafel 4.5 genannten Funktionen realisiert. Im DOS sind

keine komplexen Funktionen zur Bildschirmsteuerung definiert, da der Kommandointerpreter eine zeichenweise Ein- und Ausgabe verwendet. Für graphische Anwendungen muß man daher die BIOS-Funktionen verwenden.

**Tafel 4.6. BIOS-Bildschirmfunktionen**

| Nr.(hex.) | Funktion |
|---|---|
| 00 (00h) | Bildschirmmodus einstellen |
| 01 (01h) | Cursorgröße festlegen |
| 02 (02h) | Cursorposition setzen |
| 03 (03h) | Cursorposition abfragen |
| 04 (04h) | Lichtstiftposition abfragen |
| 05 (05h) | Bildschirmseite festlegen |
| 06 (06h) | Fenster nach oben rollen |
| 07 (07h) | Fenster nach unten rollen |
| 08 (08h) | Zeichen/Attribut an Cursorposition lesen |
| 09 (09h) | Zeichen/Attribut an Cursorposition schreiben |
| 10 (0Ah) | Zeichen an Cursorposition schreiben |
| 11 (0Bh) | Farbpalette festlegen |
| 12 (0Ch) | Graphikpunkt setzen |
| 13 (0Dh) | Graphikpunkt abfragen |
| 14 (0Eh) | Zeichen ausgeben und Cursor weiterbewegen |
| 15 (0Fh) | aktuellen Bildschirmmodus ermitteln |
| 16 (10h) | EGA-Farbpalettenregister setzen |
| 17 (11h) | EGA-Zeichengeneratorroutine |
| 18 (12h) | EGA-Konfigurierung |
| 19 (13h) | Zeichenkette (String) schreiben |

Die in Tafel 4.6 genannten Funktionen werden im folgenden einzeln beschrieben. Als Interruptnummern und Auswahlcodes werden alle Werte in hex. Darstellung angegeben.

| **Bildschirmmodus einstellen** | **Int 10-00** |
|---|---|
| **Aufruf**<br>AH = 00h<br>AL = Modus (Tafel 4.3) | **Ergebnis**<br>Bildschirm gelöscht<br>Byte 449h = Modus |

In Turbo-Pascal (Version 3.0) sind spezielle Prozeduren definiert, die eine Auswahl des Bildschirmmodus gestatten. Da die Anzahl der unterstützten Adapter ständig zunimmt, wurden in Version 4.0 in der Unit Graph für die einzelnen Adapter gesonderte Routinen definiert. Im Abschn. 8.3. sind die verfügbaren Pascalroutinen aufgeführt. Bei neuen und speziellen Adaptern gelten manchmal gesonderte Festlegungen. So ist z.B. beim Bildschirmadapter des Schneider-PC1512 die volle Farbpalette im Graphikmodus verfügbar. Dazu wird das BIOS-Interrupt 15h (reserviert für Kassette beim IBM-PC) neu belegt und zur Einstellung der Farbmaske u.a. benutzt. Die Besonderheiten sind dem technischen

Handbuch zum Adapter zu entnehmen.

| **Cursorgröße definieren (Textmodus)** | **Int 10-01** |
|---|---|
| **Aufruf**<br>AH = 01h<br>CH = Bits 0...4 Startzeile<br>CL = Bits 0...4 Endzeile | **Ergebnis**<br>Byte 460h = CL<br>Byte 461h = CH |

Der Cursor erscheint normalerweise nur im Textmodus. Die Werte für Anfangs- und Endzeile im Rasterbild eines Zeichens sind vom verwendeten Adapter abhängig. Ein Strichcursor ergibt sich beim CGA z.B. mit CH=6 und CL=7. Wird CH=20h gesetzt, so wird der Cursor abgeschaltet.

| **Cursor positionieren** | **Int 10-02** |
|---|---|
| **Aufruf**<br>AH = 02h<br>DH = Zeilennummer<br>DL = Spaltennummer<br>BH = Seitennummer | **Ergebnis**<br>Wort 0:450h [Seite] |

Als Adresse (0, 0) wird die linke obere Ecke angenommen. Die maximale Seitennummer ist vom Bildschirmadapter (Größe des Bildschirmspeichers) abhängig. Im Graphikmodus ist Seitennummer=0 anzugeben.

| **Cursorposition abfragen** | **Int 10-03** |
|---|---|
| **Aufruf**<br>AH = 03h<br>BH = Seitennummer | **Ergebnis**<br>DH = Zeilennummer<br>DL = Spaltennummer<br>CX = Cursorgröße, wie AH=1 |

Die Angabe der Position ist wie bei AH=2. Im Graphikmodus ist Seitennummer=0 anzugeben. Die Position des Cursors ist z.B. wichtig bei der Bearbeitung von Bildschirmmasken.

| **Lichtgriffelposition abfragen** | **Int 10-04** |
|---|---|
| **Aufruf**<br>AH = 04h | **Ergebnis**<br>AH = 01h Griffel aktiv<br>DH = Zeilennummer (Text)<br>DL = Spaltennummer (Text)<br>CH = Pixelzeile (Graphik)<br>BX = Pixelspalte (Graphik) |

Als Ergebnis wird sowohl die Position im Textmodus als auch die Position im Graphikmodus geliefert. Wenn AH = 00h ist, so ist der Lichtgriffel nicht aktiv.

| **Bildschirmseite festlegen** | **Int 10-05** |
|---|---|
| **Aufruf**<br>AH = 05h<br>AL = Seitennummer | **Ergebnis**<br>Byte 0:462 = Seitennummer |

Als Standard ist Seite 0 aktiv. Durch die Seitennummer wird die Anfangsadresse des aktuellen Bildschirmspeichers festgelegt.

| **Fenster nach oben rollen (Scrolling)** | **Int 10-06** |
|---|---|
| **Aufruf**<br>AH = 06h<br>AL = Anzahl Zeilen<br>CX = Zeile,Spalte oben li.<br>DX = Zeile,Spalte unten re.<br>BH = Attribut für Leerzeilen | **Ergebnis**<br>Bildfenster wird um Anzahl Zeilen nach oben gerollt und unten Leerzeilen eingefügt |

Beim Rollen des Bildschirmfensters wird zuerst eine Anzahl Leerzeilen eingeschoben und dann wird zeilenweise neugeschrieben. Wird in AL = 0 angegeben, so wird das ganze Fenster gelöscht.

| **Fenster nach unten rollen (Scrolling)** | **Int 10-07** |
|---|---|
| **Aufruf**<br>AH = 07h<br>AL = Anzahl Zeilen<br>CX = Zeile,Spalte oben,li.<br>DX = Zeile,Spalte unten re.<br>BH = Attribut für Leerzeilen | **Ergebnis**<br>Bildfenster wird um Anzahl Zeilen nach unten gerollt und oben Leerzeilen eingefügt |

Diese Funktion ist analog zur vorhergehenden, mit dem Unterschied, daß oben Leerzeilen eingefügt werden.

| **Zeichen und Attribut (Cursor) lesen** | **Int 10-08** |
|---|---|
| **Aufruf**<br>AH = 08h<br>BH = Bildschirmseite | **Ergebnis**<br>AL = Zeichen<br>AH = Attribut |

Die Attribute werden nur im Textmodus bereitgestellt.

| **Zeichen und Attribut (Cursor) schreiben** | **Int 10-09** |
|---|---|
| **Aufruf**<br>AH = 09h<br>AL = Zeichen<br>BL = Attribut<br>BH = Bildschirmseite<br>CX = Anzahl Wiederholungen | **Ergebnis**<br>Zeichen wird ab Cursorpos. CX-mal geschrieben, Cursorposition bleibt erhalten |

Im Textmodus wird am Ende einer Zeile automatisch in einer neuen Zeile weitergeschrieben. Es ist zu beachten, daß der Cursor nicht bewegt wird. Im Graphikmodus wird die in BL angegebene Farbe als Vordergrundfarbe verwendet. Ist Bit 7 auf 1 gesetzt, so erfolgt ein XOR-Verknüpfung (Exklusiv-ODER) der Farbbits mit den aktuellen Farbbits im Speicher. Die Zeichen sind dadurch in jedem Fall anders als vorher.

| **Zeichen (Cursor) schreiben** | **Int 10-0A** |
|---|---|
| **Aufruf**<br>AH = 0Ah<br>AL = Zeichen<br>BH = Bildschirmseite<br>CX = Anzahl Wiederholungen | **Ergebnis**<br>Zeichen wird ab Cursorpos. CX-mal geschrieben, Cursorposition und Attribut bleiben erhalten |

Diese Routine arbeitet wie Routine 09h; das Attribut einer Zeichenposition wird dabei jedoch nicht verändert.

| **Farbpalette festlegen (Graphik)** | **Int 10-0B** |
|---|---|
| **Aufruf**<br>AH = 0Bh<br>BH = Farbpalettennummer<br>BL = Farbwert für Palette | **Ergebnis**<br>Einstellung für folgende Pixelausgaben |

Diese Funktion ist von den Möglichkeiten des verwendeten Adapters und vom eingestellten Bildschirmmodus abhängig. Für den CGA-Adapter gelten die Werte aus Tafel 4.5, die in BH angegebene Nummer darf im Bereich von 0 bis 127 liegen. Es ist zu beachten, daß mit der Festlegung einer neuen Palette alle Pixel auf dem Bildschirm in den neuen Farben ausgegeben werden, da intern eine Harwareeigenschaft des CRT-Controllers umprogrammiert wird. Die Farbmöglichkeiten des CGA sind im Graphikmodus nicht sehr ergiebig. Bessere Möglichkeiten werden mit der EGA-Karte und den neuen Adaptern zum PS/2 geboten.

| **Pixel setzen (Graphik)** | **Int 10-0C** |
|---|---|
| **Aufruf**<br>AH = 0Ch<br>AL = Farbe (Bits 0...6)<br>XOR-Verknüpfung (Bit7)<br>CX = Spalte<br>DX = Zeile | **Ergebnis**<br>Pixel an Position (Zeile, Spalte) wird mit Farbe AL gesetzt. Ist in AL Bit7=1, wird aktuelles Farbattribut XOR-verknüpft |

Bei der EGA-Karte kann zusätzlich in Register BH die Bildschirmseite angegeben werden.

| **Pixel lesen (Graphik)** | **Int 10-0D** |
|---|---|
| **Aufruf**<br>AH = 0Dh<br>CX = Spalte<br>DX = Zeile | **Ergebnis**<br>AL = Pixelattribut (DX,CX) |

Hier kann in BH ebenfalls die Bildschirmseite angegeben werden.

| **Zeichen schreiben mit Cursorbewegung** | **Int 10-0E** |
|---|---|
| **Aufruf**<br>AH = 0Eh<br>AL = Zeichen<br>BL = Vordergrundfarbe<br>BH = Bildschirmseite | **Ergebnis**<br>Schreiben im TTY-Modus Cursor wird weiterbewegt, automatischer Zeilenvorschub und Rollen |

Es wird die Bildschirmbreite des aktuellen Bildschirmmodus verwendet. Die Ausgabeform entspricht einer normalen Terminalausgabe (TTY-Modus).

| **Bildschirmmodus feststellen** | **Int 10-0F** |
|---|---|
| **Aufruf**<br>AH = 0Fh | **Ergebnis**<br>AL = aktueller Modus<br>AH = Spaltenanzahl<br>BH = Bildschirmseite |

Es werden hier die mit Funktion 00h eingestellten Parameter bereitgestellt. Die Informationen sind z.B. beim Programmbeginn erforderlich, wenn man eine bestimmte Bildschirmstruktur erstellen will.

| Zeichenkette (String) schreiben | Int 10-13 |
|---|---|
| **Aufruf**<br>AH = 13<br>AL = 0...3 Unterfunktion<br>ES:BP = Zeiger auf String<br>CX = Länge String<br>DX = Cursorposition<br>BH = Seite | **Ergebnis**<br>String ab Cursorposition:<br>bei AL=0,1, mit BL-Attrib.<br>bei AL=2,3, indiv. Attrib.<br>bei AL=0,2, Cursor fest,<br>sonst Cursor mitbewegt |

Bei den Unterfunktionen gelten folgende Festlegungen:

AL = 0 Cursor nicht bewegen, Attribut in BL
AL = 1 Cursor weiterbewegen
AL = 2 String enthält abwechselnd Zeichen/Attribut
AL = 3 wie AL = 2, Cursor wird weiterbewegt.

### 4.3.3. Disketten- und Festplattenfunktionen

In den Abschnitten 2.5. und 2.6. sind einige Angaben zur physischen Struktur der externen Speicher Diskette und Festplatte gemacht worden. Für die Programmierung ist die logische Struktur von Bedeutung. Organisationseinheit im BIOS ist der Sektor, der durch eine dreidimensionale Koordinate bestimmt wird (Zylindernummer, Kopfnummer, Sektornummer). Es ist zu beachten, daß das DOS im Unterschied zum BIOS eine fortlaufende Nummer zur Adressierung eines Sektors verwendet.

Das BIOS benötigt zur korrekten Funktion einige Angaben zu technischen Parametern des Disketten- oder Festplattenlaufwerks. Diese Angaben sind in der Laufwerksparametertabelle zusammengefaßt. Eine Version dieser Tabelle steht im ROM. Bei der Systeminitialisierung wird sie in den RAM kopiert und durch DOS modifiziert. Die Anfangsadresse wird in den Interruptvektor 30 (1Eh) eingetragen. Dieser Vektor kann von jedem Anwendungsprogramm auf eine neue Tabelle eingestellt werden. In Tafel 4.7 ist der Aufbau der Diskettenparametertabelle gezeigt. Die angegebenen Standardwerte sind von der Version des ROM-BIOS abhängig. Durch Modifikation einzelner Werte kann bei vielen Diskettenlaufwerken eine wesentliche Steigerung des Datentransfer erreicht werden. Es kann vorkommen, daß bei zu kurzen Werten z.B. für die Motorstepzeit die Disketten nicht mehr auf allen PC gelesen werden können. Bei der Bearbeitung von Disketten, die nicht im Format von MS-DOS beschrieben wurden, sind z.B. die Sektorlänge und die Sektoranzahl zu verändern.

**Tafel 4.7. Diskettenparametertabelle (Adresseninterrupt 1Eh)**

| Byte | Bedeutung (FDC floppy disk controller) |
|---|---|
| 0 | Steuerbyte 0 für FDC (Motorsteprate, Kopfschreibzeit) |
| 1 | Steuerbyte 1 für FDC (Kopfladezeit, DMA-Modus) |
| 2 | Motornachlaufzeit (Standard = 25h, d.h. etwa 2 Sekunden) |
| 3 | Sektorlängencode (0=128, 1=256, 2=512, 3=1024) |
| 4 | Anzahl Sektoren in einer Spur (Standard 9) |
| 5 | Lücke zwischen den Sektoren (Standard = 2Ah) |
| 6 | max. Datentransferlänge, wenn Sektorlänge fehlt |
| 7 | Länge der magn. Lücke beim Formatieren (Standard = 50h) |
| 8 | Bytewert bei der Formatierung (Standard = F6h) |
| 9 | Kopfruhezeit (Standard = 25) |
| 10 | Motoranlaufzeit in 1/8 Sekunden (Standard = 4) |

Für Festplatten existieren zwei weitere Tabellen, deren Anfangsadressen in den Interruptvektoren 41h und 46h gespeichert sind. Diese Tabellen haben einen anderen Aufbau als die Diskettentabellen. Einige Parameter lassen sich durch BIOS-Funktionen feststellen. Die Tabellen sollten nicht modifiziert werden. Die Integration fremder Festplattenlaufwerke realisiert man am besten mit den Programmen, die vom Hersteller mitgeliefert werden (z.B. SpeedStor für Laufwerke von MAXTOR).

Für Disketten und Festplatten sind die in Tafel 4.8 genannten BIOS-Funktionen realisiert.

**Tafel 4.8. BIOS-Funktionen für Disketten und Festplatten**

| Nr. (hex.) | Funktion |
|---|---|
| 0 (00h) | Diskettenlaufwerke zurücksetzen |
| 1 (01h) | Diskettenstatus ermitteln |
| 2 (02h) | Sektor lesen (read) |
| 3 (03h) | Sektor schreiben (write) |
| 4 (04h) | Sektor überprüfen (verify) |
| 5 (05h) | Spur formatieren |

Für den PC/AT sind weitere Funktionen vorhanden, die hier nicht beschrieben werden, da sie auf sehr spezielle Geräteeigenschaften abgestimmt sind. Informationen dazu findet man im Technischen Handbuch zum AT (ROM-BIOS-Listing).

Bei allen Funktionen ist die Verwendung der Register gleich. Die Funktionen werden darum nicht einzeln aufgeführt. Es sei noch einmal darauf hingewiesen, daß im BIOS eine andere Form der Adressierung eines Sektors als bei den DOS-Funktionen erfolgt.

| **Aufruf** | **Ergebnis** |
|---|---|
| AH = Funktionsnummer<br>AL = Anzahl Sektoren<br>DL = Laufwerksnummer<br>DH = Kopfnummer<br>CH = Spurnummer<br>CL = Sektornummer<br>ES:BX = Pufferanfang | CF = 0 kein Fehler, AH = 0<br>CF = 1 Fehler, AH = Status |

Das **Rücksetzen des Diskettenlaufwerks** ändert nicht den Disketteninhalt; es wird lediglich ein definierter Anfangszustand hergestellt, der z.B. bei der Befehlswiederholung nach fehlerhaften Operationen notwendig ist.

Nach jedem Diskettenzugriff wird mit CF angezeigt, ob ein Fehler aufgetreten ist. Mit der Routine **Diskettenstatus ermitteln** (AH = 1) wird in AL die Fehlerart bereitgestellt. Tafel 4.9 erläutert die einzelnen Bit.

**Tafel 4.9. Statusbyte bei Diskettenoperationen**

| Wert | Fehlerart |
|---|---|
| 00h | kein Fehler |
| 01h | ungültiger Befehl |
| 02h | Adreßmarke (Sektorkennung) nicht gefunden |
| 03h | Schreibfehler (Diskette schreibgeschützt) |
| 04h | Sektor nicht gefunden oder nicht magnetisierbar |
| 08h | DMA-Fehler |
| 09h | DMA-Grenze von 64 KByte überschritten |
| 10h | CRC-Prüfsummenfehler |
| 20h | Controller defekt |
| 40h | Seek-Fehler (Spur nicht gefunden) |
| 80h | Laufwerk reagiert nicht (Time-Out) |

Beim **Lesen** können ein oder mehrere Sektoren in einen Puffer eingelesen werden. Die Puffergröße muß so gewählt werden, daß alle Sektoren nacheinander untergebracht werden können. Beim **Schreiben** wird ein Puffer auf ein oder mehrere Sektoren übertragen. Sowohl beim Lesen als auch beim Schreiben ist zu beachten, daß evtl. die Motoranlaufzeit abgewartet werden muß. Bei Fehlern ist die Operation mehrfach zu wiederholen. Beim **Überprüfen** werden die angegebenen Sektoren gelesen und ihre CRC-Summe getestet. Beim Formatieren wird immer eine ganze Spur behandelt. Im Pufferbereich stehen dazu die Sektorkennungen als Einträge von 4 Byte Länge zur Verfügung. Die Sektorkennungen werden bei Disketten fortlaufend vergeben; bei Festplatten wird in der Regel ein Sektorversatz eingestellt, um den Zugriff zu beschleunigen. Der

Controller übernimmt die Sektorkennungen unverändert in die Sektoradreßmarke. Beim Lesen und Schreiben sucht der Controller nach den Sektorkennungen auf einer eingestellten Spur. Durch falsche Eintragungen lassen sich spezielle Sektoren einrichten, die z.B. für den Kopierschutz genutzt werden können. Wichtige Laufwerksparameter sind in der Diskettenparametertabelle enthalten.

### 4.3.4. Tastaturfunktionen

Im Abschn. 2.4. wurde die physische Verbindung zwischen Tastatur und Rechner kurz beschrieben. Das BIOS behandelt die über Interrupt 9 signalisierte Tastenbetätigung, indem am Port 60h der Scan-Code (vgl. Tafel 2.4) gelesen und in einen Tastencode mit einer Länge von 2 Byte umgewandelt wird. Im höheren Byte wird der zur Taste gehörende ASCII-Wert gespeichert und im unteren Byte der Scan-Code. Der Code wird in Abhängigkeit von Umschalt- und Sondertasten gebildet. Zu diesen Tasten gehören

- **Umschalttaste links** (Shift links)
- **Umschalttaste rechts** (Shift rechts)
- **Festumschalttaste** (Caps-Lock-Taste)
- **Ctrl-Taste**
- **Alt-Taste**
- **Num-Lock-Taste**
- **Ins-Taste.**

Der für die Umwandlung erforderliche Tastaturstatus wird in den RAM-Positionen 417h und 418h abgelegt. Der Aufbau ist in Tafel 4.10 erläutert. Eine Manipulation der Bit ist nicht zu empfehlen. Die normalen Umschalttasten (Shift) gelten für die obere Belegung der Tastenreihen. Die Caps-Lock-Taste gilt nur für die Buchstabentasten und kehrt deren Belegung um. Die Num-Lock-Taste schaltet den Cursorblock auf Zahlenbelegung um.

**Tafel 4.10. Tastaturstatusbytes**

| Bit **Statusbyte 1** (417h) | Bit **Statusbyte 2** (418h) |
|---|---|
| 7 Ins (Einfügen) aktiv | 7 Ins gedrückt |
| 6 Caps Lock aktiv | 6 Caps Lock gedrückt |
| 5 Num Lock aktiv | 5 Num Lock gedrückt |
| 4 Scroll Lock aktiv | 4 Scroll Lock gedrückt |
| 3 Alt aktiv | 3 Ctrl-Num Lock aktiv |
| 2 Ctrl aktiv | 2 - |
| 1 Shift links gedrückt | 1 - |
| 0 Shift rechts gedrückt | 0 - |

Einige Tastenkombinationen haben eine besondere Bedeutung und führen zu direkten Reaktionen des BIOS.

**<Ctrl-Break>** (<Ctrl-C>, <Ctrl-Scroll Lock>) bewirkt die Aktivierung des Interrupts 1Bh. Die normale Reaktion des DOS ist der Abbruch des aktiven Programms. Der Interrupt kann durch eigene Behandlungsroutinen ersetzt werden.

**<Ctrl-Alt-Del>** bewirkt den Aufruf des Interrupts 19h und führt zum Neustart des Systems. Die POST-Routinen mit Hardwareüberprüfungen werden nicht aufgerufen; der Neustart geht dadurch etwas schneller als beim Einschalten. Die Routine kann nur aktiviert werden, wenn Interrupts zugelassen sind (Maschinenbefehl STI). Sollte diese Tastenkombination zu keinem Erfolg führen, so ist der PC abzuschalten und neu zu starten.

**<Ctrl-Num Lock>** bewirkt eine Pause in der Programmbearbeitung, bis eine Zeichentaste gedrückt wird. In diesem Modus sind Interrupts durch andere externe Ereignisse möglich und werden abgearbeitet (z.B. eine durch Interrupt gesteuerte Kommunikation über serielle Schnittstellen, Druckerspooler u.a.).

**<Shift-PrtSc>** bewirkt den Aufruf von Interrupt 05h und führt zu einem Bildschirmausdruck (hardcopy). Die Interruptroutine kann durch eine eigene Routine ersetzt werden. Zum System gehört das Programm GRAPHICS.COM, das zum Ausdruck des Bildschirms im Graphikmodus dient. Eine eigene Routine ist z.B. auch für den Bildschirmausdruck beim MGA-Adapter (Hercules) erforderlich.

Durch eine Kombination der Alt-Taste mit einer dreistelligen Zahl, die über den Nummernblock eingegeben wird, ist die Erfassung aller 256 Zeichen des erweiterten ASCII-Codes möglich. Der Nummerncode ergibt sich aus der Stellung des Zeichens in der Tabelle (vgl. Anhang). Der Scan-Code im höheren Byte des Tatstaturcodes ist in diesem Fall 00h. Bei der Tastatureingabe mit Turbo-Pascal (Version 3.0) werden die Sondertasten durch ein ESC-Zeichen (1Bh) eingeleitet.

Die Bereitstellung mehrerer Umschalttasten erhöht die Anzahl möglicher Tastaturcodes erheblich. Für den PC sind 97 **Sondertasten** definiert, denen kein Zeichen des erweiterten ASCII-Codes zugeordnet ist, die aber für spezielle Aktionen genutzt werden können. Bei den Sondertasten ist das untere Byte des Tastencode 00h. Im Anhang befindet sich eine Aufstellung aller Tastenkombination und Sondertasten mit den zugordneten Tastencodes.

Einige Zeichen sind auf der Tastatur doppelt belegt (Shift-Taste, '*', Zahlen, '-', '+'). Ist für die Auswertung im Programm wesentlich, welche der Tasten benutzt wurde, so ist zusätzlich zum ASCII-Zeichen der Scan-Code im höheren Byte des Tastaturcodes zu berücksichtigen.

Der Inhalt des Tastaturpuffers (RAM-Bereich 41Eh bis 43Dh) wird durch Tastaturfunktionen des BIOS bereitgestellt, die durch

Interrupt 16h aufgerufen werden. Es sind die in Tafel 4.11 genannten Tastaturfunktionen realisiert.

**Tafel 4.11. BIOS-Tastaturfunktionen**

| Nr.(hex.) | Funktion |
|---|---|
| 0 (00h) | Tastencode aus Tastaturpuffer lesen |
| 1 (01h) | Status Tastaturpuffer feststellen |
| 2 (02h) | Umschaltstatus feststellen |

| **Tastencode aus Tastaturpuffer lesen** | **Int 16-00** |
|---|---|
| **Aufruf**<br>AH = 00h | **Ergebnis**<br>AL = ASCII-Zeichen<br>(AL = 0 Sondertaste)<br>AH = Scan-Code<br>(AH = 0 Eingabe mit Alt) |

Wenn kein Tastencode im Puffer vorliegt, wartet die Routine auf die nächste Eingabe. Will man das Warten vermeiden, so ist vorher der Tastaturstatus abzufragen (Funktion 01h).

| **Status Tastaturpuffer feststellen** | **Int 16-01** |
|---|---|
| **Aufruf**<br>AH = 01h | **Ergebnis**<br>ZF = 0 Tastencode in AX<br>(AL, AH wie bei Funkt. 00) |

Im Unterschied zu Funktion 16-00 wird nicht auf das nächste Zeichen gewartet, sondern nur das ZF-Bit im Flagregister gesetzt. Der Tastaturpuffer wird nicht verändert.

| **Umschaltstatus feststellen** | **Int 16-02** |
|---|---|
| **Aufruf**<br>AH = 02h | **Ergebnis**<br>AL = Statusbyte (417h) |

Mit dieser Routine läßt sich ermitteln, welche von den Umschalttasten betätigt wurde. Durch Änderungen der betreffenden Bit (Tafel 4.10) läßt sich z.B. die Bedeutung von Tasten verändern.

### 4.3.5. Funktionen für die serielle Schnittstelle

Der PC wird mit einer oder mehreren Schnittstellen (COMn) ausgeliefert. Diese Schnittstellen sind für asynchrone Übertragungen (z.B. Rechnerkopplungen, Modemanschluß, seriellen Drucker) geeignet. Der verwendete Baustein für die Übertragung ist programmierbar. Es lassen sich eine ganze Reihe von Übertragungsparametern einstellen. Die **Übertragungsgeschwindigkeit** wird in Baud (Bit/Sekunde) angegeben. Die Baudrate ist vom Übertragungsweg abhängig. Bei Modemübertragungen ist ein Wert von 1200 oder 2400 Baud erreichbar. Bei direkten Kopplungen ist der Standardwert 9600 und kann in einigen Fällen bis auf 19200 Baud gesteigert werden. Weitere Parameter sind die **Parität**, die **Anzahl Datenbits** und die **Anzahl Stopbit**. Die Einstellung muß beim Sender und Empfänger gleich sein, damit eine Übertragung zustande kommt.

Die in Tafel 4.12 genannten Funktionen sind realisiert und können durch Interrupt 14h aufgerufen werden.

**Tafel 4.12. BIOS-Funktionen für die serielle Schnittstelle**

| Nr.(hex.) | Funktion |
|---|---|
| 0 (00h) | Parameter für serielle Schnittstelle |
| 1 (01h) | Zeichen senden |
| 2 (02h) | Zeichen empfangen |
| 3 (03h) | Status ermitteln |

Die Parameter werden in einem Byte verschlüsselt, das folgenden Aufbau hat:

Bit
765 43 2 10

| Baudrate | Parität | Stopbit | Datenbit |
|---|---|---|---|
| 000 110 | 00 keine | 0 1 | 00 - |
| 001 150 | (none) | 1 2 | 01 - |
| 010 300 | 01 ungerade | | 10 7 |
| 011 600 | (odd) | | 11 8 |
| 100 1200 | 10 keine | | |
| 101 2400 | (none) | | |
| 110 4800 | 11 gerade | | |
| 111 9600 | (even) | | |

Der Baustein 8250 bzw. ein entsprechender Baustein, wie z.B. USART 8251A, der für den Datentransfer verantwortlich ist, kann auch mit höherer Baudrate betrieben werden. Für Parameter, die

nicht in der Tabelle angegeben sind, muß der Baustein direkt programmiert werden (vgl. Technisches Handbuch).

Die Schnittstellenfunktionen liefern bei der Statusabfrage und im Fehlerfall eine Information über den aktuellen Zustand an den Ein- und Ausgängen, die folgenden Aufbau hat:

| **Statusanzeigen bei COM-Funktionen** | |
|---|---|
| Bit in AH (Leitungsstatus) | BIT in AL (Modemstatus) |
| 7 Time-out-Fehler | 7 RLSD (receive line sign) |
| 6 Transfer-Shift leer | 6 RI |
| 5 Transfer-Hold leer | 5 DSR (data set ready) |
| 4 Unterbrechungsfehler | 4 CTS (clear to send) |
| 3 Framingfehler | 3 DRLSD (delta-RLSD) |
| 2 Paritätsfehler | 2 TERI |
| 1 Überlauf | 1 DDSR (delta-DSR) |
| 0 Daten bereit | 0 DCTS (delta-CTS) |

Die vier definierten COM-Funktionen bieten wenig Programmierkomfort. Eine Pufferung der Zeichen wird nicht durchgeführt. Bei der Programmierung der Schnittstellen sind daher besondere Übertragungsprotokolle zu realisieren, damit ein sicherer Ablauf gewährleistet ist. Bei Verwendung eines Modems bieten sich die Modemsignale zur Steuerung an. Sehr verbreitet ist auch das Xon/Xoff-Protokoll, bei dem mit Xoff (13h) der Empfänger signalisiert, daß er kein Zeichen abnehmen kann. Mit Xon (11h) wird die Empfangsbereitschaft signalisiert. Eine durch Interrupt gesteuerte Behandlung der COM-Schnittstelle mit Pufferung der Zeichen wird z.B. im Programmpaket Turbo-Talk (Lauer&Wallwitz) oder Turbo-Asynch (Blaise) realisiert. Die COM-Schnittstelle wird z.B. auch vom Kommunikationsprogramm KERMIT benutzt, das eine eigene Übertragungsroutine realisiert.

| **Initialisierung der COM-Schnittstelle** | **Int 14-00** |
|---|---|
| **Aufruf** | **Ergebnis** |
| AH = 00h | AH = Leitungsstatus |
| AL = Parameterbyte | AL = Modemstatus |
| DX = Port | |

| **Zeichen senden** | **Int 14-01** |
|---|---|
| **Aufruf** | **Ergebnis** |
| AH = 01h | AH = Leitungsstatus |
| AL = Zeichen | (Fehler, wenn Bit7=1) |
| DX = Port | |

Vor Aufruf der Funktion sollte die Sendebereitschaft durch

Aufruf von Funktion 03 (Modemstatus) festgestellt werden, weil sonst der Aufruf mit Fehler beendet wird.

| **Zeichen empfangen (warten)** | **Int 14-02** |
|---|---|
| **Aufruf**<br>AH = 02h<br>DX = Port | **Ergebnis**<br>AH = Leitungsstatus<br>AL = Zeichen |

Im Fehlerfall ist durch Aufruf von Funktion 03 die Fehlerursache zu bestimmen.

| **COM-Status ermitteln** | **Int 14-03** |
|---|---|
| **Aufruf**<br>AH = 03<br>DX = Port | **Ergebnis**<br>AH = Leitungsstatus<br>AL = Modemstatus |

### 4.3.6. Druckerfunktionen

Die Ansteuerung des Druckers ist im Vergleich zu den anderen Geräten relativ einfach. Als Druckerausgang wird eine Parallelschnittstelle angenommen. Eine Umleitung des Druckers auf eine serielle Schnittstelle wird durch spezielle Programme ermöglicht, die an den Druckerinterrupt gebunden werden, z.B. DOS-Kommando MODE. Es sind 3 Druckerfunktionen vorgesehen (Tafel 4.13).

**Tafel 4.13. BIOS-Funktionen zur Druckersteuerung**

| Nr. (hex) | Funktion |
|---|---|
| 0 (00h) | ein Byte an Drucker senden |
| 1 (01h) | Drucker initialisieren |
| 2 (02h) | Druckerstatus ermitteln |

Nach jeder Funktion steht im Register AH der **Druckerstatus**, wobei die einzelnen Bits die folgende Bedeutung haben:

| Bit | **Druckerstatus** |
|---|---|
| 7 | Drucker nicht aktiv (not busy) |
| 6 | Bestätigung vom Drucker (ACK) |
| 5 | Papierende |
| 4 | Drucker ausgewählt |
| 3 | Ausgabefehler |
| 2 | - |
| 1 | - |
| 0 | Time-out |

Alle Druckerfunktionen werden über den Interrupt 17h aufgerufen. Als Druckernummer ist die Nummer der Parallelschnittstelle anzugeben (0 = LPT1, 1 = LPT2, ...).

| **Ein Byte an Drucker senden** | **Int 17-00** |
|---|---|
| **Aufruf**<br>AH = 00h<br>AL = Zeichen<br>DX = Druckernummer | **Ergebnis**<br>AH = Druckerstatus |

| **Drucker initialisieren** | **Int 17-01** |
|---|---|
| **Aufruf**<br>AH = 01h<br>DX = Druckernummer | **Ergebnis**<br>AH = Druckerstatus |

| **Druckerstatus ermitteln** | **Int 17-03** |
|---|---|
| **Aufruf**<br>AH = 02h<br>DX = Druckernummer | **Ergebnis**<br>AH = Druckerstatus |

Der Interrupt 17h kann durch Anwendungsprogramme belegt werden, die eine eigene Druckeranpassung durchführen. In einer solchen Routine ist auch eine komfortable Fontbehandlung möglich, wie sie z.B. durch das Programm LETTRIX realisiert wird.

### 4.3.7. Sonstige BIOS-Funktionen

In diesem Abschnitt werden die restlichen BIOS-Interrupts zusammengefaßt, die nicht einem speziellen Gerät zugeordnet werden können (Tafel 4.14). Sie dienen einerseits der Anzeige einiger Systemparameter, und andererseits lösen sie spezielle Funktionen aus. Einige Interrupts enthalten nur Adressen und verweisen auf wichtige Systemtabellen; sie heißen darum Adreßinterrupts und werden bei den entsprechenden Funktionen mitbehandelt. Die Funktionen zur Ansteuerung eines Kassettenrecorders werden in diesem Buch nicht behandelt, da bei den meisten PC dieses Gerät nicht unterstützt wird. Ebenso wird auf das ROM-Basic nicht eingegangen.

**Tafel 4.14. Sonstige BIOS-Funktionen**

| Nr.(hex.) | Funktion |
|---|---|
| 5 (05h) | Bildschirmdruckroutine |
| 17 (11h) | Ausstattungsliste lesen |
| 18 (12h) | Speichergröße feststellen |
| 25 (19h) | Urlader aktivieren (Bootprogramm) |
| 26 (1Ah) | Uhrzeit |
| 27 (1Bh) | Tastaturunterbrechung |
| 28 (1Ch) | Zeitgeberunterbrechung |

Die **Bildschirmdruckroutine (Int 05h)** wird vom BIOS aufgerufen, wenn die Tastenkombination <Shift-PrtSC> (print screen) eingegeben wird. In der Routine werden die Standardfunktionen zum Lesen der Daten aus dem Bildschirmpuffer und die BIOS-Druckerfunktion für LPT1 aufgerufen. Im RAM wird auf Adresse 500h ein Status eingetragen. Während des Druckens ist ein weiterer Aufruf der Routine nicht möglich. Das DOS-Programm GRAPHICS.COM wird an diesen Interrupt angebunden, damit ein Graphikausdruck des Bildschirms erzeugt werden kann.

Im RAM-Bereich des BIOS ist auf Adresse 410h die **Ausstattungsliste** abgelegt, die beim Starten des PC durch die POST-Routine aufgebaut wird. Durch **Interrupt 11h** wird diese Liste in Register AX bereitgestellt und kann im Programm ausgewertet werden. Die Angaben sind für einen voll ausgebauten PC unvollständig; z.B. reichen die Bit zur Angabe des Bildschirmmodus nicht aus, und die Angaben zu Festplatten fehlen ganz. Die Angaben ergeben sich im wesentlichen aus der Stellung der Konfigurationsschalter auf der Hauptplatine und aus den angeschlossenen COM-Ports (UART 8250) und Druckerports.

| **Ausstattungsliste (Register AX)** | **Int 11** |
|---|---|
| Bits von AH | Bits von AL |
| 7...6 Anzahl Drucker | 7...6 Anzahl Diskettenlw. |
| 5 nicht benutzt | 5...4 Bildschirmmodus |
| 4 Spieleadapter | 3...2 Speicher Hauptplat. |
| 3...1 Anzahl COM-Ports | 1 unbenutzt |
| 0 nicht benutzt | 0 Diskettenlw. vorhand. |

Die **Speichergröße** (RAM-Bereich 413h) wird durch **Interrupt 12h** im Register AX bereitgestellt. Die Angabe erfolgt in KByte. Die Speichergröße wird aus der Stellung von Schaltern auf der Hauptplatine ermittelt. Der Wert muß nicht mit der tatsächlichen Größe übereinstimmen.

Das Urladerprogramm wird durch **Interrupt 19h** aufgerufen und führt zum gleichen Ergebnis wie die Tastenkombination <Ctrl-Alt-

Del>. Die POST-Routine wird nicht ausgeführt. Dadurch ist ein Neustart wesentlich schneller. DOS arbeitet dabei mit den Werten, die im RAM-Bereich durch das POST-Programm abgelegt wurden bzw. die durch Anwenderprogramme modifiziert wurden. Eine sinnvolle Anwendung des Interrupts 19h ist z.B. der Neustart bei offensichtlicher Verletzung des Kopierschutzes. Eine Speicheranalyse des Programms ist dann nach dem Neustart nicht mehr möglich.

Die **Uhrzeit** wird im BIOS durch einen Zeitgeber gesteuert, der einen Zähler von 4 Byte (RAM-Adresse 46Ch) bei jedem Zeitgeberinterrupt (Int 08h) im Zeittakt erhöht. Wenn die aktuelle Uhrzeit (**Interrupt 1Ah**) angefordert wird, vergleicht das BIOS den aktuellen Zählerstand mit dem Wert für Mitternacht, der auf Speicherstelle 470h gespeichert ist. Wenn der aktuelle Wert größer ist, so wird der Zähler wieder auf den Wert 0 gestellt und in einem Anzeigefeld (RAM-Adresse 471h) eine Anzeige gesetzt. Wenn DOS die Uhrzeit benötigt, wird aus dem Zählerstand die Uhrzeit berechnet und bei Zählerüberlauf das Datum neu eingestellt. Die Überlaufanzeige wird dabei zurückgesetzt.

| **Uhrzeit - Zählerstand lesen** | **Int 1A-00** |
|---|---|
| **Aufruf**<br>AH = 00h | **Ergebnis**<br>AL > 0 Zählerüberlauf<br>CX = Zähler oberer Teil<br>DX = Zähler unterer Teil |

| **Uhrzeit - Zählerstand setzen** | **Int 1A-01** |
|---|---|
| **Aufruf**<br>AH = 01h<br>CX = Zähler oberer Teil<br>DX = Zähler unterer Teil | **Ergebnis**<br>neuer Zählerstand |

Für den AT sind eine Reihe weiterer Funktionen realisiert, die direkt das Datum verwalten und auch eine Weckfunktion enthalten. Auf diese Routinen wird hier nicht näher eingegangen, da sie nicht auf jedem PC existieren.

Der **Tastaturinterrupt (Int 1Bh)** wird aufgerufen, wenn die Sondertastenkombination <Ctrl-Break> betätigt wird. Vom BIOS wird eine Routine ohne weitere Funktion angebunden. Vom DOS oder vom Anwenderprogramm werden hier eigene Routinen angebunden.

Der **Zeitgeberinterrupt (Int 1Ch)** wird bei jedem Zeitgebertakt aufgerufen. Mit diesem Interrupt läßt sich eine Zeitsteuerung z.B. für Multi-Tasking realisieren. Vom BIOS ist hier eine Dummy-Routine angebunden.

## 4.4. Nutzung der DOS-Funktionen

### 4.4.1. DOS-Funktionen (Übersicht)

Alle DOS-Systemdienste werden über Interrupts angefordert. Von den 256 möglichen Interrupts, die der Prozessor 8086/8088 zuläßt, sind die Interrupts 20h-3Fh für DOS reserviert. Die in Tafel 4.15 nicht angegebenen Interrupts sind entweder intern verwendet oder nicht belegt. Eine besondere Bedeutung hat der Interrupt 33 (21h), der für Programme die eigentliche Schnittstelle zum DOS darstellt. Die DOS-Funktionen lassen sich den einzelnen Aufgaben von DOS wie folgt zuordnen:

- **Standardeingabe und -ausgabe für zeichenorientierte Geräte**
- **Speicherverwaltung (memory management)**
- **Prozeßverwaltung (process management)**
- **Dateiverwaltung (file and directory management)**
- **Netzwerkfunktionen (Microsoft network)**
- **Systemverwaltung (system management).**

Die Liste der DOS-Funktionen spiegelt die Entwicklung von DOS über mehrere Versionen wider. In der Version 1.00 waren die Systemrufe zur Dateiverwaltung an CP/M angelehnt. In Version 2.00 erfolgte die Einführung neuer Dienste zur Dateiverwaltung, die sich stärker an UNIX bzw. XENIX anlehnten. Die alten Rufe wurden aus Gründen der Kompatibilität beibehalten, sollten für Neuentwicklungen aber nicht mehr benutzt werden. Die Systemaufrufe der Version 1.00 werden auch als traditionelle DOS-Funktionen und die ab Version 2.00 als erweiterte DOS-Funktionen bezeichnet. Diese Unterteilung ist für die Nutzung nicht von Bedeutung. Die Funktionsaufrufe werden in den folgenden Abschnitten entsprechend den genannten Aufgaben beschrieben. Eine nach Funktionsnummern geordnete Liste findet man im Anhang. Die Funktionen zur Dateiverwaltung werden im Abschn. 5. (Dateisystem) gesondert behandelt.

Alle DOS-Funktionen werden über den DOS-Interrupt 21h aufgerufen. Im Register AH wird der eigentliche Funktionscode übergeben. In den folgenden Beschreibungen der einzelnen Funktionen wird der Registerinhalt beim Aufruf und bei der Rückkehr (Ergebnis) angegeben. In Turbo-Pascal ist der Aufruf von Interrupts durch die Prozedur Intr(<Int>,<Registerrecord>) möglich. Für den Aufruf der DOS-Funktionen (Int = 21h) ist eine eigene Prozedur Msdos(<Registerrecord>) definiert. Bei der Verwendung einiger DOS-Funktionen sind spezielle Maßnahmen zur Sicherung der Register erforderlich. Auf diesen Umstand wird dann gesondert hingewiesen. Im Abschn. 11. ist eine Sammlung von Funktionen und Prozeduren enthalten, die die Nutzung der DOS-Dienste zeigt.

**Tafel 4.15. DOS-Interrupts (20h-3Fh)**

| Nr.(hex.) | Funktion |
|---|---|
| 32 (20h) | Programm beenden |
| 33 (21h) | DOS-Funktion aufrufen |
| 34 (22h) | Adresse Programmendebehandlung |
| 35 (23h) | Adresse Unterbrechungsbehandlung (Ctrl-C) |
| 36 (24h) | Adresse Fehlerbehandlung |
| 37 (25h) | Diskette/Festplatte lesen (absolut) |
| 38 (26h) | Diskette/Festplatte schreiben (absolut) |
| 39 (27h) | Programm speicherresident beenden |
| 47 (2Fh) | Druckerspooler (ab Version 3.00) |
| 51 (33h) | Mausschnittstelle |

Die von DOS genutzten Interrupts fallen für die systematische Beschreibung der DOS-Funktionen etwas aus dem Rahmen; sie werden darum an dieser Stelle kurz beschrieben. Die Interrupts 22h-24h sind keine echten Interrupts, sondern enthalten Adressen, die von allen Stellen des Systems aus erreichbar sind. Mit den Interrupts 25h und 26h ist eine Umgehung des Dateisystems von DOS möglich. Die Funktionen werden benötigt, wenn Datenträger bearbeitet werden sollen, die nicht dem DOS-Standard entsprechen. Der DOS-Interrupt 2Fh wurde mit der Version DOS 3.00 eingeführt. Die Realisierung dieser Funktion als Interrupt läßt eine zeitgesteuerte Behandlung des Druckers zu.

| **Programm beenden** | **Int 20** |
|---|---|
| **Aufruf**<br>- | **Ergebnis**<br>Prozeß wird beendet |

Dieser Interrupt wurde durch eine neue DOS-Funktion (4Ch) ersetzt und sollte nicht mehr genutzt werden. Vor dem Aufruf muß sichergestellt werden, daß das Register CS auf den Programmanfang (PSP, vgl. Abschn. 4.4.4.) zeigt. Offene Dateien sollten vorher mit der Funktion CLOSE abgeschlossen werden. Die Adressen für die Interrupts 22h-24h werden aus dem PSP zurückgespeichert.

| **DOS-Funktion aufrufen** | **Int 21-nn** |
|---|---|
| **Aufruf**<br>AH = nn (DOS-Funktion) | **Ergebnis**<br>abhängig von DOS-Funktion |

Dieser Interrupt ruft den Interpreter für DOS-Funktionen. Die einzelnen Funktionen werden in den folgenden Abschnitten genauer behandelt.

| **Adresse Programmendebehandlung** | **Int 22** |
|---|---|
| **Aufruf**<br>durch DOS bei Programmende | **Ergebnis**<br>zurück zum rufenden Prog. |

Das rufende Programm ist in der Regel COMMAND. Soll an einer definierten Stelle eines anderen rufenden Programms die Verarbeitung fortgesetzt werden, so ist die Adresse im Interrupt 22h so zu verändern, daß sie auf die gewünschte Stelle zeigt. Das Lesen und Ändern des Vektors sollte mit den DOS-Funktionen 35h (get interrupt vector) und 25h (set interrupt vector) erfolgen. Der beim Aufruf eines Programms gültige Wert für Int 22h wird im PSP (Offset 0Ah) gespeichert und bei Beendigung zurückgeladen.

| **Adresse Unterbrechungsbehandlung (Ctrl-C)** | **Int 23** |
|---|---|
| **Aufruf**<br>durch DOS bei Int 1Bh | **Ergebnis**<br>Programmende (Standard) |

Der Tastaturinterrupt 1Bh (Ctrl-C) wird von DOS nicht bei allen Funktionsaufrufen abgefragt. Der Test kann durch das Kommando BREAK verändert werden. Die Standardroutine im DOS kann durch eine eigene Routine ersetzt werden (Nutzung der Funktionsaufrufe 25h und 35h, vgl. Abschn. 4.4.6.). Werden bei Einsprung in diese Routine alle Register gesichert, so kann die Beendigung mit dem Befehl IRET (interrupt return) erfolgen. Wird die Routine mit einem Befehl FAR RET verlassen, so bestimmt DOS anhand des CF, ob das Programm beendet (CF = 1) oder fortgesetzt (CF = 0) werden soll. Beim Einsprung in diese Routine haben die Register die Werte, die sie beim Aufruf von DOS hatten. Es können alle Funktionsaufrufe benutzt werden.

| **Adresse Fehlerbehandlung** | **Int 24** |
|---|---|
| **Aufruf**<br>durch DOS bei kritischen Fehlern | **Ergebnis**<br>Fehlermeldung und Abfrage (Abort, Retry, Ignore) |

Die Adresse der Fehlerbehandlungsroutine kann mit den DOS-Funktionen 25h und 35h auf eine eigene Fehlerbehandlung umgesetzt werden. Die Programmierung einer solchen Routine ist nicht sehr einfach, weil eine ganze Reihe von Bedingungen eingehalten werden müssen. Das DOS verzweigt in diese Routine, nachdem es bereits mehrere Male versucht hat, den Fehler zu beheben. Beim Eintritt in die Fehlerbehandlung sind Interrupts nicht zugelassen. Infor-

mationen, die zur Behandlung des Fehlers erforderlich sind, findet man in einem Gerätekontrollblock (Adresse BP:SI), auf dem Stack, im Register AX und Register DI. Der Gerätekontrollblock (device driver header) beschreibt das Gerät, bei dem der Fehler aufgetreten ist. Das Stackregister verweist auf die Registerwerte beim Aufruf von DOS. Ist AH < 128, stammt der Fehler von einem Laufwerk (Blockgerät). In AL steht die Laufwerksnummer (0 = A:), in AH eine Information über den Bereich auf dem Datenträger und die Art des Zugriffs. Ist AH > 127, so ist der Fehler entweder bei einem zeichenorientierten Gerät aufgetreten (Gerätetyp steht im Gerätesteuerblock), oder die FAT (file allocation table) des Datenträgers ist defekt. Der Fehlercode steht in DI und kann mit der DOS-Funktion 59h näher bestimmt werden. Bei der Rückkehr zu DOS wird in AL ein Code bereitgestellt, der angibt, ob der Fehler ignoriert (AL= 0), die Operation wiederholt (AL = 1) oder das Programm beendet werden soll (AL = 2).

| **Diskette/Festplatte lesen (absolut)** **Int 25**<br>**Diskette/Festplatte schreiben (absolut) Int 26** | |
|---|---|
| **Aufruf**<br>AL = Gerätenummer<br>DS:BX = Puffer (DTA)<br>CX = Anzahl Sektoren<br>DX = Startsektor | **Ergebnis**<br>AL = Fehlercode (DOS)<br>AH = Fehlercode (BIOS)<br>CF = 0 kein Fehler<br>CF = 1 Fehler |

Die Routinen sind den entsprechenden BIOS-Routinen sehr ähnlich. Die Sektoradressierung ist jedoch unterschiedlich. Während im BIOS als Adresse eine Angabe mit Zylinder, Kopf und Sektor auf der Spur erfolgt, wird von DOS eine sequentielle Zählung vom Beginn des Datenträgers durchgeführt. Es ist zu beachten, daß DOS das Flagregister nicht vom Stack entfernt. Beim Lesen mehrerer Sektoren muß der Pufferbereich groß genug gewählt werden. Diese Interrupts sollten nur in Ausnahmefällen benutzt werden.

| **Programm speicheresident beenden** **Int 27** | |
|---|---|
| **Aufruf**<br>CS:DX = Adresse Codeende+1 | **Ergebnis**<br>Programm resident beendet |

Das speicherresidente Programm hat eine maximale Größe von 64 KByte. Anstelle des Interrupts sollte die DOS-Funktion 31h (keep process) verwendet werden. Für EXE-Programme, die am oberen Speicherende liegen, ist dieser Interrupt nicht geeignet.

| Druckerspooler | Int 2F-nn |
|---|---|
| **Aufruf**<br>AL = 00h Status ermitteln | **Ergebnis**<br>AL = FFh installiert<br>= 00h nicht installiert<br>= 01h Int nicht frei |
| AL = 01h Datei übergeben<br>DS:DX = Adr. Dateiblock | Dateiname in Drucker-warteschlange eingetragen |
| AL = 02h Einträge entfernen<br>DS:DX = Adr. Dateinamen<br>(* und ? in Namen erlaubt) | Einzelne Dateieinträge werden entfernt |
| AL = 03h alle Dateieinträge entfernen | Ausdruck wird abgebrochen<br>Druckerwarteschlange leer |
| AL = 04h Druck anhalten | DS:DI = Adr. Warteschlange |
| AL = 05h Druck fortsetzen | Ausdruck wird fortgesetzt |

Der bei Funktion 01h übergebene Dateiblock hat eine Länge von 5 Byte. Das 1. Byte enthält einen Ebenencode, auf den eine segmentierte Adresse folgt. Die Adresse verweist auf einen Dateinamenstring, der mit 00h beendet ist (ASCIIZ-String). Die Angabe von '*' und '?' ist nicht erlaubt. Die Einträge in der Druckerwarteschlange sind 64 Byte lang und enthalten vollständige Dateinamen. Der letzte Eintrag beginnt mit 00h. Der 1. Eintrag verweist auf die aktuelle Druckdatei.

### 4.4.2. Ein- und Ausgabe für zeichenorientierte Geräte

Für zeichenweise Ein- und Ausgabe stellt DOS die folgenden Funktionen zur Verfügung, die z.T. BIOS-Funktionen entsprechen, jedoch auf einem etwas höheren Niveau arbeiten. Die Geräte sind als Standard mit einer festen Nummer (handle) verbunden, die eine Eingliederung in das Dateisystem (vgl. Abschn. 5.6.) zulassen und damit eine einheitliche Behandlung von Geräten und Dateien gestatten. Die Zuordnung zu Tastatur, Bildschirm, Drucker und Hilfsport kann durch Umleitung (redirection) verändert werden. Die Funktionen für die Standardgeräte sind bereits in der Version 1.0 vorhanden und entsprechen weitgehend den CP/M-Funktionen.

**Tafel 4.16. Funktionen zur Standardeingabe**

| Standardeingabe (Tastatur) |
|---|
| 01h Zeichen von Standardeingabe lesen (mit Echo) |
| 08h Zeichen von Standardeingabe lesen (kein Echo) |
| 06h Zeichen direkt von Standardeingabe lesen |
| 07h Zeichen direkt von Standardeingabe lesen (Warten) |
| 0Bh Status Eingabepuffer testen |
| 0Ah String von Standardeingabe lesen |
| 0Ch Puffer leeren und Eingabefunktion ausführen |

**Tafel 4.17. Funktionen zur Standardausgabe**

| Standardausgabe (Bildschirm) |
|---|
| 02h Zeichen auf Standardausgabe schreiben |
| 06h Zeichen direkt auf Standardausgabe schreiben |
| 09h String auf Standardausgabe schreiben |

**Tafel 4.18. Funktion für Standarddrucker**

| Standarddrucker (PRN, LPT1) |
|---|
| 05h Zeichen auf Drucker schreiben |

**Tafel 4.19. Hilfsport für Hilfsport**

| Hilfsport (AUX, COM1) |
|---|
| 03h Zeichen von Hilfsport lesen |
| 04h Zeichen nach Hilfsport schreiben |

**Funktionen für Standardeingabe (CON)**

Die Standardeingabe ist normalerweise der Tastatur zugewiesen. Mit den 7 Funktionen zur Standardeingabe lassen sich alle notwendigen Eingabeformen realisieren. Zum Zugriff auf die Tastatur werden intern die BIOS-Routinen genutzt. Die etwas komplizierte Behandlung der Zeichen im BIOS wird vereinfacht. Es werden alle Zeichen im erweiteren ASCII-Code dargestellt. Bei Sondertasten wird beim Lesen eine Null (00h) geliefert. Der folgende Aufruf liefert den erweiterten Scan-Code, der bei den BIOS-Routinen bereits erläutert wurde.

| Zeichen von Standardeingabe lesen (Echo) Int 21-01 | |
|---|---|
| **Aufruf**<br>AH = 01h | **Ergebnis**<br>AL = Zeichen (ASCII) |

Eine Unterbrechung mit Ctrl-C (Aufruf Int 23h) ist möglich. Ist kein Zeichen im Zeichenpuffer (type ahead buffer) vorhanden, so wartet die Routine auf eine Eingabe. Das gelesene Zeichen wird sofort auf dem Standardausgabegerät ausgegeben (Echo). Diese Funktion ist sinnvoll, wenn der eingegebene Text gleichzeitig auf dem Bildschirm protokolliert werden soll.

| Zeichen von Standardeingabe lesen Int 21-08 | |
|---|---|
| **Aufruf**<br>AH = 08h | **Ergebnis**<br>AL = Zeichen (ASCII) |

Eine Unterbrechung mit Ctrl-C ist möglich. Ist kein Zeichen im Zeichenpuffer, so wartet die Routine auf eine Eingabe. Im Unterschied zu Routine 01 wird kein Echo erzeugt. Diese Routine ist z.B. bei der Eingabe von Zugriffsschlüsseln (password) sinnvoll, wobei die eingegebenen Zeichen nicht erkannt werden sollen.

| Zeichen direkt von Standardeingabe lesen Int 21-06 | |
|---|---|
| **Aufruf**<br>AH = 06h<br>DL = FFh (Kennung f. Lesen) | **Ergebnis**<br>ZF = 1 Puffer leer<br>ZF = 0 Zeichen gelesen<br>AL = Zeichen (ASCII) |

Der Unterschied zu Routine 08 besteht darin, daß eine Unterbrechung mit Ctrl-C nicht möglich ist und daß auf die Eingabe nicht gewartet wird. Die Unterscheidung, ob ein Zeichen gelesen ist oder nicht, erfolgt durch Auswertung von ZF (zero flag).

| Zeichen direkt von Standardeingabe lesen (Warten) 21-07 | |
|---|---|
| **Aufruf**<br>AH = 07h | **Ergebnis**<br>AL = Zeichen |

Der Unterschied zu Routine 08 besteht darin, daß eine Unterbrechung mit Ctrl-C nicht möglich ist.

| **Status Eingabepuffer testen** | **Int 21-0B** |
|---|---|
| **Aufruf**<br>AH = 0Bh | **Ergebnis**<br>AL = 00h Puffer leer<br>AL = FFh Zeichen vorhanden |

Diese Funktion testet den Pufferstatus, ohne ihn zu verändern.

| **String von Standardeingabe lesen** | **Int 21-0A** |
|---|---|
| **Aufruf**<br>AH = 0Ah<br>DS:DX = Adr. Puffer | **Ergebnis**<br>Zeile nach Puffer gelesen<br>(einschließlich <cr> ) |

Der Puffer kann eine variable Länge haben. Die Maximallänge steht im 1. Byte des Puffers. Nach dem Aufruf steht die aktuelle Länge im 2. Byte. Der Zeichenstring steht ab 3.Byte. Bevor der Eingabestring nicht mit <cr> abgeschlossen ist, können die üblichen Editiertasten zur Korrektur in der Eingabezeile benutzt werden.

| **Puffer leeren, Eingabefunktion ausführen** | **Int 21-0C** |
|---|---|
| **Aufruf**<br>AH = 0Ch<br>AL = Funktions-Nr.<br>(weitere Angaben entsprechend Funktions-Nr.) | **Ergebnis**<br>entsprechend Funktions-Nr. |

Diese Funktion gestattet das Löschen des Eingabepuffers, bevor eine der Funktionen mit Warten auf Eingabe (01h, 06h, 07h, 08h, 0Ah) ausgeführt wird. Das Leeren des Puffers kann z.B. erforderlich sein, um einen definierten Zustand nach falscher Bedienung einzustellen.

**Funktionen für Standardausgabe (CON)**

Die Standardausgabe ist normalerweise dem Bildschirm zugeordnet. Die Funktionen zur Standardausgabe sind wesentlich einfacher als die Bildschirmfunktionen des BIOS. Der Datenaustausch erfolgt lediglich in Form von Einzelzeichen; der Zugriff auf den Bildschirmspeicher ist im DOS nicht definiert. Man kann zwar mit dem speziellen Bildschirmtreiber ANSI.SYS den Cursor auf dem

Bildschirm positionieren, eine direkte Graphikprogrammierung ist damit jedoch nicht möglich. Graphikprogramme und andere Programme, die eine schnelle Ausgabe benötigen, arbeiten daher mit den BIOS-Routinen.

| **Zeichen auf Standardausgabe schreiben** | **Int 21-02** |
|---|---|
| **Aufruf**<br>AH = 02h<br>DL = Zeichen | **Ergebnis**<br>Zeichen bei Cursorposition dargestellt |

Die Ausgabe von <Backspace> (08h) bewegt den Cursor eine Stelle nach links. Bei allen anderen ASCII-Zeichen wird der Cursor weiterbewegt (TTY-Modus). Nach der Ausgabe wird geprüft, ob eine Unterbrechung durch Ctrl-C vorliegt.

| **Zeichen direkt auf Standardausgabe schreiben** | **21-06** |
|---|---|
| **Aufruf**<br>**AH = 06h**<br>**DL = Zeichen** (00h...FEh) | **Ergebnis**<br>Zeichen bei Cursorposition dargestellt |

**Diese Funktion arbeitet wie Funktion 02; auf eine Unterbrechung mit Ctrl-C wird jedoch nicht reagiert. Die Ausgabe des ASCII-Wertes FFh ist nicht möglich, da die Funktion dann eine Eingabe realisiert (vgl. bei Standardeingabe).**

| **String auf Standardausgabe schreiben** | **Int 21-09** |
|---|---|
| **Aufruf**<br>AH = 09h<br>DS:DX = Adr. String | **Ergebnis**<br>String wird ab Cursorposition ausgegeben |

Der durch DS:DX adressierte String muß mit dem Dollarzeichen (24h) abgeschlossen sein. Das Endezeichen wird nicht mit ausgegeben. Als Nebenbedingung ergibt sich dadurch auch, daß der String kein Dollarzeichen enthalten darf. Als Alternative kann die allgemeinere DOS-Funktion 40h (Datei schreiben) verwendet werden.

**Funktion für Standarddrucker (LPT1, PRN)**

Als Standarddrucker ist der erste über Paralleladapter angeschlossene Drucker zugeordnet. Mit dem DOS-Kommando MODE kann die Ausgabe auf einen seriellen Drucker (COM1, COM2) gelenkt werden.

| **Zeichen auf Standarddrucker schreiben** | **Int 21-05** |
|---|---|
| **Aufruf**<br>AH = 05h<br>DL = Zeichen | **Ergebnis**<br>Zeichen auf Standard-drucker (akt. Position) |

Das in DL gspeicherte Zeichen wird ausgegeben. Eine Unterbrechung mit Ctrl-C wird berücksichtigt.

**Funktionen für Hilfsport (AUX, COM1)**

Der Standardhilfseingang ist dem Port COM1, der identisch ist mit dem Port AUX, zugeordnet. Mit dem DOS-Kommando MODE ist eine Umlenkung auf COM2 möglich. Die Funktionen arbeiten nicht interruptgesteuert, verwenden keinen Puffer und liefern keine Angaben über den Status der angeschlossenen Leitung, der im Fehlerfall wichtig sein kann. In solchen Fällen ist die BIOS-Routine 14h zu verwenden. Die von DOS eingestellten Leitungsparameter sind: 2400 Baud, keine Parität, 1 Stopbit, 8 Datenbit. Sollen andere Werte eingestellt werden, so sind die BIOS-Routinen oder das MODE-Kommando zu verwenden.

| **Zeichen von Hilfsport lesen** | **Int 21-03** |
|---|---|
| **Aufruf**<br>AH = 03h | **Ergebnis**<br>AL = Zeichen |

Die Routine wartet, bis ein Zeichen bereit ist. Eine Unterbrechung mit Ctrl-C ist möglich.

| **Zeichen nach Hilfsport schreiben** | **Int 21-04** |
|---|---|
| **Aufruf**<br>AH = 04h<br>DL = Zeichen | **Ergebnis**<br>- |

Das in DL gespeicherte Zeichen wird ohne weitere Rückmeldung ausgegeben. Eine Unterbrechung mit Ctrl-C ist möglich.

### 4.4.3. Speicherverwaltung (memory management)

Eine der wichtigsten Ressourcen des PC ist sein Speicher. Jedes Programm benötigt zum Ablauf eine bestimmte Speichergröβe. Die einfachste Speicherverwaltung besteht darin, dem Programm immer den gesamten Speicher zur Verfügung zu stellen. Diese Zuteilungsstrategie wurde bei einfachen Betriebssystemen angewendet. Die Programme im COM-Format verwenden auch heute noch dieses Verfahren. Moderne Betriebssysteme enthalten jedoch auch die Möglichkeit, Teile des Speichers zu belegen und wieder freizugeben. Diese Aufgabe des Betriebssystems heiβt Speicherverwaltung und wird im DOS durch die in Tafel 4.20 genannten Funktionen realisiert. Besonders wichtig ist die Speicherverwaltung bei der Verwaltung von Prozessen. Durch sorgfältige Planung der Reihenfolge beim Laden von speicherresidenten Programmen kann eine Zerstückelung des Speichers vermieden werden. Man sollte jeweils beim Anfordern nur den Speicherplatz angeben, der auch tatsächlich benötigt wird.

**Tafel 4.20. Funktionen zur Speicherverwaltung**

| Funktionen zur Speicherverwaltung |
|---|
| 48h Speicher anfordern (allocate memory)<br>49h Speicher freigeben (free allocated memory)<br>4Ah Speicheranforderung ändern (set block)<br>58h Zuteilungsstrategie setzen/lesen |

Im MS-DOS/PC-DOS steht am Beginn eines zugewiesenen Speicherbereichs ein Speichersteuerblock (memory control block), der die Bereichsgröβe, den Namen des Prozesses, dem der Speicher zugeordnet ist, und eine Adresse zum nächsten freien Speicherbereich enthält. Nicht zugewiesene Bereiche werden als frei betrachtet. Alle Speicherbereiche werden in einer Bereichsliste miteinander verkettet. Wenn Speicher mit einer bestimmten Größe angefordert wird, so sucht DOS in der Bereichsliste nach einem geeigneten Speicherbereich und teilt ihn dem anfordernden Programm zu. Bei der Freigabe wird der entsprechende Bereich wieder als frei gekennzeichnet. Speicher wird immer in der geforderten Länge zugewiesen. Ist ein geeigneter Bereich zu lang, so wird der Bereich zerteilt, indem am Ende des belegten Bereiches ein neuer Speichersteuerblock erzeugt und in die Liste eingefügt wird. Wenn Programme starten, so sollte eine der ersten Schritte sein, den Speicher auf das erforderliche Maβ zu reduzieren. Wenn ein Prozeβ beendet wird, gibt DOS den für den Prozeβ angeforderten

Speicherbereich zurück. Versucht ein Programm, Speicher zurückzugeben, der ihm nicht zugeteilt worden ist, so wird ein Fehler angezeigt (vgl. DOS-Fehlerliste im Anhang).

| **Speicher anfordern (allocate memory)** | **Int 21-48** |
|---|---|
| **Aufruf**<br>AH = 48h<br>BX = Speicherbereich in Paragraphen | **Ergebnis**<br>CF = 0 kein Fehler<br>AX = Segmentadresse<br>CF = 1 Fehler<br>AX = Fehlernummer<br>BX = max. verfügbare Größe |

DOS versucht, Speicher in der geforderten Größe zu belegen. Wenn die Anforderung erfüllt werden kann, ist CF = 0, und in AX steht die Segmentadresse. Bei Fehlern ist CF = 1, und in AX wird eine Fehlernummer geliefert. Als Zusatzinformation wird noch die maximal verfügbare Segmentgröße in BX mitgeteilt.

| **Speicher freigeben (free allocated memory)** | **Int 21-49** |
|---|---|
| **Aufruf**<br>AH = 49h<br>ES = Segmentadr. Speicher | **Ergebnis**<br>CF = 0 kein Fehler<br>CF = 1 Fehler<br>AX = Fehlernummer |

Der mit ES adressierte Speicherbereich wird als frei gekennzeichnet. Ist nach Ausführung der Funktion CF = 1, so ist ein Fehler aufgetreten, dessen Fehlernummer in AX steht.

| **Speicheranforderung verändern (set block)** | **Int 21-4A** |
|---|---|
| **Aufruf**<br>AH = 4Ah<br>ES = Segmentadr. Speicher<br>BX = neue Größe (Paragraph) | **Ergebnis**<br>CF = 0 kein Fehler<br>CF = 1 Fehler<br>AX = Fehlernummer<br>BX = max. verfügbare Größe |

Mit dieser Funktion kann der belegte Speicher vergrößert oder verkleinert werden. Bei Fehlern wird CF = 1 gesetzt und in AX eine Fehlernummer geliefert. Die maximal verfügbare Speichergröße steht dann in BX. Diese Funktion wird z.B. gebraucht, wenn COM-Programme, die den ganzen freien Speicher belegen, Speicherplatz zurückgeben, um zusätzliche Speicheranforderungen zuzulassen.

| Zuteilungsstrategie setzen/lesen | Int 21-58 |
|---|---|
| **Aufruf**<br>AH = 58h<br>AL = 00h Strategie lesen<br>AL = 01h Strategie setzen<br>BX = Strategienummer | **Ergebnis**<br>CF = 0 kein Fehler<br>AX = Strategienummer<br>(0 first fit, 1 best fit,<br>2 last fit) |

Mit dieser Routine kann die Strategie ermittelt oder gesetzt werden, mit der DOS seine Speicherzuteilung realisiert. Bei der Strategie 0 (first fit) wird der erste geeignete Speicherblock zugeteilt; bei der Strategie 1 (best fit) wird der kleinste geeignete Block zugewiesen, und bei Strategie 2 (last fit) wird der erste geeignete Block vom Listenende verwendet. Die Zuteilungsstrategie hat einen Einfluß auf die Zerstückelung des Gesamtspeichers. Wenn Programme speicherresident beendet werden, so ist es besonders wichtig, auf die Speicherverteilung zu achten, weil DOS keine Verschiebungen realisiert.

### 4.4.4. Prozeßverwaltung (process management)

DOS ist kein Multi-Tasking-Betriebssystem; trotzdem ist eine relativ gut ausgebaute Prozeßverwaltung vorhanden. DOS nutzt selbst diverse Funktionen zum Laden, Ausführen und Beenden von Programmen. Diese Funktionen können auch von Anwendungsprogrammen genutzt werden.

**Tafel 4.21. Funktionen zur Prozeßverwaltung**

| Prozeßverwaltung (process management) | |
|---|---|
| 4B00h | Programm laden und ausführen (load and execute) |
| 4B03h | Programm laden (load overlay) |
| 4Ch | Prozeß beenden (end process) |
| 31h | Prozeß speicherresident beenden (keep process) |
| 4Dh | Returncode von Kindprozeß lesen (get return code of child process) |
| 62h | Programmsegmentpräfix lesen (get PSP) |

Bevor ein Programm zur Ausführung kommen kann, muß die Programmdatei vom externen Speicher geladen werden. Jedes Programm kann einen Kindprozeß erzeugen und ihm die Steuerung übergeben. Der erzeugende Prozeß hat weitgehende Kontrolle über die Prozeßumgebung (environment string, geöffnete Kanäle, Returncode) des

Kindprozesses. Vor jedem Programmodul wird im Speicher ein **Programmsegmentpräfix PSP** (program segment prefix) angelegt, das alle wichtigen Daten zum Prozeß enthält. Der Aufbau des PSP wird in Tafel 4.22 gezeigt. Die Informationen sind etwas heterogen, weil sich auch im PSP noch die Versionen des DOS mit den unterschiedlichen Strategien auswirken.

**Tafel 4.22. Aufbau des PSP (program segment prefix)**

| Offset | Bytes | Beschreibung |
|---|---|---|
| 00-01h | 2 | INT 20h |
| 02-03h | 2 | Segmentadresse nach Programmende |
| 04h | 1 | reserviert |
| 05-09h | 5 | DOS-Aufruf (far call) zur INT-21h-Routine |
| 0A-0Dh | 4 | Adresse (IP,CS) Enderoutine INT 22h |
| 0E-11h | 4 | Adresse (IP,CS) Ctrl-C-Routine INT 23h |
| 12-15h | 4 | Adresse (IP,CS) Fehlerroutine INT 24h |
| 16-2Bh | | reserviert |
| 2C-2Dh | 2 | Segmentadresse der Umgebung (environment) |
| 2E-4Fh | | reserviert |
| 50-52h | 3 | DOS-Aufruf (INT 21h, FAR RET) |
| 53-5Bh | | reserviert |
| 5C-6Bh | 36 | FCB1 (file control block) |
| 6C-7Fh | 36 | FCB2 |
| 80h | 1 | Länge Kommandostring |
| 81-FFh | 127 | Kommandostring |

Der PSP hat eine Länge von 256 (100h) Byte. Jedes Programm kann auf die Informationen im eigenen PSP zugreifen. Im PSP steht auf 2Ch die Segmentadresse des **Umgebungsstrings (environment)**. In diesem maximal 32 KByte langen Feld kann das rufende Programm Informationen übermitteln. Alle Einträge sind als ASCIIZ-Strings aufgebaut, d.h. sind mit einer Null (00h) abgeschlossen. Der Abschluß wird durch einen Leerstring (nur 00h) gebildet. Zur Kennzeichnung der einzelnen Einträge wird ein Name verwendet. Alle Einträge haben dadurch die gleiche Form (NAME=Parameter). Beim Laden des Kommandointerpreters COMMAND wird vom System ein Umgebungsstring angelegt, der mit DOS-Kommandos verändert werden kann (SET, PROMPT, PATH). Der Umgebungsstring ist eine Kopie des Umgebungsstrings vom rufenden Programm. Ein weiterer Parameter ist der ab PSP-Adresse 80h gespeicherte **Kommandostring**. Hier werden bei Aufruf durch COMMAND alle Zeichen der Kommandozeile abgelegt (eine Ausnahme bilden hierbei jedoch Anweisungen zur Umlenkung der Standardeingabe und Standardausgabe mit >name und <name). Im ersten Byte ist die Anzahl Zeichen im Kommandostring eingetragen.

| Programm laden und ausführen (EXEC) Int 21-4B00 | |
|---|---|
| **Aufruf**<br>AH = 4Bh<br>AL = 00h<br>DS:DX = Dateiname (ASCIIZ)<br>ES:BX = Parameterblock | **Ergebnis**<br>CF = 0 kein Fehler<br>CF = 1 Fehler<br>AX = Fehlernummer |

Wenn ein Programm (COMMAND oder Anwendungsprogramm) diese Funktion aufruft, wird der ASCIIZ-String verwendet, um das zu ladende Programm zu lokalisieren. Der String enthält die vollständige Dateibezeichnung (Gerät, Pfad, Dateiname). Von DOS wird der entsprechende Speicherplatz angefordert und als Programmsegment eingerichtet. Am Anfang wird das PSP mit einer Länge von 256 (100h) Byte angelegt. Die zum Aufrufzeitpunkt aktiven Adreßinterrupts (22h, 23h, 24h) werden im PSP abgespeichert. Die Routine ist Bestandteil des residenten Teils von COMMAND.

Beim Laden wird die Programmdatei ab Offset 100h in das Programmsegment eingetragen. Der Ladevorgang ist vom Format der Programmdatei abhängig (vgl. Abschn. 5.6.). Beim **COM-Format** wird der gesamte verfügbare Speicher angefordert und am Segmentende ein Stack mit einer Länge von 256 (100h) Bytes eingerichtet. Das erste Wort auf dem Stack wird auf 0000h gesetzt. Soll das Programm evtl. ein weiteres Programm aufrufen, so muß ein Teil des Speichers freigegeben werden (Funktion 4Ah). Es ist dabei zu beachten, daß dann vorher der Stack neu angelegt werden muß. Die Register haben nach dem Laden folgende Werte:

| | |
|---|---|
| CS = Adresse PSP | IP = 0100h |
| DS = Adresse PSP | ES = Adresse PSP |
| SS = Adresse PSP | SP = FFFEh. |

Beim **EXE-Format** ist durch den Linker ein Vorspann angelegt, in dem alle wichtigen Angaben zum Laden des Programms enthalten sind (vgl. Abschn. 5.6.). Es werden nur DS und ES auf das PSP eingestellt. Die Werte für CS, IP, SS, und SP werden aus dem Vorspann übernommem und korrigiert mit der Adresse des Programmsegments.

Der mit ES:BX adressierte **Parameterblock** enthält Adressen, die den Umgebungsstring, die Kommandozeile und die Dateisteuerblöcke FCB1 und FCB2 verfügbar machen. Der Parameterblock hat den in Tafel 4.23 gezeigten Aufbau.

**Tafel 4.23. Programmparameterblock**

| Offset | Länge | Bedeutung Programmparameterblock |
|---|---|---|
| 00-01h | 2 | Adresse des Umgebungsstrings |
| 02-05h | 4 | Segmentadresse der Kommandozeile |
| 06-09h | 4 | Segmentadresse FCB1 |
| 0A-0Dh | 4 | Segmentadresse FCB2 |

Die Adressenangabe des Umgebungsstrings bezieht sich auf den aktuellen PSP. Ist hier der Wert 000h angegeben, so wird der Umgebungsstring des rufenden Programms verwendet. Die Adressen für FCB1 und FCB2 sind nur von Bedeutung, wenn die traditionellen DOS-Funktionen zur Dateiverwaltung verwendet werden.

Alle geöffneten Dateien werden dem Kindprozeß zur Verfügung gestellt. Das ist z.B. besonders für die Standarddateien (stdin, stdout, stderr, stdaux, stdprn) wichtig, die immer durch DOS automatisch eröffnet werden. Auf diese Weise lassen sich sehr leicht Filterprogramme realisieren (vgl. Abschn. 6.4.).

| **Programm laden (load overlay)** | **Int 21-4B03** |
|---|---|
| **Aufruf**<br>AH = 4Bh<br>AL = 03h<br>DS:DX = Dateiname (ASCIIZ)<br>ES:BX = Parameterblock | **Ergebnis**<br>CF = 0 kein Fehler<br>CF = 1 Fehler<br>AX = Fehlernummer |

Diese Funktion ist als eingeschränkte Variante der EXEC-Funktion zu verstehen. Es wird vorausgesetzt, daß der im Parameterblock angegebene Speicherbereich bereits dem rufenden Programm zugewiesen ist und eine ausreichende Größe hat. Die mit DS:DX adressierte Programmdatei wird nur geladen. Ein PSP wird nicht angelegt, und es erfolgt auch kein automatischer Start. Die Funktion wird hauptsächlich zum Laden von Overlayprogrammen genutzt. Der Parameterblock hat dabei folgenden Aufbau:

| Offset | Länge | Bedeutung Programmparameterblock |
|---|---|---|
| 00-01h | 2 | Ladeadresse des Programms |
| 02-03h | 2 | Verschiebungswert für Lademodul |

Der Wert für die Ladeadresse ist auf den PSP des rufenden Programms bezogen.

| **Prozeß beenden (EXIT)** | **Int 21-4C** |
|---|---|
| **Aufruf**<br>AH = 4Ch<br>AL = Returncode | **Ergebnis**<br>- |

Mit dieser Funktion wird der laufende Prozeß beendet und dem rufenden Prozeß ein Returncode übermittelt, der dort durch die Funktion WAIT (4Dh) festgestellt werden kann. Ist das rufende Programm der Kommandointerpreter mit einer Stapeldatei, so kann der Returncode durch das Unterkommando IF abgefragt werden. Durch Auswertung dieses Fehlercodes ist die Reaktion auf bestimmte Abbruchbedingungen möglich. Die Bedeutung der Returncodes ist bei Funktion 4Dh erläutert. Bei Programmbeendigung werden alle geöffneten Dateien geschlossen.

Diese Funktion ersetzt die traditionelle Funktion 00h und den Interrupt 20h. Die Anwendung der neuen Funktion 4Ch ist zu empfehlen, weil sie sich in das neue Dateisystem einfügt.

| **Prozeß speicherresident beenden** | **Int 21-31** |
|---|---|
| **Aufruf**<br>AH = 31h<br>AL = Returncode<br>DX = Speicher (Paragraphen) | **Ergebnis**<br>- |

Die Funktion erfüllt den gleichen Zweck wie die vorhergehende. Der Unterschied besteht darin, daß der Programmtext bzw. ein Teil davon speicherresident bleiben kann. Die Größe des residenten Teils (Angabe in Paragraphen einschließlich) wird in Register DX angegeben. Mit dem Interrupt 27h kann das gleiche Ergebnis erzielt werden, wobei dort maximal 64 KByte belegt werden können. Mit dieser Funktion werden die TSR-Programme realisiert, die als Hilfsroutinen z.B. auf einen bestimmten Tastendruck reagieren. Ein Beispiel dafür ist das Programm Sidekick (Borland), das eine ganze Reihe von Zusatzfunktionen bereitstellt. Bei der Arbeit mit residenten Programmen ist auf die Aufrufreihenfolge zu achten, damit der verfügbare Speicher nicht zerstückelt wird.

| **Returncode von Kindprozeß lesen (WAIT)** | **Int 21-4D** |
|---|---|
| **Aufruf**<br>AH = 4Dh | **Ergebnis**<br>AX = Returncode |

Mit dieser Funktion wird der Returncode eines Kindprozesses bereitgestellt, der bei Beendigung mit Funktion 4Ch oder 31h übergeben wird. Der Wert, der nur einmal gelesen werden kann, wird in AX bereitgestellt. In AH steht ein Wert, der die Beendigungsart angibt (0 normale Beendigung, 1 Abbruch durch Ctrl-C, 2 kritischer Fehler, 3 Beendigung durch Funktion 31h). In AL steht der Returncode des Kindprozesses.

| **Programmsegmentpräfix lesen (get PSP)** | **Int 21-62** |
|---|---|
| **Aufruf**<br>AH = 62h | **Ergebnis**<br>BX = Segmentadresse des aktiven Prozesses (PSP) |

Die in Register BX gelieferte Segmentadresse zeigt auf den PSP des laufenden Prozesses. Der Zugriff kann von Bedeutung sein, wenn zeitgesteuert mehrere Prozesse verwaltet werden sollen.

### 4.4.5. Netzwerkverwaltung

Durch DOS wird ein einfaches Netzwerk unterstützt, das aus einem Server und einer oder mehreren Arbeitsstationen besteht. Dabei wird in den Arbeitsstationen eine Zuweisungsliste verwaltet, in der eine Umlenkung von Geräten und Dateien zum Server eingetragen ist. Bei Anwendung dieser Funktionen lassen sich z.B. große Festplatten oder Drucker zentral aufstellen und dezentral nutzen. Es gibt für DOS verschiedene Realisierungen von Netzwerken. Hier sollen die DOS-Funktionen behandelt werden, die durch MS-DOS bereitgestellt werden. Eine wesentliche Erweiterung der Netzwerkfunktionen erfolgt im OS/2 als Weiterentwicklung von MS-DOS. Wesentliche Erweiterungen dieser Funktionen werden mit größeren Programmpaketen, wie z.B. NETBIOS, angeboten. Einige Informationen dazu sind im Abschn. 10.4. zu finden. Eine umfassende Behandlung ist in diesem Buch nicht möglich.

**Tafel 4.24. Funktionen zur Netzwerkverwaltung**

| **Netzwerkfunktionen** | |
|---|---|
| 4409h | IOCTL - Laufwerk lokal oder im Netzwerk ? |
| 440Ah | IOCTL - Kanal lokal oder im Netzwerk ? |
| 5Ch | Zugriffsschutz setzen/freigeben |
| 5Eh | Netzwerkfunktionen |
| 5Fh | Verwaltung der Netzzuweisungsliste |

Die Funktionsgruppe 44h gilt allgemein für die Verwaltung der Gerätetreiber. Das Netzwerkkonzept ordnet sich in diese Technologie ein.

| **Zugriffsschutz setzen/freigeben** | **Int 21-5C** |
|---|---|
| **Aufruf**<br>AH = 5Ch<br>AL = 00h Schutz setzen<br>AL = 01h Schutz freigeben<br>CX:DX = Anfang Bereich<br>SI:DI = Länge Bereich | **Ergebnis**<br>CF = 0 kein Fehler<br>CF = 1 Fehler<br>AX = Fehlernummer |

Der Zugriffsschutz ist erforderlich, wenn im Netzwerk mehrere Programme auf eine Datei zugreifen können. Mit dieser Funktion kann man einen Dateibereich für den Zugriff durch ein anderes Programm sperren. Die Bereichsadresse wird als Offset zum Dateianfang angegeben. Bemerkt DOS beim Zugriff, daß ein Dateibereich gesperrt ist, so wird bis zu dreimal wiederholt, bevor eine Fehlermeldung (Int 24h) ausgegeben wird. Die Anzahl Zugriffsversuche kann durch die IOCTL-Funktion (44h) verändert werden.

| **Spezielle Netzwerkfunktionen** | **Int 21-5E** |
|---|---|
| **Aufruf**<br>AH = 5Eh<br>AL = 00h Netzwerkname lesen<br>DS:DX = Pufferadr.(16 Byte)<br>AL = 02h Druckerstring<br>BX = Zuweisungsindex<br>CS:DI = Stringadresse<br>CX = Stringlänge | **Ergebnis**<br>CF = 0 kein Fehler<br>CF = 1 Fehler<br>AX = Fehlernummer |

Diese Funktion dient dazu, Informationen über eine Arbeitsstation zu ermitteln oder eine Kopfzeile an den Druckerserver zu senden.

| **Verwaltung der Netzzuweisungsliste** | **Int 21-5F** |
|---|---|
| **Aufruf**<br>AH = 5Fh<br>AL = 02h Eintrag lesen<br>AL = 03h Eintrag setzen<br>AL = 04h Eintrag löschen | **Ergebnis**<br>CF = 0 kein Fehler<br>CF = 1 Fehler<br>AX = Fehlernummer |

Mit dieser Funktion lassen sich die Einträge in der Netzzuweisungsliste manipulieren. Das Ende der Liste wird beim Lesen durch Fehlernummer = 18 angezeigt. Jeder Eintrag bedeutet die Umleitung eines Gerätes oder einer Datei zum Server. Weitere Angaben sind dem Netzwerkhandbuch zu entnehmen.

### 4.4.6. Systemverwaltung

In diesem Abschnitt werden allgemeine Systemfunktionen zusammengefaßt, die sich den anderen Gruppen nicht zuordnen lassen. Hierzu zählen folgende Funktionen:

- Datum und Uhrzeit lesen/setzen
- Interruptvektoren lesen/setzen
- allgemeine Funktionen (Version lesen, Status Ctrl-C, Länderinformation, Fehlerroutine).

| **Datum und Uhrzeit** | |
|---|---|
| **Aufruf**<br>AH = 2Ah **Datum lesen** | **Ergebnis**<br>AL = Wochentag (0 = So)<br>CX = Jahr (1980..2099)<br>DH = Monat (1...12)<br>DL = Tag (1...31) |
| AH = 2Bh **Datum setzen**<br>CX = Jahr (1980...2099)<br>DH = Monat (1...12)<br>DL = Tag (1...31) | AL = 00h Datum gültig<br>Al = FFh Datum ungültig |
| AH = 2Ch **Zeit lesen** | CH = Stunde (0...23)<br>CL = Minute (0...59)<br>DH = Sekunde (0...59)<br>DL = Hundertstel (0...99) |
| AH = 2Dh **Zeit setzen**<br>CH = Stunde (0...23)<br>CL = Minute (0...59)<br>DH = Sekunde (0...59)<br>DL = Hundertstel (0..99) | AL = 00h Zeit gültig<br>AL = FFh Zeit ungültig |

Datum und Uhrzeit werden vom DOS verwendet, um für Dateien einen Änderungsstand zu verwalten.

| Interruptvektoren | |
|---|---|
| **Aufruf**<br>AH = 35h **Int-Vektor lesen**<br>AL = Interruptnummer | **Ergebnis**<br>ES:BX = Interruptroutine |
| AH = 25h **Int-Vektor setzen**<br>AL = Interruptnummer<br>DS:DX = Interruptroutine | |

Man kann beim 8086/8088 die Interruptvektoren direkt verändern. Für eine konsistente Behandlung durch das DOS wurden dafür jedoch diese beiden Funktionen eingeführt.

Die starke Verbreitung von MS-DOS/PC-DOS hat zur Folge, daß die **Besonderheiten einzelner Länder** berücksichtigt werden muß. Einige Werte werden durch die Wahl der Betriebssystemversion festgelegt, andere lassen sich bei der Systemkonfigurierung einstellen. Die etwas umständlich handhabbare Funktion für die Länderspezifik liefert in einem Informationsfeld alle für die Gestaltung von Listenausdrucken erforderlichen Angaben. Hier soll auf diese Informationen nicht weiter eingegangen werden.

| Allgemeine Systemfunktionen | |
|---|---|
| **Aufruf**<br>AH = 30h **Version lesen**<br>(Version.Änderung) | **Ergebnis**<br>AL = Version (z.B. 03h)<br>AH = Änderung (z.B. 00h) |
| AH = 33h **Status Ctrl-C**<br>AL = 00h Status lesen<br>AL = 01h Status setzen<br>DL = 00h off/01h on | DL = aktueller Status<br>(00h = off, 01h = on) |
| AH = 38h **Länderspezifik**<br>DX = FFFFh Code setzen<br>AL = Ländercode<br>DS:DX = Puffer f.Code lesen<br>AL = Ländercode | CF = 0 kein Fehler<br>AX = Fehlernummer<br><br>DS:DX = Länderinformation<br>BX = Ländercode |
| AH = 59h **Fehlerroutine**<br>Aufruf nach Funktionen<br>(CF = 1) | AX = Fehlernummer<br>BH = Fehlerklasse<br>BL = Maßnahmenvorschlag<br>CH = Fehlerort |

Die allgemeine Fehlerroutine (59h) kann nach dem fehlerhaften Aufruf (CF = 1) der meisten DOS-Funktionen eine genauere Fehlerbeschreibung liefern. Zusätzlich zu der Fehlernummer in AX wird eine Angabe zur Fehlerklasse, zu möglichen Maßnahmen und zum Fehlerort gemacht. Die Fehlerliste ist im Anhang zu finden.

# 5. Dateisystem

Eine der wichtigsten Aufgaben von DOS ist die Verwaltung von Dateien und Programmen. In diesem Abschnitt sind alle für die Programmierung wichtigen Informationen zusammengefaßt, die mit der Datenverwaltung und dem dazu erforderlichen Dateisystem zusammenhängen. Im Abschn. 5.1. werden die Gerätebesonderheiten, die dabei eine Rolle spielen, die davon abgeleiteten Dateitypen und das Konzept der Gerätetreiber (driver) behandelt. Im Abschn. 5.2. wenden wir uns der Dateistruktur zu, deren Verständnis für eine effektive Datenbehandlung unerläßlich ist. In den Abschnitten 5.3. und 5.4. werden die von DOS bereitgestellten Funktionen übersichtsartig behandelt und spezielle Möglichkeiten zur Vereinfachung der Dateiarbeit dargestellt. Eine genaue Beschreibung der einzelnen Funktionen ist im Abschn. 5.5. zu finden. Die Behandlung der einzelnen Möglichkeiten erfolgt hier nicht alphabetisch, sondern in der Zuordnung zu Geräten, Inhaltsverzeichnis und Datei. Im Anhang ist eine alphabetische Übersicht der Kommandos und Funktionen zu finden. Abschn. 5.6. geht auf die verschiedenen Programmformate näher ein, und im Abschn. 5.7. werden einige Probleme der Datensicherung behandelt.

## 5.1. Logische Geräte

### 5.1.1. Gerätebesonderheiten, Dateitypen

Vorraussetzung für die Arbeit mit Daten sind entsprechende Geräte, die Informationen speichern oder übertragen können. Nach der Art der Behandlung werden im DOS **zeichenorientierte Geräte** und **blockorientierte Geräte** unterschieden. Für zeichenorientierte Geräte gilt die allgemeine Einschränkung, daß sie nur für sequentiell organisierte Dateien geeignet sind. Bei diesen Geräten ist die Zeichenbehandlung zumeist mit der Darstellung oder Übertragung verbunden. Tafel 5.1 gibt eine Übersicht über die allgemein verfügbaren Geräte. Will man neuentwickelte Geräte in die DOS-Struktur einordnen, so muß eine Zuordnung zu einer dieser Gruppen erfolgen.

Im Abschn. 2. ist auf die physikalischen Besonderheiten der einzelnen Gerätetypen kurz eingegangen worden. Aus der Sicht des Betriebssystems bezeichnet man das Gerät oft auch als physisches Gerät, um einen Unterschied zum logischen Gerät hervorzuheben. Das logische Gerät gliedert sich voll in die Dateiverwaltung ein. Es sind für die Geräte eindeutige Bezeichner festgelegt, die im DOS durchgängig verwendet werden. Diese Namen sind reserviert und

werden nur in der festgelegten Bedeutung akzeptiert. Die Datenverwaltung (Dateisystem) arbeitet nur mit logischen Geräten. Will man besondere Eigenschaften von physischen Geräten ansprechen, so sind die in Abschn. 4.3. beschriebenen BIOS-Routinen zu nutzen bzw. direkt die entsprechenden Bausteine über Ports und Adressen zu programmieren.

**Tafel 5.1. Gerätetypen (Standardausstattung)**

| Blockorientierte Geräte | Zeichenorientierte Geräte |
|---|---|
| Diskette (A:, B:, ...)<br>Festplatte (C:, D:, ...) | Bildschirm (CON)<br>Tastatur (CON)<br>Drucker (PRN, LPT1)<br>serielle Schnittstelle(COM1) |

Die Schnittstelle zwischen dem Betriebssystem und den Geräten sind **Gerätetreiber**. Das sind spezielle Programme, die alle Gerätebesonderheiten kennen und zum Betriebssystem eine festgelegte Schnittstelle einhalten. Die Konfigurierung des Betriebssystems gestattet die Einbindung eigener Gerätetreiber (vgl. Abschnitt 4.2.).

Die Programmierung eigener Gerätetreiber ist eine etwas aufwendige Angelegenheit. Sie bietet jedoch die Möglichkeit der vollwertigen Eingliederung spezieller Geräte in die Philosophie des Betriebssystems und damit eine weitgehend geräteunabhängige Programmierung. Für die Standardgeräte werden alle Gerätetreiber mit dem Betriebssystem ausgeliefert. Zur Einbindung spezieller blockorientierter Geräte wird ab Version 3.00 ein Treiber DRIVER.SYS ausgeliefert, bei dem einige Parameter eingestellt werden können. Ab Version 3.30 werden auch für Tastatur und Bildschirm konfigurierbare Treiberprogramme angeboten. Zur Erleichterung der Programmierung ist ein Treiberprogrammbeispiel für einen RAM-Disk im Quelltext (Assembler) auf der Systemdiskette enthalten. Das Programm erläutert die Schnittstellen zum Betriebssystem. Die wichtigsten Steuerblöcke und Funktionen für die Treiberprogrammierung sollen hier aufgeführt und kurz erläutert werden.

### 5.1.2. Gerätetreiber

Alle Standardtreiber und die beim Systemladen installierten Treiber sind in einer Liste miteinander verkettet. Jeder Treiber in der Liste hat zwei Einsprungspunkte, den Strategie- und den Interrupteinsprungspunkt. Zusammen mit einem Attributfeld und

einem Namenfeld bilden diese Angaben den Treibervorspann mit dem in Tafel 5.2 gezeigten Aufbau.

**Tafel 5.2. Treibervorspann**

| **Treibervorspann (device header)** |
|---|
| **DWORD** Zeiger auf nächsten Treiber<br>(beim letzten Treiber -1) |
| **WORD** Attribute<br>Bit15 = 1 zeichenorientiertes Gerät<br>= 0 blockorientiertes Gerät<br>Bit14 = 1 IOCTL wird unterstützt<br>Bit13 = 1 kein IBM-Format (Blockgerät)<br>Bit11 = 1 Open/Close/RM wird unterstützt<br>Bit3 = 1 aktuelles Clock-Gerät<br>Bit2 = 1 aktuelles NUL-Gerät<br>Bit1 = 1 aktuelle Standardausgabe (stdout)<br>Bit0 = 1 aktuelle Standardeingabe (stdin)<br>**WORD** Strategieeinsprungspunkt (strategy entry point)<br>**WORD** Interrupteinsprungspunkt (interrupt entry point<br>**8Byte** Gerätename für Zeichengeräte |

Die Unterteilung des Treibermoduls in Strategie- und Interruptroutine geht von der Überlegung aus, daß in einem modernen Betriebssystem Gerätefunktionen weitgehend asynchron bearbeitet werden. MS-DOS ist bisher immer ein Single-Task-System geblieben; die Gestaltung der Gerätetreiber läßt jedoch einen durch Interrupts gesteuerten Betrieb zu. Wenn eine Gerätefunktion aufgerufen wird, so wird ein **Anforderungsblock (request header)** gebildet und die Strategieroutine aufgerufen. Der Anforderungsblock hat den in Tafel 5.3 gezeigten Aufbau und wird in eine modulinterne Warteschlange eingereiht, die durch die Interruptroutine bearbeitet wird. Da die Interruptroutine nicht als eigene Task läuft, wird sofort nach Verlassen der Strategieroutine die Interruptroutine aufgerufen und die Anforderung bearbeitet. Der Treiber legt im Anforderungsblock eine Statusinformation ab und gibt die Steuerung an DOS zurück.

**Tafel 5.3. Anforderungsblock (driver)**

| **Anforderungsblock (request header)** |
|---|
| **Byte** Länge des Anforderungsblocks<br>**Byte** Einheit (Nummer innerhalb eines Treibers)<br>**WORD** Gerätefunktion<br>**WORD** Gerätestatus (Bit8 = fertig, Bit9 = belegt,<br>Bit15 = Fehler, Bit0...7 = Fehler-Nr.)<br>**8Byte** reserviert |

Die Verbindung zwischen Gerätetreiber und Programm wird durch DOS vermittelt. Teilweise wird eine Anforderung durch die DOS-Funktion 44h (IOCTL - Ein-/Ausgabe-Kontrolle für Kanäle) ausgelöst. Die anderen Funktionen werden bei der Systemkonfigurierung (Auswertung der Datei CONFIG.SYS mit dem Parameter DEVICE=...) aufgerufen oder aus DOS-Funktionen zur Dateiverwaltung abgeleitet.

Der direkte Aufruf der Treiberroutinen ist nicht vorgesehen. Bei der Erstellung eines eigenen Treibers sollte man sich an ein Beispiel halten (Datei VDISK.LST auf der Systemdiskette oder Mustertreiber im Technischen Handbuch zum MS-DOS). Hier werden lediglich die definierten Gerätefunktionen genannt (Tafel 5.4).

**Tafel 5.4. Gerätefunktionen im Treiber**

| Gerätefunktionen im Treiber | | | |
|---|---|---|---|
| 0 | INIT | 10 | OUTPUT STATUS |
| 1 | MEDIA CHECK | 11 | OUTPUT FLUSH |
| 2 | BUILD BPB | 12 | IOCTL OUTPUT |
| 3 | IOCTL INPUT | 13 | DEVICE OPEN |
| 4 | INPUT | 14 | DEVICE CLOSE |
| 5 | NON-DESTRUCTIVE INPUT | 15 | REMOVABLE MEDIA |
| 6 | INPUT STATUS | 16 | OUTPUT UNTIL BUSY |
| 7 | INPUT FLUSH | 19 | IOCTL REQUEST |
| 8 | OUTPUT | 23 | GET DRIVE MAP |
| 9 | OUTPUT VERIFY | 24 | SET DRIVE MAP |

Eine wichtige Informationsquelle für Geräteparameter im DOS ist der durch den Treiber bereitgestellte **BIOS-Parameterblock**, der entweder aus festen Werten im Treiber oder bei Aktivierung eines Blockgeräts eingelesen wird. Die Geräteinformationen werden dort durch Formatierungsroutinen eingetragen. Es ist dabei zu beachten, daß für Festplatten eine Aufteilung des gesamten Speicherplatzes in bis zu vier logische Plattengeräte (partition) möglich ist. Für die Einteilung wird ein spezielles Dienstprogramm für Festplatten FDISK benutzt (vgl. Abschnitte 2.6. und 6.). Voraussetzung für eine Aufteilung ist die physische Strukturierung der Festplatte in Sektoren unter Verwendung der BIOS-Funktion Formatieren (Int 13h, Funktion 05). Obwohl man für die normale Systemnutzung die genaue logische Struktur der Diskette oder Festplatte nicht kennen muß, ist doch in einigen Fällen (z.B. bei nicht behebbaren Fehlern auf einem Datenträger) die direkte Analyse von Sektoren sinnvoll. Daher soll hier kurz auf die logische Gerätestruktur eingegangen werden.

### 5.1.3. Logische Struktur von Disketten und Festplatten

Alle unter DOS formatierten Disketten und Festplatten enthalten Sektoren mit einer Länge von 512 Bytes. Das Urladerprogramm erwartet im ersten physischen Sektor des Ladegeräts einen Urladereintrag (boot record). Dieses Stück Programm steuert den weiteren Ablauf. Bei Disketten wird das Betriebssystem bzw. ein separates Programm (stand-alone program) eingelesen. Bei Festplatten heißt dieses Programm Haupturladereintrag (master boot record), der nach einer Partition sucht, die eine aktives Betriebssystem enthält. Die Kennzeichung einer Partition als aktiv erfolgt ebenfalls mit dem Programm FDISK bzw. einem entsprechenden Programm.

Auf den Haupturladereintrag folgt eine Tabelle, die die einzelnen Bereiche beschreibt (Tafel 5.5).

**Tafel 5.5. Festplattenbereichstabelle**

| | **Bereichstabelle (partition table)** | | | | |
|---|---|---|---|---|---|
| Partition 1 Start | Boot | Kopf | Sektor | Zylinder | Boot: 80h = aktiv |
| Ende | System | Kopf | Sektor | Zylinder | System: 00h = ? 01h = DOS12 04h = DOS16 |
| | relative Sektornummer (Start) | | | | |
| | Anzahl Sektoren | | | | |
| Partition 4 Start | Boot | Kopf | Sektor | Zylinder | |
| Ende | System | Kopf | Sektor | Zylinder | |
| | relative Sektornummer (Start) | | | | |
| | Anzahl Sektoren | | | | |
| Endekennung | 55AAh | | | | |

Die Kopf- und Sektornummern können in einem Byte untergebracht werden. Die Zylindernummer kann größer als 256 werden. Es werden darum zusätzlich Bits aus dem Feld für die Sektornummer verwendet. Ist Boot = 80h, so handelt es sich um die aktive Partition, von der geladen werden soll. Die Sektoradressen beim Zugriff auf die Partition werden mit den relativen Sektornummern korrigiert. Beim Aufruf des Urladereintrags einer Partition wird die Adresse des Tabelleneintrags in DS:DI übergeben. Im Feld

"System" steht eine Kennzeichnung des verwendeten Betriebssystems.

Jede Partition hat wiederum die Struktur einer Diskette. Die logische Diskettenstruktur wird durch das DOS-Kommando FORMAT bzw. ein entsprechendes Programm erzeugt und hat die in Tafel 5.6 genannten Bestandteile.

**Tafel 5.6. Logische Diskettenstruktur**

| Logische Diskettenstruktur | |
|---|---|
| Urladereintrag (boot record) | 1 Sektor |
| FAT 1 (file allocation table) | Länge variabel |
| FAT 2 (Kopie von FAT 1) | Länge wie FAT 1 |
| Wurzelverzeichnis (root directory) | Länge variabel |
| Datenbereich (data area) | |

Der **Urladereintrag** steht immer im Sektor 1 der Diskette oder Partition und ist bei allen DOS-Formaten gleich aufgebaut. Zusätzlich zum Ladeprogramm sind Informationen enthalten, die das spezielle Datenträgerformat beschreiben und die in den BIOS-Parameterblock (BPB) des Treibers für blockorientierte Geräte übernommen werden. Der Urladereintrag hat folgenden Aufbau:

**Tafel 5.7. Urladereintrag (boot record)**

| Format des Urladereintrags (boot record) | |
|---|---|
| **3 BYTE** | Sprungbefehl zum Ladeprogramm |
| **8 BYTE** | OEM-Name und Version (z.B. IBM 3.0) |
| **WORD** | Sektorlänge in Byte |
| **BYTE** | Clustergröße (Potenz von 2) |
| **WORD** | reservierte Sektoren (ab Sektor 0) |
| **BYTE** | Anzahl FAT |
| **WORD** | Anzahl Einträge im Wurzelverzeichnis |
| **WORD** | Anzahl Sektoren auf Diskette oder Partition |
| **BYTE** | Formatkennzeichen (media descriptor) |
| **WORD** | Anzahl Sektoren in FAT 1 bzw. FAT 2 |
| **WORD** | Anzahl Sektoren auf einer Spur |
| **WORD** | Anzahl Köpfe |
| **WORD** | Anzahl spezieller Sektoren (hidden sectors) |
| | Ladeprogramm |

Das Formatkennzeichen (z.B. F8h Festplatte, FDh 360 KByte, F9h 1,2 MByte) ist nicht sehr aussagekräftig, weil es keine eindeutige Zuordnung zu allen möglichen Formaten gibt. In DOS wird es nicht weiter verwendet. Vom Gerätetreiber sollte eine eindeutige Bezeichnung im BPB gebildet werden, damit DOS den Diskettenwechsel erkennen kann.

Die **Dateizuordnungstabelle FAT** (file allocation table) wird zur Verwaltung der **Speichereinheiten (cluster)** im Datenbereich genutzt. Als Clustergröße wird im DOS eine Anzahl Sektoren (1, 2, 4, 6, ...) gewählt. Jeder Cluster hat eine Nummer. Im Inhaltsverzeichnis steht die Nummer des ersten Clusters der Datei. In der FAT steht im Eintrag mit dieser Nummer die Nummer des Folgeclusters und so fort. Auf der letzten Clusterposition einer Datei steht in der FAT eine Endekennung. Die Anzahl Bit, die für einen Eintrag gebraucht werden, ist von der Anzahl Cluster auf dem Datenträger abhängig. Für Disketten und Festplatten (der XT hatte ursprünglich eine Festplatte von 10 MByte) wurden anfangs 12 Bit für eine Clusternummer gewählt, um den Platzbedarf für die FAT klein zu halten. Maximale Clusternummer ist dabei 4086. Für größere Festplatten (z.B. 20 MByte) sind 12 Bit nicht mehr ausreichend. Im DOS werden darum zwei FAT-Versionen (Clusternummern mit 12 Bit oder 16 Bit) verwendet. Im Urladereintrag wird eine Kennzeichnung der FAT-Version eingetragen (Feld System). FAT-Einträge haben folgende Bedeutung:

| | |
|---|---|
| 0000h | Cluster ist frei |
| 0002h - FFEFh | Nummer eines Folgeclusters |
| FFF0h - FFF7h | Cluster fehlerhaft (nicht benutzt) |
| FFF8h - FFFFh | Endekennung (letzter Cluster der Datei). |

Bei der FAT-Version mit 12 Bit entfällt die erste Hexadezimalstelle. Alle Cluster einer Datei bilden durch diese Speicherungsform eine über die FAT verkettete Liste. Die Cluster einer Datei stehen dadurch manchmal sehr verstreut auf dem Datenträger. Werden die Zugriffszeiten zu lang, so muß der Datenträger durch Umkopieren reorganisiert werden. Die Umrechnung einer Clusternummer in die entsprechende logische Sektornummer erfolgt durch das DOS. Sollte für direkten Zugriff auf spezielle Sektoren (Int 25h und Int 26h) die Umrechnung im Anwendungsprogramm erforderlich sein, so ist zu beachten, das die beiden ersten Einträge der FAT für Sonderzwecke benutzt sind, d.h., der Datenbereich beginnt mit Clusternummer = 2.

Das **Wurzelverzeichnis** (root directory) folgt direkt auf die FAT-Bereiche. Die Einträge in diesem Verzeichnis beschreiben jeweils eine Datei oder ein Unterverzeichnis. Alle Einträge haben eine Länge von 32 Byte. Die Anzahl der für das Wurzelverzeichnis reservierten Sektoren ist vom Datenträger abhängig; die daraus abgeleitete Anzahl von Verzeichniseinträgen wird vom Formatie-

rungsprogramm im Urladereintrag festgehalten. Weitere Angaben zur Verzeichnisstruktur sind im Abschn. 5.2. zu finden.

Der **Datenbereich** ist in Form von Clustern organisiert, die über die FAT verwaltet werden. Hier werden die Dateien und Unterverzeichnisse gespeichert. Der Datenbereich belegt den verbleibenden Platz der Diskette oder Partition. Es sei noch einmal darauf hingewiesen, daß die Datei nicht notwendig in aufeinanderfolgenden Clustern gespeichert sein muß. Der Speicherplatz wird so belegt, wie freie Einträge in der FAT gefunden werden. Zur Verringerung des Positionieraufwands werden die Clustergrößen meist so gewählt, daß die Sektoren eines Zylinders innerhalb des gleichen Clusters liegen (z.B. bei einer Diskette die beiden Spuren auf der Ober- und Unterseite).

## 5.2. Dateistruktur

Die Dateien sind im Datenbereich eines Datenträgers gespeichert. Mit der Version 2.00 von MS-DOS/PC-DOS wurde das einfache, an das bekannte CP/M-System angelehnte Dateisystem sehr stark erweitert und verbessert. Für die Dateiverwaltung wurden Konzepte des bewährten UNIX (bzw. der Microsoft-Implementierung XENIX) übernommen, die sich in neuen Systemfunktionen und einer erweiterten Dateistruktur auf Disketten und Festplatten widerspiegeln. Wichtige Konzepte der Dateiverwaltung sind

- **hierarchische Verzeichnisstruktur**
- **Dateiverwaltung liegt vollständig im DOS**
- **Verbindung zwischen Programm und Datei über Kanäle**
- **Dateien werden als Bytestrings aufgefaßt.**

Die hierarchische Verzeichnisstruktur wird im Abschn. 5.2.1. erläutert. Im Abschn. 5.2.2. folgt eine kurze Information zur (veralteten) Dateiverwaltung der Version 1.00 des DOS. Auf die Konzeption der Dateiverwaltung ab Version 2.00 wird im Abschn. 5.3. näher eingegangen. Die verschiedenen DOS-Dienstleistungen werden durch Funktionsaufrufe bereitgestellt, die im Abschn. 5.4. beschrieben werden.

### 5.2.1. Hierarchische Verzeichnisstruktur

Im Abschn. 5.1. ist das Wurzelverzeichnis als einer der Bereiche einer Diskette oder Partition auf der Festplatte kurz erläutert worden. Dieses Wurzelverzeichnis bildet den Ausgangspunkt für den Zugriff zu einer Datei im Datenbereich.

Jeder Eintrag im Verzeichnis hat eine Länge von 32 Byte mit

dem in Tafel 5.8 gezeigten Aufbau. Alle Felder können mit DOS-Funktionen gelesen und verändert werden.

**Tafel 5.8. Format eines Verzeichniseintrags**

| Verzeichniseintrag (directory entry) | | |
|---|---|---|
| **Offset** | **Länge** | **Bedeutung** |
| 00-07h | 8 | Dateiname |
| 08-0Ah | 3 | Namenserweiterung |
| 0Bh | 1 | Attribute |
| 0C-15h | 10 | reserviert |
| 16-17h | 2 | Uhrzeit des letzten Schreibens |
| 18-19h | 2 | Datum des letzten Schreibens |
| 1A-1Bh | 2 | Nummer des ersten Clusters der Datei |
| 1C-1Fh | 4 | Dateilänge in Byte |

Der **Dateiname** ist der bis zu 8 Zeichen lange Hauptname einer Datei. Ist der Name kürzer, so wird mit Leerzeichen aufgefüllt. Der letzte Eintrag im Verzeichnis enthält im ersten Byte eine 00h. Gelöschte Einträge werden im ersten Byte mit E5h ('σ') überschrieben. Beginnt der Dateiname mit 2E2Eh, so verweist der Eintrag auf das übergeordnete Verzeichnis (parent directory), das in der DOS-Anzeige mit dem Kommando DIR durch zwei Punkte gekennzeichnet ist.

Die **Namenserweiterung** hat eine maximale Länge von 3 Byte und dient der weiteren Unterscheidung von Dateien mit gleichem Dateinamen. Der auf der Ebene von COMMAND angegebene Punkt tritt intern nicht in Erscheinung. Nicht belegte Byte sind mit Leerzeichen aufgefüllt.

Die **Attribute** gestatten eine interne Kennzeichnung einzelner Verzeichniseinträge, z.B. Datei nur lesbar, Datei ist Bestandteil des Systems oder Eintrag verweist auf Unterverzeichnis. Die Attribute sind wie folgt kodiert:

**Tafel 5.9. Dateiattribute**

| Dateiattribute | | |
|---|---|---|
| **Wert** | **Bit** | **Bedeutung** |
| 01h | 0 | Schreibschutz - nur lesen (read only) |
| 02h | 1 | versteckte Datei (hidden file) |
| 04h | 2 | Systemdatei (system file) |
| 08h | 3 | Datenträgeretikett (volume label) |
| 10h | 4 | Unterverzeichnis (subdirectory) |
| 20h | 5 | Datei verändert |

Mit Bit 0 kann eine Datei vorm Überschreiben geschützt werden. Auch gegen unbeabsichtigtes Löschen ist dieses Bit ein guter Schutz. Viele DOS-Erweiterungen setzen sich über diese Anzeige hinweg; meistens wird aber eine Bestätigung für das Überschreiben oder Löschen abgefragt.

Mit Bit 1 und Bit 2 wird die Anzeige von Dateien unterdrückt. Solche versteckten Dateien (z.B. Systemdateien IBMBIO.COM und IBMDOS.COM) können beliebig verändert werden, wenn man den Namen kennt, lassen sich aber mit DOS-Funktionen normalerweise nicht kopieren und verändern. Viele Erweiterungsprogramme zu DOS (z.B. Norton Commander) ignorieren dieses Bit und lassen die Veränderung dieses Attributs zu.

Mit Bit 3 wird ein Eintrag als Datenträgeretikett gekennzeichnet. Zu diesem Eintrag gehört keine Datei. Es ist nur ein solcher Eintrag sinnvoll. Der erste wird bei der Anzeige des Verzeichnisses verwendet.

Bit 4 kennzeichnet einen Eintrag als Verweis auf ein Unterverzeichnis. Die mit der Clusternummer adressierte Datei erzeugt eine weitere Ebene im hierarchischen Dateisystem.

Bit 5 wird durch Archivierungsprogramme (z.B. BACKUP) abgefragt, um veränderte Dateien zu sichern. Bei jeder Änderung einer Datei wird dieses Bit gesetzt. Bei der Archivierung wird dieses Bit zurückgesetzt.

Die **Uhrzeit des letzten Schreibens** wird beim Verändern einer Datei aktualisiert. Ist die Uhrzeit richtig gestellt, kann mit diesem Feld zusammen mit dem Datum eine Dateiauswahl getroffen werden. Die Uhrzeit enthält in binärer Form Stunde, Minute und Sekunde. Die Sekunde wird nur in 2er-Schritten erfaßt. Es ist zu beachten, daß zuerst das niedere Byte und dann das höhere Byte gespeichert wird.

| **Uhrzeit** | |
|---|---|
| Bit 15...11 | Stunde (0...23) |
| Bit 10... 5 | Minute (0...59) |
| Bit 4... 0 | Sekunden/2 (0...29) |

Das **Datum des letzten Schreibens** ist, ebenso wie die Uhrzeit, in binärer Form abgelegt. Hier ist ebenfalls das niedere Byte zuerst gespeichert.

| **Datum** | |
|---|---|
| Bit 15...9 | Jahr (1980...2099) |
| Bit 8...5 | Monat (1...12) |
| Bit 4...0 | Tag (1...31) |

Die Felder **Nummer des ersten Clusters der Datei** und **Dateilänge** sind als Binärwerte eingetragen, wobei die Speicherung der Worte und Byte zu beachten ist. Der Zugriff über die Clusternummer ist im Zusammenhang mit der FAT (Abschn. 5.1.) erläutert. Die Dateilänge ist in Anzahl Byte angegeben.

Bei umfangreichen Datensammlungen mit sehr vielen Teilgebieten hat sich die Unterteilung in Gruppen (z.B. Kataloge in einer Bibliothek, Aktenordner im Büro) bewährt. Ebenso wird im DOS-Dateisystem durch die Einführung **mehrerer Verzeichnisstufen (hierarchisches Indexsystem)** eine effektive Organisationsform erreicht. Der **Zugriffsweg (Pfad)** hat dabei eine Baumstruktur. Das folgende Bild 5.1 zeigt ein Beispiel für einen Zugriffsbaum.

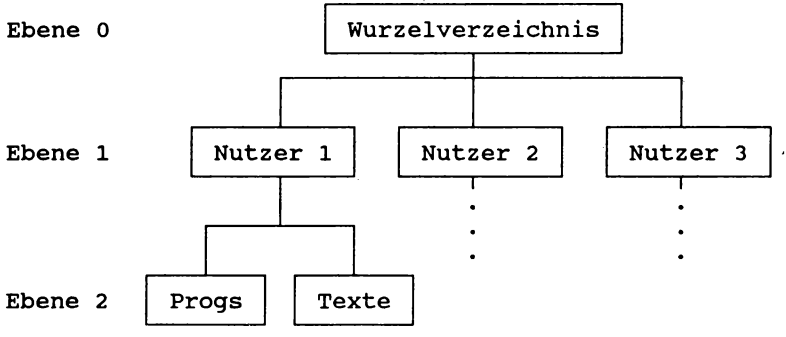



**Bild 5.1. Beispiel einer hierarchischen Verzeichnisstruktur**

Ein Unterverzeichnis enthält zwei Einträge, die DOS für die eigene Verwaltung braucht. Der eine Verzeichniseintrag enthält im Namensfeld einen Punkt ('.' = 2Eh) und verweist auf sich selbst (Clusternummer des Unterverzeichnisses). Der zweite Eintrag enthält zwei Punkte ('..' = 2E2Eh) und verweist auf das übergeordnete Verzeichnis. Ist die Clusternummer = 0, so handelt es sich um das Wurzelverzeichnis. Diese beiden Verzeichniseinträge erscheinen auch in der Ausgabe des DOS-Kommandos DIR.

Zur Verwaltung der Verzeichnisse gibt es DOS-Funktionen, die im Abschn. 5.4. beschrieben werden.

### 5.2.2. Dateiverwaltung der DOS-Version 1.00

In diesem Buch soll nicht im Detail auf die Dateiverwaltung der Version 1.00 eingegangen werden, da die neuen Systemdienste ab Version 2.00 wesentlich mehr leisten und auch von Microsoft empfohlen wird, die alten Dienste nicht mehr zu verwenden. Die in

der Übergangszeit erforderliche Rücksicht auf Nutzer der Version 1.00 ist heute nicht mehr erforderlich. Alle wichtigen Programmpakete unterstützen bereits das hierarchische Dateisystem und basieren damit auf den neuen Systemdiensten.

Das Dateisystem der Version 1 ist einstufig. Die Dateidienste arbeiten immer mit dem aktuellen Verzeichnis (in der Regel mit dem Wurzelverzeichnis). Pfadnamen sind nicht bekannt.

Die folgende Aufzählung der Dateifunktionen (Tafel 5.10) dient nur der Information, um evtl. bei der Fehlersuche in alten Programmen die Bedeutung eines Funktionsaufrufs zu ermitteln.

**Tafel 5.10. Dateifunktionen der DOS-Version 1.00**

| Dateifunktionen der DOS-Version 1.00 | |
|---|---|
| **Funkt.** | **Bedeutung** |
| **Datei** | |
| 16h | Datei anlegen (create file) |
| 0Fh | Datei eröffnen (open file) |
| 10h | Datei schließen (close file) |
| 17h | Datei umbenennen (rename file) |
| 13h | Datei löschen (delete file) |
| **Lesen und Schreiben** | |
| 14h | Datensatz sequentiell lesen (sequ. read) |
| 15h | Datensatz sequentiell schreiben (sequ. write) |
| 21h | Datensatz wahlfrei lesen (random read) |
| 22h | Datensatz wahlfrei schreiben (random write) |
| 27h | Datensätze wahlfrei lesen (random block read) |
| 28h | Datensätze wahlfrei schreiben (random block w.) |
| **Dateisuche im Verzeichnis** | |
| 11h | ersten Verzeichniseintrag suchen (search first) |
| 12h | nächsten Verzeichniseintrag suchen (s. next) |
| **allg. Verwaltung** | |
| 23h | Dateigröße bestimmen (get file size) |
| 28h | Dateigröße verändern (CX = 0) |
| 24h | relative Satznummer setzen (set rel. record) |

Die hier genannten Dateifunktionen setzen eine aufwendige Verwaltung im Programm mit Hilfe eines Dateikontrollblocks FCB (file control block) voraus. Der Anwender ist für einen großen Teil der Verwaltungsarbeit selbst verantwortlich. Man kann dadurch natürlich auch leichter eigene Modifikationen unterbringen als bei der Dateiverwaltung von Version 2.00. Ohne weitere Erläuterungen wird das Format des FCB hier nur angeführt, weil in

einigen Programmen mitunter Verweise auf den FCB auftauchen.

**Tafel 5.11. Dateikontrollblock FCB**

| Dateikontrollblock FCB | | |
|---|---|---|
| **Offset** | **Länge** | **Bedeutung** |
| 00h | 1 | Laufwerk |
| 01-08h | 8 | Dateiname |
| 09-0Bh | 3 | Namenerweiterung |
| 0C-0Dh | 2 | aktueller Block |
| 0E-0Fh | 2 | Satzlänge |
| 10-13h | 4 | Dateilänge |
| 14-15h | 2 | Datum des letzten Schreibens |
| 16-17h | 2 | Uhrzeit des letzten Schreibens |
| 18-1Fh | 8 | reserviert |
| 20h | 1 | Byteposition |
| 21-24h | 4 | relative Satznummer |

Wenn man das FCB-Format mit dem Verzeichniseintrag vergleicht, so findet man einander entsprechende Felder. In einigen Fällen kann der FCB weitere Felder vor dem Standard-FCB belegen (maximal 7 Byte). Der FCB ist ab Version 2.00 nicht mehr erforderlich, da alle wichtigen Informationen direkt durch DOS verwaltet werden.

## 5.3. Dateiverwaltung ab DOS-Version 2.00

### 5.3.1. Grundlegende Konzepte der Dateiverwaltung

Als eine der wichtigsten Aufgaben von DOS ist die Verwaltung der Dateien anzusehen. Ein sehr großer Teil der Systemfunktionen ist daher auch mit diesem Aufgabengebiet verbunden. Die Dateiverwaltung ist vollständig innerhalb der Systemkomponente DOS realisiert.

Ähnlich wie beim Betriebssystem UNIX (bzw. XENIX) wird eine Datei durch eine 16-Bit-Zahl gekennzeichnet, die **Dateinummer (file handle) oder Kanal (channel)** genannt wird. Diese Nummer ersetzt den FCB der Version 1.00. Beim Anlegen oder Eröffnen einer Datei wird vom System eine eindeutige Dateinummer vergeben, die in nachfolgenden DOS-Aufrufen zur Identifikation der Datei verwendet wird. Die maximale Anzahl gleichzeitig eröffneter Datei wird durch einen konfigurierbaren Parameter (FILES = n) in der Datei CONFIG.SYS (vgl. Abschn. 4.2.) festgelegt. Einige Dateinum-

mern sind immer vorhanden und standardmäßig durch DOS eröffnet und Geräten zugewiesen. Es gilt die in Tafel 5.12 angegebene Festlegung, die z.T. durch Systemrufe (46h) oder Zuweisung in der Kommandozeile (redirection) mit den Zeichen '>', '<' oder '>>' verändert werden kann.

**Tafel 5.12. Standarddateinummern**

| Standardzuweisungen für Dateinummern 0...4 | | |
|---|---|---|
| **Nr.** | **Verwendung** | **Gerät** |
| 0 | Standardeingabe (stdin) | CON: |
| 1 | Standardausgabe (stdout) | CON: |
| 2 | Standardfehlerausgabe (stderr) | CON: |
| 3 | Standardhilfsgerät (stdaux) | COM: AUX: |
| 4 | Standarddrucker (stdprn) | PRN: |

Aus dieser Zuweisung ist zu entnehmen, daß die Dateiverwaltung im DOS **keine Unterscheidung zwischen Datei und Gerät** macht. Geräte sind entweder standardmäßig im DOS vorhanden oder werden bei der Konfigurierung (Datei CONFIG.SYS, DEVICE-Parameter) als neues Gerät eingebunden. Bei Einhaltung der Schnittstellenbedingungen kann jedes Gerät in das DOS eingegliedert werden. Dieser Sachverhalt vereinfacht die Programmierung erheblich und unterstützt die Geräteunabhängigkeit der Programme. Man kann z.B. eine Ausgabe auf den Drucker mit dem gleichen Programm auch in eine Datei machen.

Ebenfalls von UNIX wurde die Darstellung von Zeichenketten als ASCIIZ-Strings übernommen. Ein **ASCIIZ-String** ist eine Zeichenfolge variabler Länge, die mit einer Null (00h) abgeschlossen wird. Die Null sollte daher nicht als Zeichen in Strings verwendet werden. Der Dateiname wird einschließlich der Geräteangabe und des Suchpfades bei der Eröffnung oder beim Neuanlegen einer Datei als ASCIIZ-String angegeben. Der ASCIIZ-String für vollständige Dateibezeichner ist in DOS einheitlich geregelt. Sowohl beim Aufruf der Dateifunktionen als auch bei der Eingabe über COMMAND hat ein Dateibezeichner folgenden Aufbau:

[**Gerät:**][**Suchpfad**]**Dateiname**[**.Namenserweiterung**][00h]

Alle Angaben sind wahlweise und werden, wenn sie nicht angegeben werden, durch Standardwerte ersetzt. Als Gerät wird das aktuelle Laufwerk verwendet. Standardsuchpfad ist das aktuelle Verzeichnis, und als Dateinamen werden Leerzeichen eingetragen. Im Suchpfad wird der vollständige Zugriffsweg über die erforderlichen Indexstufen angegeben. Die Indexstufen werden durch das

Trennsymbol '\' voneinander abgegrenzt. Das Wurzelverzeichnis wird durch ein einzelnes Trennsymbol angegeben. Weitere Angaben dazu im Abschn. 6. bei der Beschreibung der DOS-Kommandos.

Die Dateifunktionen melden Fehlersituationen in einer einheitlichen Form mit einem **Standardfehlercode**, der im Register AX nach einem fehlerhaften Aufruf (Kennzeichnung CF = 1) zur Auswertung bereitgestellt wird. Dieser Standardfehlercode kann durch den Aufruf der DOS-Funktion "Fehlercodes lesen (59h)" um weitere Informationen ergänzt werden. Alle Fehlerinformationen bilden zusammen den **erweiterten Fehlercode**. Die komplette Fehlerliste ist im Anhang zu finden; weitere Informationen zur Fehlerbehandlung stehen bei der DOS-Funktion 59h.

Im DOS wird eine **Datei als Bytefolge (string of bytes)** angesehen. Die interne Blockung und der Verkehr mit den Geräten sind für das Programm nicht sichtbar (transparent). Ein Anwendungsprogramm mu$\beta$ eine spezielle Satzstruktur auf diese Bytefolge abbilden. Das Lesen oder Schreiben einer Datei erfolgt durch Angabe eines Puffers und einer Byteanzahl, die die Anzahl der in den Puffer zu lesenden oder aus dem Puffer zu schreibenden Bytes angibt.

Mit der Version 3.10 wurde die Möglichkeit vorgesehen, da$\beta$ mehrere Prozesse auf eine Datei zugreifen. Der gleichzeitige Zugriff ist z.B. bei der Installation eines Netzwerks oder bei der Realisierung von Multi-Tasking innerhalb von DOS erforderlich.

### 5.3.2. DOS-Funktionen zur Dateiverwaltung (Übersicht)

In diesem Abschnitt soll eine kurze Übersicht zu Dateiverwaltungsfunktionen im DOS gegeben werden. Eine genaue Beschreibung der Funktonsaufrufe folgt im Abschn. 5.5. Die Reihenfolge der Beschreibung ist in beiden Abschnitten identisch und versucht, die Funktionen in Gruppen einzuteilen, weil sonst leicht die Übersicht verlorengeht. Der Anhang enthält eine vollständige Liste der Funktionsaufrufe in der Reihenfolge der Funktionsnummern. Folgende Funktionsgruppen lassen sich unterscheiden:

- **Gerätefunktionen für Standardgeräte und Laufwerke (Tafel 5.13)**
- **Verzeichnisfunktionen (Tafel 5.14)**
- **Dateifunktionen zur Ein- und Ausgabe (Tafel 5.15)**
- **sonstige Datenverwaltungsfunktionen (Tafel 5.16).**

Es existieren noch weitere Dateifunktionen, die jedoch für die Dateiverwaltung von Version 1.00 bestimmt sind (vgl. Abschn. 5.2.2.). Sie werden darum hier nicht mit aufgeführt. Im Abschn.

5.5. werden die Funktionen in dieser Reihenfolge beschrieben.

**Tafel 5.13. DOS-Funktionen zur Geräteansteuerung**

| Gerätefunktionen | |
|---|---|
| **Nr.** | **Funktion** |
| 0Dh | Laufwerk zurücksetzen, Puffer leeren (reset disk) |
| 0Eh | aktuelles Laufwerk einstellen (select disk) |
| 19h | aktuelles Laufwerk feststellen (get current disk) |
| 1Bh | Parameter akt. Laufw. lesen (get def. drive data) |
| 1Ch | Parameter belieb. Laufwerk lesen (get drive data) |
| 44h | Gerätesteuerung mit Treiber-IOCTL |

**Tafel 5.14. Verzeichnisfunktionen**

| Verzeichnisfunktionen | | |
|---|---|---|
| **Nr.** | **Funktion** | |
| 39h | Verzeichnis anlegen (create directory) | MKDIR |
| 3Ah | Verzeichnis löschen (remove directory) | RMDIR |
| 3Bh | Verzeichnis wechseln (change current dir) | CHDIR |
| 41h | Eintrag löschen (delete dir. entry - | UNLINK |
| 43h | Attribute setzen/lesen (get/set file attr) | CHMOD |
| 47h | aktuelles Verzeichnis melden (get current dir) | |
| 4Eh | ersten Eintrag suchen (find first file) | |
| 4Fh | nächsten Eintrag suchen (find next file) | |
| 56h | Dateinamen ändern (change directory entry) | |
| 57h | Datum/Uhrzeit setzen/lesen (get/set date/time) | |

**Tafel 5.15. Funktionen zur Arbeit mit Dateien**

| Dateifunktionen | |
|---|---|
| **Nr.** | **Funktion** |
| 3Ch | Kanal anlegen (create handle) |
| 3Dh | Kanal eröffnen (open handle) |
| 3Eh | Kanal schließen (close handle) |
| 3Fh | Kanal lesen (read handle) |
| 40h | Kanal schreiben (write handle) |
| 42h | Kanalposition verändern (move file pointer) |
| 45h | Kanal duplizieren (duplicate file handle) |
| 46h | Kanal kopieren (force duplicate file handle) |
| 5Ah | Kanal temporär anlegen (create temporary file) |
| 5Bh | Kanal neu anlegen (create new file) |
| 5Ch | Dateibereich sperren/freigeben (lock/unlock) |

**Tafel 5.16. Sonstige Datenverwaltungsfunktionen**

| Sonstige Datenverwaltungsfunktionen | |
|---|---|
| **Nr.** | **Funktion** |
| 29h | Dateinamen in String suchen (parse file name) |
| 2Eh | Status für Verifizieren ändern (set verify flag) |
| 54h | Status für Verifizieren lesen (get verify state) |
| 1Ah | DTA-Adresse setzen (set disk transfer address) |
| 2Fh | DTA-Adresse lesen (get disk transfer address) |
| 36h | Platz auf Laufwerk lesen (get disk free space) |

## 5.4. Spezielle Funktionen

Im DOS sind einige interessante Funktionen realisiert, die aus der Funktionsbeschreibung nicht unbedingt zu erkennen sind. Es handelt sich um Funktionen zur

- **Kanalumlenkung (redirection)**
- **Kanalvererbung (pipes)**
- **Realisierung von Datenfiltern.**

Die **Kanalumlenkung** wird durch die Funktion 46h (Kanal kopieren) realisiert. Durch diese Funktion ist eine unkomplizierte Neufestlegung einer Ein- oder Ausgabe möglich. Die Anpassung an konkrete Bedingungen ist z.B. manchmal erforderlich, um eine Ausgabe statt auf Bildschirm in eine Datei zu schreiben.

Als **Kanalvererbung** bezeichnet man die Tatsache, daß beim Aufruf eines Kindprozesses alle geöffneten Kanäle des rufenden Prozesses weitergegeben werden. Alle Standardkanäle werden, da sie immer durch DOS geöffnet werden, zur Verfügung gestellt. So kann z.B. die Standardausgabe (stdout) des Vaterprozesses direkt als Standardeingabe (stdin) des Kindprozesses gelesen werden, wenn eine Umlenkung programmiert wird. Im Multi-Task-Betrieb wäre die direkte Weiterverarbeitung jedes einzelnen Zeichens möglich. Da beim DOS diese Betriebsart noch nicht realisiert ist, sollte man einen bestimmten Dateiabschnitt bearbeiten (z.B. Größe eines Puffers oder eines Satzes), damit der Organisationsaufwand nicht zu groß wird.

Der Aufbau von **Filtern** ist mit den gleichen Mechanismen realisierbar. Diese Dateiverarbeitungsform wird besonders auf Kommandoebene benutzt, um ganze Verarbeitungsketten zu realisieren, z.B. kann die selektierende Funktion des DIR-Kommandos dazu genutzt werden, eine Dateiauswahl für die weitere Verarbeitung durch das SORT-Kommando zu treffen. Die einfachen DOS-Kommandos,

wie z.B. FIND, DIR, SORT, MORE, TYPE, können als Filter eingesetzt werden. Die dazu erforderliche Dateiumlenkung wird durch die Spezialzeichen '<', '>' und '|' gesteuert (vgl. Abschnitte 6. und 11.).

## 5.5. DOS-Funktionen zur Dateibehandlung

Die DOS-Funktionen werden, wie schon die BIOS-Funktionen, als Assemblerroutinen definiert und lassen sich aus allen Programmiersprachen aufrufen. In diesem Buch wird die Programmiersprache Turbo-Pascal als Beispiel verwendet. Im Abschn. 8. werden die Interfacebedingungen für die anderen Programmiersprachen (Versionen von Microsoft) definiert und in einigen Beispielen im Abschn. 11. erläutert.

Alle DOS-Funktionen und damit auch die Funktionen zur Datenverwaltung werden über den DOS-Interrupt 21h aufgerufen (vgl. Abschn. 4.4.).

### 5.5.1. Gerätefunktionen

Die Gerätefunktionen wirken direkt auf die durch Treiberprogramme angesteuerten Geräte. Obwohl die Dateiverwaltung keine Unterschiede zwischen Geräten und Dateien macht und beide durch Kanäle steuert, sind doch einige spezielle Funktionen für die direkte Steuerung vorhanden. Die Funktionen werden hier einzeln vorgestellt.

| **Laufwerk zurücksetzen (reset disk)** | **Int 21-0D** |
|---|---|
| **Aufruf**<br>AH = 0Dh | **Ergebnis**<br>- |

Mit dieser Funktion wird sichergestellt, daß alle internen Plattenpuffer zurückgeschrieben werden, falls der Inhalt verändert worden ist. Nach diesem Aufruf sind alle Puffer als frei gekennzeichnet. In der Regel ist dieser Aufruf nicht erforderlich, da Dateien durch den Aufruf Close geschlossen werden sollten, bei dem jede Datei ordnungsgemäß abgeschlossen wird und die belegten Puffer freigegeben werden. Diese Funktion wird mitunter benutzt, um einen definierten Zustand herzustellen, wenn z.B. die laufende Verarbeitung durch Ctrl-C abgebrochen wird.

| **Aktuelles Laufwerk einstellen (select disk) Int 21-0E** | |
|---|---|
| **Aufruf**<br>AH = 0Eh<br>DL = Nummer des Laufwerks<br>(0 = A:, 1 = B:,...) | **Ergebnis**<br>AL = Anzahl log. Laufwerke |

Mit dieser Funktion wird das in DL angegebene Laufwerk als aktuelles Laufwerk eingestellt, das bei fehlender Geräteangabe als Standardwert eingetragen wird.

| **Aktuelles Laufwerk lesen (get current disk) Int 21-19** | |
|---|---|
| **Aufruf**<br>AH = 19h | **Ergebnis**<br>AL = aktuelles Laufwerk<br>(0 = A:, 1 = B:, ...) |

Mit dieser Funktion wird das aktuelle Laufwerk bestimmt, das mit der Funktion 0Eh eingestellt wurde.

| **Parameter akt. Laufwerk lesen (get default drive data) Int 21-1B** | |
|---|---|
| **Aufruf**<br>AH = 1Bh | **Ergebnis**<br>DS:BX = Adr. FAT-ID-Byte<br>AL = Clustergröβe<br>CX = Sektorgröβe<br>DX = Clusteranzahl |

Mit dieser Funktion werden einige Parameter des Urladereintrags bzw. des BIOS-Parameterblocks bereitgestellt.

| **Parameter beliebiges Laufwerk lesen (get drive data) Int 21-1C** | |
|---|---|
| **Aufruf**<br>AH = 1Ch<br>DL = Gerätenummer<br>(0 = aktuelles Gerät)<br>(1 = A:, 2 = B:,...) | **Ergebnis**<br>DS:BX = Adr. FAT-ID-Byte<br>AL = Clustergröβe<br>CX = Sektorgröβe<br>DX = Clusteranzahl |

Diese Funktion stellt die gleichen Daten bereit wie die Funktion 1Bh. Hier kann das Gerät jedoch ausgewählt werden. Es ist zu beachten, daβ die Gerätenummer von der sonst gebräuchlichen Zählung abweicht.

| **Gerätesteuerung mit Treiber-IOCTL** | **Int 21-44nn** |
|---|---|
| **Aufruf**<br>AH = 44h<br>BX = Kanalnummer<br>DS:DX = Adr. Puffer<br>CX = Anzahl Bytes | **Ergebnis**<br>CF = 0 kein Fehler,<br>AX = Anzahl übertr. Bytes<br>CF = 1,<br>AX = Fehlernummer |
| AL = 00h **Status lesen** (DX = Treiberattribute)<br>AL = 01h **Status ändern** (DX = Treiberattribute)<br>AL = 02h **Steuerstring lesen (Kanal)**<br>AL = 03h **Steuerstring schreiben (Kanal)**<br>AL = 04h **Steuerstring lesen (Laufwerk in BL)**<br>AL = 05h **Steuerstring schreiben (Laufwerk in BL)**<br>AL = 06h **Prüfen Eingabebereitschaft**<br>AL = 07h **Prüfen Ausgabebereitschaft**<br>AL = 08h **Prüfung Datenträgerwechsel**<br>AL = 0Bh **Wiederholungszähler setzen bei Mehrfachzugriff** | |

Diese Funktion ist durch die vielen Unterfunktionen etwas unübersichtlich; sie stellt jedoch die direkte Verbindung zum Treiber dar. Bei den Unterfunktionen 00h und 01h werden in DX die Treiberattribute gespeichert. Ist Bit 7 gesetzt, so handelt es sich um ein Gerät, und die Bits 0...4 entsprechen den Bits der Treiberattribute. Bit 5 zeigt den Verarbeitungsmodus (1 = binär, 0 = ASCII) an, wenn es sich um Standardeingabe oder Standardausgabe handelt. Bit 6 ist gesetzt, wenn EOF erreicht ist. Ist Bit 7 = 0, so handelt es sich um ein Laufwerk, das in Bit 0...5 angezeigt wird (0 = A:, 1 = B:). Die Funktionen 02h...05h werden nur ausgeführt, wenn das Bit 14 im Treiberattributfeld gesetzt ist. Die Anzahl übertragener Bytes wird in CX eingetragen. Bei den Funktionen 06h und 07h wird im Register AL angezeigt, ob der Kanal zum Lesen oder Schreiben bereit ist (AL = FFh bereit, AL = 00h nicht bereit). Funktion 08h wird benutzt, um festzustellen, ob der Datenträger gewechselt werden kann. Die Laufwerksnummer wird in BL übergeben, und die Anzeige erfolgt mit AX = 0 (Datenträger kann gewechselt werden). Mit der Funktion 0Bh kann festgelegt werden, wie oft ein Zugriff bei Fehlern wiederholt werden soll. Die Anzahl wird in CX und die Zeit zwischen den Versuchen in DX übergeben.

### 5.2.2. Verzeichnisfunktionen

Die Verzeichnisfunktionen legen die hierarchische Dateistruktur von DOS fest und realisieren einen Zugriffspfad. Weitere Informationen zur Verzeichnisstruktur sind im Abschn. 5.2. zu finden.

| Verzeichnis anlegen (create directory) Int 21-39 | |
|---|---|
| **Aufruf**<br>AH = 39h<br>DS:DX = Pfadname (ASCIIZ) | **Ergebnis**<br>CF = 0 kein Fehler<br>CF = 1 Fehler<br>AX = Fehlernummer |

Mit dieser Funktion wird ein Verzeichniseintrag auf der durch Pfadname angegebenen Ebene realisiert.

| Verzeichnis löschen (remove directory) Int 21-3A | |
|---|---|
| **Aufruf**<br>AH = 3Ah<br>DS:DX = Pfadname (ASCIIZ) | **Ergebnis**<br>CF = 0 kein Fehler<br>CF = 1 Fehler<br>AX = Fehlernummer |

Mit dieser Funktion kann ein leeres Verzeichnis gelöscht werden. Sind im Verzeichnis noch Eintragungen vorhanden, so erfolgt eine Fehlermeldung, und das Verzeichnis wird nicht gelöscht. Das Löschen von Eintragungen erfolgt mit der DOS-Funktion 21-41.

| Verzeichnis wechseln (change current dir) Int 21-3B | |
|---|---|
| **Aufruf**<br>AH = 3Bh<br>DS:DX = Pfadname (ASCIIZ) | **Ergebnis**<br>CF = 0 kein Fehler<br>CF = 1 Fehler<br>AX = Fehlernummer |

Mit dieser Funktion wird das aktuelle Verzeichnis eingestellt, das bei fehlender Angabe zum Suchpfad einer Datei ergänzt wird.

| Eintrag löschen (delete directory entry) Int 21-41 | |
|---|---|
| **Aufruf**<br>AH = 41h<br>DS:DX = Pfadname (ASCIIZ) | **Ergebnis**<br>CF = 0 kein Fehler<br>CF = 1 Fehler<br>AX = Fehlernummer |

Der zu einer Datei gehörende Verzeichniseintrag wird gelöscht, indem das erste Zeichen im Dateinamenfeld mit E5h über-

schrieben wird. Eine Datei mit dem Attribut <nur Lesen> kann nicht gelöscht werden; es ist vorher der Funktionsaufruf 43h erforderlich, um das Attribut zu verändern.

| **Attribute setzen/lesen (get/set file attr) Int 21-43** | |
|---|---|
| **Aufruf**<br>AH = 43h<br>DS:DX = Pfadname (ASCIIZ)<br>AL = 00h **Attribute lesen**<br>AL = 01h **Attribute setzen**<br>CX = Attribute | **Ergebnis**<br>CF = 0 kein Fehler<br>CF = 1 Gehler<br>AX = Fehlernummer<br>CX = Attribute (lesen) |

Mit diesem Funktionsaufruf können die Attribute <Verzeichnis> und <Datenträgeretikett> nicht verändert werden. Zu diesem Zweck müßte man die entsprechenden Bits im Verzeichnissektor mit einem Dienstprogramm oder eigenem Programm direkt verändern. Alle anderen Attribute lassen sich verändern. Die Kennzeichnung als Systemdatei bzw. als versteckte Datei ist als Datenschutz daher nur bedingt geeignet. Die Kennzeichnung eines Dateieintrags mit dem Attribut <Verzeichnis> ist möglich, führt aber gewollt oder ungewollt zu einer gewissen Verwirrung. Gegen versehentliches Löschen ist das Attribut <nur Lesen> ein wirksamer Schutz. Die Bitbelegung der Dateiattribute ist in Tafel 5.9 gezeigt.

| **Aktuelles Verzeichnis melden (get cur. dir) Int 21-47** | |
|---|---|
| **Aufruf**<br>AH = 47h<br>DS:SI = Adresse Pfadname<br>DL = Laufwerk<br>(0 = akt, 1 = A:, ...) | **Ergebnis**<br>CF = 0 kein Fehler<br>CF = 1 Fehler<br>AX = Fehlernummer |

Der volle Pfadname für das in DL spezifizierte Laufwerk wird im Pufferbereich (DS:SI) bereitgestellt. Es ist zu beachten, das für jedes Laufwerk ein aktuelles Verzeichnis definiert ist.

| **Ersten Eintrag suchen (find first file) Int 21-4E** | |
|---|---|
| **Aufruf**<br>AH = 4Eh<br>DS:DX = Pfadname (ASCIIZ)<br>CX = Attribute für Suche | **Ergebnis**<br>CF = 0 kein Fehler<br>CF = 1 Fehler<br>AX = Fehlernummer |

Mit dieser Funktion kann eine Suche im durch DS:DX angegebenen Verzeichnis und den in CX stehenden Attributen gestartet werden. Im DTA-Bereich, der mit Funktion 1Ah eingestellt bzw. mit Funktion 2Fh ermittelt werden kann, wird ein Zwischenergebnis nach dem Aufruf abgelegt, der für die weitere Suche im Verzeichnis mit Funktion 4Fh benutzt wird. Der Bereich im DTA hat folgenden Aufbau:

21 Byte reserviert für DOS
1 Byte Attribut der gefundenen Datei
2 Byte Uhrzeit
2 Byte Datum
2 Byte Dateigröße (niederes Wort)
2 Byte Dateigröße (höheres Wort)
13 Byte Dateiname mit Erweiterung.

| **Nächsten Eintrag suchen (find next file) Int 21-4F** | |
|---|---|
| **Aufruf**<br>AH = 4Fh<br>DS:DX = Daten v. Funkt. 4Eh | **Ergebnis**<br>CF = 0 kein Fehler<br>CF = 1 Fehler<br>AX = Fehlernummer |

Dieser Aufruf kann nur erfolgen, wenn mit dem Funktionsaufruf 4Eh vorher die notwendigen Daten in DS:DX abgelegt wurden. Nach dem Aufruf stehen die Daten wie bei Funktion 4Eh im DTA-Bereich. Falls keine Datei mehr gefunden wird, erfolgt eine Fehlermeldung.

| **Dateinamen ändern (change directory entry) Int 21-56** | |
|---|---|
| **Aufruf**<br>AH = 56h<br>DS:DX = ASCIIZ (alt)<br>ES:DI = ASCIIZ (neu) | **Ergebnis**<br>CF = 0 kein Fehler<br>CF = 1 Fehler<br>AX = Fehlernummer |

Mit dieser Funktion wird ein Dateiname auf einen neuen Wert gesetzt. Mit DS:DX wird der alte Dateiname mit vollständigem Suchpfad angegeben und mit ES:DI der neue Name. Das Laufwerk muß, falls es angegeben wird, mit dem alten übereinstimmen. Der Pfad für den neuen Dateinamen darf sich vom alten unterscheiden, wobei automatisch eine Datei von einem Verzeichnis in ein anderes übertragen wird. Versteckte Dateien, Systemdateien und Verzeichnisse lassen sich auf diese Weise nicht umbennen.

| **Datum/Uhrzeit setzen/lesen** | **Int 21-57** |
|---|---|
| **Aufruf**<br>AH = 57h<br>BX = Kanal | **Ergebnis**<br>CF = 0 kein Fehler<br>CF = 1 Fehler<br>AX = Fehlernummer |
| AL = 00h **lesen** | CX = Uhrzeit<br>DX = Datum |
| AL = 01h **setzen**<br>CX = Uhrzeit<br>DX = Datum | |

Datum und Uhrzeit haben die gleiche Form wie im Verzeichniseintrag bzw. müssen vorm Aufruf in diese Form gebracht werden.

### 5.5.3. Dateifunktionen (Ein- und Ausgabe)

Diese Dateifunktionen realisieren die eigentliche Dateiarbeit. Im Programm wird eine Datei mit einem Kanal verknüpft und dann nur noch über die Kanalnummer angesprochen. Der Programmieraufwand ist dadurch relativ gering. Ein Vorteil dieser Arbeitsweise liegt darin, daß vom DOS auch Geräte wie Dateien behandelt werden können.

| **Kanal anlegen (create handle)** | **Int 21-3C** |
|---|---|
| **Aufruf**<br>AH = 3Ch<br>CX = Dateiattribute<br>DS:DX = Dateiname (ASCIIZ) | **Ergebnis**<br>CF = 0 kein Fehler<br>CF = 1 Fehler<br>AX = Kanal bzw. Fehler-Nr. |

Diese Funktion generiert eine neue Datei oder kürzt eine existierende auf die Länge 0. Die Attribute werden so gesetzt, wie sie in CX angegeben werden (Bitbelegung wie im Verzeichniseintrag). Die neuangelegte Datei ist für Lese- und Schreibzugriff geöffnet.

| **Kanal eröffnen (open handle)** | **Int 21-3D** |
|---|---|
| **Aufruf**<br>AH = 3Dh<br>AL = Modus<br>DS:DX = Dateiname (ASCIIZ) | **Ergebnis**<br>CF = 0 kein Fehler<br>AX = Kanal<br>CF = 1<br>AX = Fehlernummer |

Die mit DS:DX angegebene Datei (bzw. Gerät) wird eröffnet, wobei der Modus mit AL festgelegt wird. Der Modus wird wie folgt eingestellt:

AL = 00h Öffnen eines Kanals zum Lesen
AL = 01h Öffnen eines Kanals zum Schreiben
AL = 02h Öffnen eines Kanals zum Lesen und Schreiben.

Mit Bit 7 in AL wird festgelegt, ob Kindprozesse ebenfalls auf die Datei zugreifen können, und die restlichen Bits steuern den Mehrfachzugriff (SHARE), auf den hier nicht näher eingegangen wird. Die Kanalnummer wird bei Ausführung ohne Fehler in AX zurückgeliefert. Mit dieser Funktion lassen sich sowohl Geräte (z.B. CON:, COM1:, PRN:) als auch Dateien einem Kanal zuordnen. Nach der Eröffnung besteht in der Bearbeitung kein Unterschied.

| **Kanal schließen (close handle)** | **Int 21-3E** |
|---|---|
| **Aufruf**<br>AH = 3Eh<br>BX = Kanal | **Ergebnis**<br>CF = 0 kein Fehler<br>CF = 1 Fehler<br>AX = Fehlernummer |

Mit dieser Funktion wird eine Kanalzuweisung aufgehoben. Das Programm wird vom entsprechenden Gerät oder der Datei getrennt. Der zur Datei gehörende Verzeichniseintrag wird auf den aktuellen Stand gebracht und alle von DOS angelegten Puffer freigegeben.

| **Kanal lesen (read handle)** | **Int 21-3F** |
|---|---|
| **Aufruf**<br>AH = 3Fh<br>BX = Kanal<br>CX = Anzahl Bytes<br>DS:DX = Puffer | **Ergebnis**<br>CF = 0 kein Fehler<br>AX = Anzahl gelesener Bytes<br>CF = 1 Fehler<br>AX = Fehlernummer |

Mit dieser Funktion wird eine mit CX angegebene Anzahl Bytes in den Puffer übertragen. In AX wird die tatsächlich gelesene Anzahl eingetragen. Der Lesevorgang beginnt immer an der aktuellen Position, die vorher mit Funktion 42h eingestellt werden kann. Ist als Eingabekanal die Standardeingabe zugewiesen, so kann die Eingabe umgelenkt werden.

| Kanal schreiben (write handle) | Int 21-40 |
|---|---|
| **Aufruf**<br>AH = 40h<br>BX = Kanal<br>CX = Anzahl Bytes<br>DS:DX = Puffer | **Ergebnis**<br>CF = 0 kein Fehler<br>AX = Anzahl geschr. Bytes<br>CF = 1 Fehler<br>AX = Fehlernummer |

Mit dieser Funktion wird in einen Kanal geschrieben, unabhängig davon, ob ein Gerät oder eine Datei zugewiesen ist. Falls der Kanal die Standardausgabe ist, kann eine Umlenkung erfolgen. Kann eine Ausgabe nicht vollständig erfolgen, so ist die Byteanzahl in AX kleiner als die geforderte Anzahl (z.B. wenn kein Platz auf dem Datenträger vorhanden ist). Es sollte vorher geprüft werden, ob auf dem Datenträger genügend Platz ist.

| Kanalposition verändern (move file pointer) | Int 21-42 |
|---|---|
| **Aufruf**<br>AH = 42h<br>AL = Positioniermodus<br>00h relativ Beginn<br>01h relativ aktuell<br>02h relativ EOF<br>BX = Kanal<br>CX:DX = Verschiebung (Byte) | **Ergebnis**<br>CF = 0 kein Fehler<br>DX:AX = neue Kanalposition<br>CF = 1 Fehler<br>AX = Fehlernummer |

Mit dieser Funktion wird die Kanalposition neu eingestellt. Der durch CX:DX angegebene Wert wird als Integerzahl verwendet. Mit der Positionierart 2 (relativ EOF) kann durch Verschiebung um 0 Byte das Ende der Datei ermittelt werden.

| Kanal duplizieren (duplicate file handle) | Int 21-45 |
|---|---|
| **Aufruf**<br>AH = 45h<br>BX = Kanal | **Ergebnis**<br>CF = 0 kein Fehler<br>AX = neue Kanalnummer<br>CF = 1 Fehler<br>AX = Fehlernummer |

Mit diesem Funktionsaufruf wird eine 2. Kanalnummer für das gleiche Gerät oder die gleiche Datei vergeben. Die Kanäle können unabhängig voneinander geschlossen werden, wobei der Verzeichniseintrag aktualisiert wird. Es existiert jedoch nur eine Kanalposition für beide Kanäle.

| **Kanal kopieren (force duplicate file handle)** | **Int 21-46** |
|---|---|
| **Aufruf**<br>AH = 46h<br>BX = Kanal (Quelle)<br>CX = Kanal (Ziel) | **Ergebnis**<br>CF = 0 kein Fehler<br>CF = 1 Fehler<br>AX = Fehlernummer |

Diese Funktion lenkt einen existierenden Kanal auf einen anderen Kanal um. Falls der 2. Kanal bereits geöffnet ist, wird er vorher geschlossen und erneut geöffnet. Die Funktion ist dann sinnvoll, wenn immer der gleiche Kanal (z.B. Standardausgabe) für eine Aktion genutzt werden soll.

| **Kanal temporär anlegen (create temp. file)** | **Int 21-5A** |
|---|---|
| **Aufruf**<br>AH = 5Ah<br>CX = Attribute<br>DS:DX = Pfadname (ASCIIZ) | **Ergebnis**<br>CF = 0<br>DS:DX = temp. Dateiname<br>CF = 1 Fehler<br>AX = Fehlernummer |

Dieser Funktionsaufruf wird genutzt, um eine Datei mit einem eindeutigen Namen in dem durch den ASCIIZ-String angegebenen Verzeichnis anzulegen. Sinnvoll ist diese eindeutige Dateibezeichnung z.B. für temporäre Dateien, die nach dem Programmlauf nicht mehr benötigt werden. Die Datei muß explizit gelöscht werden.

| **Kanal neu anlegen (create new file)** | **Int 21-5B** |
|---|---|
| **Aufruf**<br>AH = 5Bh<br>CX = Attribute<br>DS:DX = Dateiname (ASCIIZ) | **Ergebnis**<br>CF = 0 kein Fehler<br>CF = 1 Fehler<br>AX = Fehlernummer |

Diese Funktion erfüllt den gleichen Zweck wie die Funktion 3Ch. Hier wird jedoch ein Fehler angezeigt, wenn die Datei bereits existiert.

| **Dateibereich sperren/freigeben (lock/unlock)** | **Int 21-5C** |
|---|---|
| **Aufruf**<br>AH = 5Ch<br>AL = 00h **sperren (lock)**<br>AL = 01h **freigeben (unlock)**<br>BX = Kanal<br>CX:DX = relative Position<br>SI:DI = Länge | **Ergebnis**<br>CF = 0 kein Fehler<br>CF = 1 Fehler<br>AX = Fehlernummer |

Diese Funktion ist zur Unterstützung des Mehrfachzugriffs gedacht und dient dazu, einzelne Bereiche (Regionen) einer Datei zeitweilig gegen Veränderungen zu schützen. Die Position ist relativ zum Dateianfang anzugeben. Die Registerpaare CX:DX und SI:DI bilden jeweils eine Integerzahl von 4 Byte. Wenn man versucht, in einem gesperrten Bereich zu lesen oder zu schreiben, dann versucht DOS mehrfach den Zugriff, bevor ein Interrupt 24h (schwerwiegender Gerätefehler) ausgelöst wird. Die Zahl der Wiederholungen und den Abstand zwischen zwei Versuchen kann man mit der Funktion 440Bh einstellen. Vor dem Schließen einer Datei sind alle Sperrungen aufzuheben. Von der Funktion wird eine Fehlermeldung geliefert, wenn ein gesperrter Bereich nochmals gesperrt werden soll. Auf diese Weise kann man feststellen, ob ein Bereich frei ist. Die Funktion ist nur im Netzwerk sinnvoll.

### 5.5.4. Sonstige Datenverwaltungsfunktionen

In diesem Abschnitt sind alle restlichen Funktionen zusammengefaßt, die mit der Dateiverwaltung zu tun haben. Nicht in die Beschreibung aufgenommen sind die traditionellen Dateifunktionen der DOS-Version 1.

| **Status für Verifizieren ändern (set verify)** | **Int 21-2E** |
|---|---|
| **Aufruf**<br>AH = 2Eh<br>AL = 00h Verify off<br>AL = 01h Verify on | **Ergebnis**<br>- |

Mit dieser Funktion kann das Verifizieren nach Schreiboperationen auf den Laufwerken an- (on) oder abgeschaltet (off) werden. Diese Verifizierfunktion sollte nur in Sonderfällen genutzt werden, da die Schreiboperationen die doppelte Zeit beanspruchen und die Disketten und Festplatten sehr selten Fehler erzeugen. Die gleiche Wirkung wie diese Funktion hat auch das interne DOS-Kommando VERIFY. Die aktuelle Stellung des Verifizierschalters kann mit der Funktion 54h abgefragt werden.

| **Status für Verifizieren lesen (get verify)** | **Int 21-54** |
|---|---|
| **Aufruf**<br>AH = 54h | **Ergebnis**<br>AL = 00h Verify off<br>AL = 01h Verify on |

Mit dieser Funktion wird der Verifizierschalter abgefragt, der durch Funktion 2Eh oder Kommando VERIFY gesetzt wurde.

| DTA-Adresse setzen (set disk transfer addr.) Int 21-1A | |
|---|---|
| **Aufruf**<br>AH = 1Ah<br>DS:DX = DTA-Puffer | **Ergebnis**<br>- |

Der DTA-Puffer wurde bei den FCB-Funktionen zur Dateiarbeit benutzt; für die Dateifunktionen der Version 2.00 ist er lediglich als Puffer für die Suche im Verzeichnis erforderlich. Im PSP ist ein Standard-DTA eingerichtet, so daß man diese Funktion nicht benötigt.

| DTA-Adresse lesen (get disk transfer address) Int 21-2F | |
|---|---|
| **Aufruf**<br>AH = 2Fh | **Ergebnis**<br>ES:BX = DTA-Puffer |

Diese Funktion stellt die von DOS intern verwendete DTA-Adresse bereit, die durch Funktion 1Ah gesetzt werden kann.

| Platz auf Laufwerk lesen (get disk free) Int 21-36 | |
|---|---|
| **Aufruf**<br>AH = 36h<br>DL = Laufwerk<br>(0 = aktuelles Lw.)<br>(1 = A:, 2 = B:,...) | **Ergebnis**<br>BX = freie Cluster<br>DX = Cluster insgesamt<br>CX = Sektorgröße in Byte<br>AX = FFFFh DL falsch |

Diese Funktion stellt einige Parameter des in DL angegebenen Laufwerks zur Verfügung. In BX wird z.B. die für das Anlegen von Dateien erforderliche Größe des freien Bereiches auf einem Laufwerk eingetragen. Die Fehlermeldung in AX weist auf eine falsche Laufwerksangabe hin.

## 5.6. Format von Programmdateien

Eine der wesentlichen Weiterentwicklungen des 8086 gegenüber dem 8080 ist die Erweiterung des Adreßraums und Nutzung von Seg-

menten. Größere Programme werden von dieser Möglichkeit Gebrauch machen. Für kleinere Programme ist in der Regel die Größe eines Segments von 64 KByte ausreichend. Beide Programmodelle

- **COM-Format** (Programmgröße bis 64 KByte)
- **EXE-Format** (Programmgröße bis max. Speicherbereich)

sind in MS-DOS realisiert. Der Programmverbinder LINK erzeugt grundsätzlich das segmentierte EXE-Programmformat (Dateierweiterung .exe). Das DOS-Kommando EXE2BIN wandelt das EXE-Format in das kompaktere COM-Format (Dateierweiterung .com) um. Da das EXE-Format aus mehreren Segmenten besteht, die beliebig im Speicher angeordnet werden können, ist beim Laden die entsprechende Anpassung der verschiebbaren Adressen vorzunehmen. Das COM-Format entspricht bereits dem Speicherabbild; lediglich die Segmentregister sind einzustellen. Die COM-Programme sind dadurch meistens schneller geladen und geringfügig schneller bei der Abarbeitung. Aus diesem Grund liegen die meisten DOS-Kommandos im COM-Format vor.

Programme werden in die Speicherverwaltung des DOS einbezogen. Der für die Programmverwaltung erforderliche Steuerblock PSP (Program Segment Prefix) ist im Abschn. 4. näher erläutert.

**COM-Format**

Das COM-Format entspricht dem Speicherabbild des Programms. Nach dem Laden ist nur ein Segment vorhanden. Die Programmgröße ist dadurch auf 64 KByte beschränkt. Die Realisierung größerer Programme ist nur durch Verwendung von Overlays möglich. Am Segmentanfang steht das PSP mit einer Länge von 256 (FFh) Byte. Das Programm beginnt bei 100h.

Beim Laden werden die folgenden Schritte ausgeführt:

- Belegen des gesamten verfügbaren Speichers (Startsegment)
- Aufbau eines PSP
- Einlesen des Lademoduls nach Startsegment
- Register laden (aktuelle Adressen).

Das COM-Format kann nicht direkt vom Linker erzeugt werden. Mit Turbo-Pascal 3.0 oder DEBUG erzeugte Programme können im COM-Format ausgegeben werden. Unter bestimmten Bedingungen kann ein EXE-Format durch das Programm EXE2BIN in das COM-Format umgewandelt werden. Falls eine Umwandlung nicht möglich ist, erfolgt eine Fehlermitteilung. Die Bedeutung des COM-Formats geht zurück, da der geringere Platzbedarf heute keine Rolle mehr spielt. Im

OS/2, der Weiterentwicklung von DOS zu einem Multi-task-Betriebssystem, gibt es das COM-Format nicht mehr. Auch der Compiler von Turbo-Pascal 4.0 erzeugt das EXE-Format. Die Verschieblichkeit der EXE-Programme läßt einerseits leicht die Steuerung mehrerer Programme im Speicher zu; andererseits spielt hierbei die Grenze von 64 KByte keine Rólle mehr.

**EXE-Format**

Die durch den Linker erzeugte EXE-Datei ist ein Programm, bei dem alle Adressen relativ zu 0000:0000 berechnet sind. Dieser Bereich ist im Speicher nicht frei, und daher wird das Programm beim Laden verschoben. Die intern verwendeten Adressen sind dabei zu korrigieren. Die Korrekturinformation wird ebenfalls vom Linker bereitgestellt. Ein EXE-Programm besteht aus zwei Teilen:

- Vorspann (Header)
- Lademodul.

Der **Vorspann** hat eine variable Länge und steht direkt vor dem Lademodul. Er enthält in einem festen Teil Steuerinformationen zur Beschreibung des Vorspanns und des Lademoduls und anschließend die Relokationstabelle mit einer variablen Anzahl von Einträgen zum verschieblichen Laden des Programms. Der Aufbau wird in Tafel 5.17 gezeigt.

**Tafel 5.17. Programmvorspann (header) beim EXE-Format**

| Offset (hex.) | Wert |
|---|---|
| 00-01 | Kennzeichen für EXE-Format (4Dh, 5Ah = 'MZ') |
| 02-03 | Länge des Lademoduls mod 512 (Rest letzte Seite) |
| 04-05 | Länge der EXE-Datei in 512-Byte-Blöcken |
| 06-07 | Anzahl Einträge der Relokationstabelle im Header |
| 08-09 | Header-Größe in 16-Byte-Paragraphen (Start Modul) |
| 0A-0B | Mindestanzahl Paragraphen nach Modul (Minalloc) |
| 0C-0D | maximale Anzahl Paragraphen nach Modul (Maxalloc) |
| 0E-0F | Anfangswert des Stacksegment-Registers (SS) |
| 10-11 | Anfangswert des Stackpointer-Registers (SP) |
| 12-13 | negative Prüfsumme aller Worte der Datei |
| 14-15 | Anfangswert des Programmzählers (IP) |
| 16-17 | Anfangswert des Codesegment-Registers (CS) |
| 18-19 | Offset der Relokatiostabelle zum Beginn der Datei |
| 1A-1B | Overlay-Nummer (0 = residenter Teil) |
| 1C-... | Relokationstabelle |

Die einzelnen Werte werden vom Linker berechnet oder als Parameter beim Binden angegeben. Die Einträge in der Relokationstabelle

(Offset, Segment) werden von der Laderoutine benutzt, um Adressen im Lademodul auf die aktuellen Speicheradressen abzuändern.

Beim Laden werden die folgenden Schritte ausgeführt:

- Einlesen des Vorspanns (fester Teil)
- Anfordern eines Speicherbereichs (Startsegment)
- Aufbau eines PSP
- Einlesen des Lademoduls in Startsegment
- Einlesen der Relokationstabelle in Arbeitsbereich
- Adressenkorrektur (Addition Adresse Startsegment)
- Register laden (Werte aus Vorspann und aktuellen Adressen).

Das Laden eines Programms erfolgt entweder durch den Kommandoprozessor oder durch ein Anwendungsprogramm (vgl. DOS-Funktion 4Bh).

## 5.7. Datensicherung, Datenschutz

Die im Abschn. 6. behandelten Kommandos gelten meistens für Datenträger bzw. Dateien, die den Anforderungen von DOS entsprechen. Zwei wichtige Gesichtspunkte lassen es sinnvoll erscheinen, von diesen Konventionen abzuweichen. Das sind

- **Fragen der Datensicherung**
  (Schutz gegen Verlust durch Löschen, Gerätefehler u.a.)
- **Fragen des Kopierschutzes**
  (Schutz gegen unerlaubtes Lesen, Kopieren u.a.).

In diesem Abschnitt sollen einige Probleme und mögliche Lösungsansätze diskutiert werden, ohne den Fragenkomplex umfassend zu behandeln.

Die **Datensicherung** wird im wesentlichen noch immer durch die Anfertigung von Kopien gewährleistet. Im DOS sind einerseits einige Möglichkeiten vorgesehen, die Datensicherheit zu gewährleisten. Das sind z.B. die Möglichkeiten, mit den Kopierkommandos COPY, XCOPY, DISKCOPY, BACKUP und RESTORE Sicherheitskopien anzufertigen und mit COMP und DISKCOMP zu vergleichen. Andererseits existieren eine Reihe von Zusatzprogrammen, die z.T. wesenlich schneller arbeiten und komfortabler in der Bedienung sind (z.B. FASTBACK, FRESTORE, TURBO-BACKUP, SAFE-GUARD). Für die Sicherung ganzer Festplatten bieten sich die "Tape Streamer" an, die wie beim Magnetband eine Kopie der Festplatte auf Magnetkassette erstellen. Die Schnelligkeit wird bei solchen Spezialkopierprogrammen durch Berücksichtigung spezieller Hardwareeigenschaften

(z.B. direkte Ansteuerung des DMA-Bausteins oder durch Lesen ganzer Sektorfolgen erreicht, wobei keine Rücksicht auf die Dateistruktur genommen wird. Die Anpassung an die Hardwarebedingungen erfolgt meistens durch ein spezielles Installationsprogramm. Bei der Beantwortung der während der Installation gestellten Fragen sollte man sehr gewissenhaft sein, um Datenverlust oder Unvollständigkeit zu vermeiden. Zum Zugriff auf die Sektoren können sehr gut auch die DOS-Funktionen zum absoluten Lesen und Schreiben von Sektoren (Int 25h und 26h) benutzt werden.

Ein weiteres Verfahren, das oft bei der Datensicherung eingesetzt wird, ist die **Datenkompression**. Dabei wird durch geeignete Textmanipulation (Zusammenfassen von gleichen Zeichen oder Ausnutzung von Buchstabenhäufigkeiten u.a.) eine starke Verdichtung der Dateien erreicht. Das ist besonders wichtig, wenn Daten zur Archivierung abgelegt werden. Die Komprimierung ist bei manchen Programmen relativ zeitaufwendig und sollte nur gemacht werden, wenn die Daten nicht ständig benötigt werden. Programme zur Datenkompression sind z.B. ARC, PKARC u.a., die weitgehend das gleiche Verfahren verwenden.

Der **Datenschutz** ist mit der Datensicherung eng verwandt, wird aber aus einem ganz anderen Grund angewendet. So ist z.B. bei Personaldaten, Geschäftsdaten u.ä. der Zugriff nur ganz bestimmten Personen vorbehalten. Auch Softwarefirmen versuchen, durch Kopierschutzverfahren das unerlaubte Kopieren ihrer Software zu verhindern. Es haben sich verschiedene Methoden herausgebildet, um den unerlaubten Zugriff zu verhindern. Einige seien hier genannt:

- Verschlüsselung der Daten mit einem Codierverfahren
- Verstecken von Verzeichnissen oder Dateien (Dateiattribut)
- eigene Zugriffsverfahren zu Clustern
- Einschränkung der Zugriffsberechtigung durch eigene Kommandointerpreter
- falsche Sektorkennungen beim Formatieren
- Fehlersektoren an festgelegten Stellen
- Installationszähler (z.B. max. eine Installation)
- Verwendung einer Masterdiskette mit Zugriffsschlüssel
- Einsatz von Spezialhardware z.B. am seriellen Port.

Es gibt hier viele verschiedene Möglichkeiten. Der sicherste Schutz ist der Verschluß des Rechners, sicher nicht immer der für die Arbeit effektivste. Alle Verfahren, die den Zugriff auf Festplatten durch DOS-Standards realisieren, lassen sich durch Starten eines Systems von Diskette umgehen. Es ist entweder der Urlader auf dem ROM zu verändern oder das Laufwerk abzubauen. Einen sicheren Schutz bieten auch die auswechselbaren Festplat-

teneinsätze, die man nach der Rechnernutzung mitnehmen kann. Zunehmend werden auch Speicherkapazität und Zugriffsgeschwindigkeit von Mikrodisketten erhöht, die damit eine brauchbare Ablage von vertraulichen Informationen sichern.

Einige Sicherheit gegen Raubkopien ist durch die Verwendung von Spezialhardware gegeben; die Daten sind aber ohne Verschlüsselung von jedem lesbar. Bei auswechselbaren Festplatten dürfte hier kein Problem mehr bestehen.

Der Kopierschutz von Software läßt sich meistens auf relativ einfache Weise entfernen bzw. umgehen. Das Kopieren ganzer Disketten in der Originalstruktur (mit fehlerhaften Sektoren, Spezialsektoren u.a.) ist z.B. mit solchen Programmen wie COPYIIPC, COPYWRIT oder durch eigene Programme mit BIOS-Funktionen sehr leicht möglich. Große Softwarehäuser liefern heute vielfach ihre Software schon ohne Kopierschutz aus. Ein guter Service, der nur für registrierte Kunden geleistet wird, und niedrige Preise durch hohe Verkaufszahlen sind eine gute Alternative. Auf jeden Fall sollte man die Installationsvorschriften genau beachten.

Besondere Aufmerksamkeit ist dem Datenschutz und der Datensicherheit bei der Anwendung von Rechnernetzen zu widmen. Hier helfen meistens gutdurchdachte Zugriffsschlüssel (password), da der Rechnerzugang z.B. nur über die Schnittstellen zugelassen werden kann und ein Serverprogramm in einer Ebene oberhalb des Anwendungsprogramms die Anforderungen bedient.

Eine durch DOS bereits vorbereitete Form des Schutzes ist durch die Anwendung der AUTOEXEC-Datei gegeben. Wird dort eine spezielle Anwendungsumgebung (z.B. ein integriertes Programmpaket für die Büroautomatisierung) bereits zu Beginn geladen, so lassen sich alle sonstigen Zugriffe zu Dateien verhindern. Es ist dabei natürlich auch die Programmunterbrechung (Ctrl-C) zu sperren.

# 6. DOS-Kommandos

Im Abschn. 4. sind die drei Hauptkomponenten von MS-DOS/PC-DOS genannt worden. Nachdem die internen Funktionen des BIOS und DOS in den Abschnitten 4. und 5. behandelt worden sind, folgt nun die Beschreibung der eigentlichen Nutzerschnittstelle des PC, die durch den **Kommandointerpreter** COMMAND.COM realisiert wird. Wichtigste Aufgabe dieses Programms ist die Entgegennahme von Kommandos des Nutzers und der Aufruf entsprechender interner Funktionen bzw. spezieller Programme, die diesen Aufruf mit internen Funktionen realisieren. Das Programm COMMAND wird im Normalfall als eine Komponente von DOS beim Systemstart geladen.

In diesem Abschnitt werden nach einer Erläuterung einiger wichtiger Begriffe der Kommandosprache die Kommandos als Übersicht eingeführt und dann einzeln beschrieben. Es wird für eine bessere Übersichtlichkeit auch hier eine Einteilung in Kommandogruppen gewählt. Eine alphabetische Kommandobeschreibung in Kurzform findet man im Anhang.

## 6.1. Überblick über DOS-Kommandos

Die grundlegenden Funktionen zur Arbeit mit MS-DOS werden in Form von Programmen (Kommandos, Befehle) auf der Systemdiskette mitgeliefert.

Man unterscheidet **interne Kommandos**, die im Kommandointerpreter COMMAND.COM enthalten sind (Tafel 6.1) und **externe Kommandos**, die jeweils durch ein separates Programm realisiert sind (Tafel 6.2). Interne Kommandos werden wesentlich schneller abgearbeitet als externe Kommandos, da kein Lesen eines Programms aus der Systembibliothek erforderlich ist. Außerdem ist bei reiner Diskettenarbeit kein Gerät mit der Systemdiskette blockiert.

In der Regel existieren für jedes Programm alternative Realisierungen anderer Softwarehäuser, die z.T. wesentliche Funktionserweiterungen aufweisen. Die DOS-Kommandos sind jedoch für die normale Arbeit mit dem PC ausreichend.

Wird ein Kommando eingegeben, so analysiert der Kommandointerpreter die Eingabeparameter und aktiviert das entsprechende interne bzw. externe Programm. Es wird nicht zwischen Klein- und Großbuchstaben unterschieden. Für die normale Arbeitweise ist die Unterteilung extern-intern nicht von Bedeutung. Werden interne Kommandos jedoch aus einem Programm heraus aufgerufen, so ist COMMAND.COM mit dem entsprechenden Kommando als Parameter zu laden.

Die hier aufgeführten Kommandos werden in den nachfolgenden Abschnitten genauer behandelt. Einige interne Kommandos sind nur für die Gestaltung von Kommandofolgen in Stapeldateien sinnvoll. Diese Kommandos werden gesondert im Abschn. 7. behandelt.

Zur **Schreibweise der Kommandos** im Rahmen dieser Beschreibung gelten folgende Hinweise:

- Schlüsselwörter werden GROSS geschrieben
- Parameter werden klein geschrieben und durch aktuelle Werte ersetzt
- wahlweise Parameter stehen in eckigen Klammern []
- /Optionen werden gesondert erläutert
- zu jedem Kommando werden Schreibweise, Parameter, /Optionen und die Funktion kurz erläutert
- unterhalb der Kurzbeschreibung folgen Hinweise zur Nutzung und in einigen Fällen Beispiele
- zu jeder Funktionsgruppe wird eine Tafel zur Kommando-übersicht vorangestellt
- Parameter, die nur zur Kompatibilität mit älteren Versionen gelten, sind nicht mit aufgeführt
- weitere Informationen sind dem DOS-Handbuch zu entnehmen.

**Tafel 6.1. Interne Kommandos**

| Kommando | Funktion |
|---|---|
| BREAK | Aktivierung/Deaktivierung Ctrl-C |
| CD/CHDIR | Dateiverzeichnis einstellen |
| CLS | Bildschirm löschen |
| COPY | Datei(en) kopieren |
| CTTY | Eingabegerät wechseln |
| DATE | Datum eingeben/anzeigen |
| DEL/ERASE | Datei(en) löschen |
| DIR | Dateiverzeichnis anzeigen |
| EXIT | Rücksprung aus COMMAND |
| MD/MKDIR | neues Dateiverzeichnis anlegen |
| PATH | Suchpfad für externe Kommandos |
| PROMPT | Format für Promptzeichen |
| REN/RENAME | Datei(en) umbenennen |
| RD/RMDIR | Dateiverzeichnis löschen |
| SET | Environment ändern |
| TIME | Uhrzeit eintragen/anzeigen |
| TYPE | Dateiinhalt ausgeben |
| VER | Versionsnummer von DOS anzeigen |
| VERIFY | Überprüfung bei Schreiboperationen |
| VOL | Datenträgerkennung anzeigen/ändern |
| n: | aktuelles Laufwerk einstellen |

In den folgenden Abschnitten werden die Kommandos zu Gruppen zusammengefaßt und einzeln beschrieben. Es wird folgende Einteilung gewählt:

- **allgemeine Systemverwaltung (Abschn. 6.2.)**
- **Geräte- und Datenträgerverwaltung (Abschn. 6.3.)**
- **Verzeichnisverwaltung (Abschn. 6.4.)**
- **Dateiverwaltung (Abschn. 6.5.).**

Bei einigen Kommandos wird auf alternative Programme hingewiesen, die die gleiche Funktion haben aber manchmal etwas schneller und komfortabler zu handhaben sind. Dabei wird kein Anspruch auf Vollständigkeit erhoben.

**Tafel 6.2. externe DOS-Kommandos**

| Kommando | Funktion |
|---|---|
| APPEND | Suchpfad für Datendateien |
| ASSIGN | Umleitung von Gerätezugriffen |
| ATTRIB | Dateiattribute anzeigen/ändern |
| BACKUP | Dateien sichern |
| CHKDSK | Überprüfung eines Datenträgers |
| COMMAND | Kommandointerpreter starten |
| COMP | Dateien vergleichen |
| DEBUG | Fehlersuchprogramm |
| DISKCOMP | Datenträger vergleichen |
| DISKCOPY | Datenträger kopieren |
| EDLIN | Zeileneditor |
| EXE2BIN | Umsetzung Programmformat |
| FASTOPEN | Beschleunigung für Dateizugriff |
| FDISK | Dienstprogramm für Festplatte |
| FIND | Text in Datei suchen |
| FORMAT | Datenträger formatieren |
| GRAPHICS | Druckertreiber für Graphik |
| GRAFTABL | Graphikzeichen laden |
| JOIN | Verzeichnis wird log. Laufwerk |
| KEYB nn | spezielle Tastaturtreiber |
| LABEL | Datenträgerkennung anzeigen/ändern |
| MODE | Modus für Peripherie einstellen |
| MORE | Filter für seitenweise Ausgabe |
| NLSFUNC | Auswahl nationaler Besonderheiten |
| PRINT | Druckerspooler |
| RECOVER | Dateien regenerieren |
| REPLACE | Dateien ersetzen |
| RESTORE | gesicherte Dateien zurückspeichern |
| SHARE | Datei-Mehrfachzugriff |
| SORT | Dateien sortieren |
| SUBST | Verzeichnis durch Laufwerk ersetzen |
| SYS | Übertragung des Systems |
| TREE | Verzeichnisstruktur anzeigen |
| XCOPY | Dateien und Verzeichnisse kopieren |

## 6.2. Kommandos zur allgemeinen Systemverwaltung

Einige Parameter des Betriebssystems lassen sich durch Kommandos einstellen. In der Regel sind diese Parameter auf allgemein gebräuchliche Werte gesetzt. Für einige Anwendungen sind manchmal jedoch andere Werte besser geeignet. Die hierfür verfügbaren Kommandos sind in diesem Abschnitt zusammengefaßt.

**Tafel 6.3. Kommandos zur Systemverwaltung**

| Kommandos zur Systemverwaltung | |
|---|---|
| **VER** | Version von DOS anzeigen |
| **DATE** | Datum anzeigen/ändern |
| **TIME** | Uhrzeit anzeigen/ändern |
| **BREAK** | Aktivierung/Deaktivierung Ctrl-C |
| **VERIFY** | Überprüfung bei Schreiboperationen |
| **CTTY** | Gerät für Kommandoeingabe festlegen |
| **PROMPT** | Format für Promptzeichen festlegen |
| **SET** | Umgebungsstring (environment) ändern |
| **COMMAND** | neuen Kommandointerpreter laden |
| **EXIT** | Kommandointerpreter beenden |

**VER - Version von DOS anzeigen**

**Syntax** : VER

**Funktion** : Die Version von MS-DOS bzw. PC-DOS wird auf dem Bildschirm angezeigt

Die Version des geladenen Betriebssystems ist für die normale Arbeit nicht von Bedeutung. Werden auf einem PC mehrere Versionen genutzt, so kann man sich mit dem Kommando **VER** die aktuelle DOS-Version anzeigen lassen. Wenn das Betriebssystem beim Start keine Datei AUTOEXEC.BAT findet, so wird automatisch die Version angezeigt.

**DATE - Datum anzeigen/ändern**

**Syntax** : DATE [datum]

**Funktion** : Mit diesem Kommando kann das Datum auf einen neuen Wert geändert werden. Wird kein Wert angegeben, so erfolgt eine Anzeige des aktuellen Wertes. Der angezeigte Wert kann dann interaktiv geändert werden.

Das Betriebssystem verwaltet für jede Datei im Inhaltsverzeichnis des Datenträgers Angaben zum aktuellen Änderungsstand. Im Interesse der Datensicherung sollte man immer mit den korrekten Werten für Datum und Uhrzeit arbeiten. Zur Erleichterung wird in vielen PC eine batteriegepufferte Uhr mitgeliefert, die für die aktuelle Uhrzeit und das Datum sorgt. In diesem Fall braucht man die Werte nicht einzugeben. Das Eingabeformat des Datums ist von der landesspezifischen Einstellung des DOS abhängig.

**Beispiel:**

| | US (USA) | GR (deutsch) |
|---|---|---|
| DATE | mm-dd-yy<br>07-31-87 | tt.mm.jj<br>31.07.87 |

**TIME - Uhrzeit anzeigen/ändern**

**Syntax** : TIME [uhrzeit]

**Funktion** : Mit diesem Kommando kann die interne Uhrzeit auf einen neuen Wert eingestellt werden. Wird kein Wert angegeben, so wird die aktule Uhrzeit angezeigt, die dann interaktiv geändert werden kann.

Für die Uhrzeit gelten ähnliche Bedingungen wie für das Datum. Das Format der Zeitangabe wird im interaktiven Modus angezeigt.

**Beispiel:**

| | US (USA) | GR (deutsch) |
|---|---|---|
| TIME | hh:mm:ss<br>09:30[:00] | hh.mm.ss<br>09.30[.00] |

**BREAK - Aktivierung/Deaktivierung CTRL-C**

**Syntax** : BREAK
BREAK ON
BREAK OFF

**Optionen** :
ON Aktivierung CTRL-C
OFF Deaktivierung CTRL-C

**Funktion** : Mit diesem Kommando wird die Tastaturunterbrechung gesteuert. Wird BREAK ohne Option angegeben, so erfolgt die Anzeige des aktuellen Status. Im Status OFF wird nur bei Ein- und Ausgaben über DOS-Funktionen eine Unterbrechung berücksichtigt. Im Status ON wird nach jedem Aufruf von DOS eine Unterbrechung akzeptiert.

Weitere Informationen zur Systemfunktion BREAK findet man im Abschn. 4.4. (DOS-Interrupt 24h) und im Abschn. 4.3.7. (BIOS-Interrupt 1Bh). In der Konfigurationsdatei CONFIG.SYS ist ebenfalls die Einstellung von BREAK möglich.

**VERIFY - Überprüfung bei Schreiboperationen**

**Syntax** : VERIFY
VERIFY ON
VERIFY OFF

**Optionen** :
ON Verifizieren einschalten
OFF Verifizieren ausschalten

**Funktion** : Mit diesem Kommando wird der Verifizierstatus angezeigt (keine Option), eingeschaltet (ON) oder abgeschaltet (OFF).

Weitere Informationen zur Verifizierfunktion des DOS findet man im Abschn. 5.5.4. bei der Beschreibung der DOS-Funktionen 2Eh und 54h.

**CTTY - Gerät für Kommandoeingabe festlegen**

**Syntax** : CTTY gerät

**Optionen** :
gerät logisches Eingabegerät CON, COM1, COM2 oder AUX

**Funktion** : Mit diesem Kommando kann das Eingabegerät für den Kommandointerpreter festgelegt werden.

Standardeingabe ist das Gerät CON. Als Eingabegerät kann jeder zeichenorientierte Treiber angegeben werden, der Ein- und Ausgabe untestützt. Vom System stehen die Geräte CON (Standardkonsole), AUX bzw. COM1 (1. serielle Schnittstelle) oder COM2 (2. serielle Schnittstelle) zur Verfügung. Dieses Kommando ist z.B. sinnvoll, wenn die Bedienung von einem anderen Rechner aus erfolgen soll oder ein externes Terminal angeschlossen wird. Es ist zu beachten, daß die direkte Programmierung der Bildschirmausgabe nur auf dem Bildschirm CON (Zugriffe auf den Bildspeicher) funktioniert, da der gesamte Datenaustausch zeichenweise erfolgt. Die Umlenkung erfolgt auf DOS-Ebene.

**Beispiele:** CTTY COM1 Bedienung über COM1
CTTY CON Bedienung wieder über CON

**PROMPT - Format für Promptzeichen festlegen**

**Syntax** : PROMPT [text]

**Optionen** : (Zeichen mit spezieller Bedeutung im Text)

| | |
|---|---|
| $$ | $-Zeichen einfügen |
| $_ | <CR/LF> einfügen (neue Zeile) |
| $B | \|-Zeichen einfügen |
| $D | aktuelles Datum einfügen |
| $E | ESC-Zeichen (1Bh) einfügen |
| $G | >-Zeichen einfügen |
| $V | DOS-Version einfügen |
| $H | Rückschritt (backspace) einfügen |
| $L | <-Zeichen einfügen |
| $N | aktuelle Laufwerksbezeichnung einfügen |
| $P | aktuelles Verzeichnis einfügen |
| $Q | =-Zeichen einfügen |
| $T | aktuelle Uhrzeit einfügen |

**Funktion** : Mit diesem Kommando wird das Promptzeichen (Bereitschaftszeichen) des Kommandointerpreters festgelegt. Die unter Optionen angegebenen Zeichen fügen eine spezielle Zeichenfolge ein. Wird das Kommando ohne Text eingegeben, so erfolgt eine Standardeinstellung (PROMPT $N$G).

Die Parameterangaben können ebenfalls klein geschrieben werden. Mit diesem Kommando lassen sich auch Steuersequenzen des Bildschirms erzeugen (vgl. Konfigurierung des Bildschirmtreibers ANSI.SYS im Abschn. 4.).

**Beispiele:** PROMPT Datum = $d Uhrzeit = $t
Ausgabe: Datum = <Datum> Uhrzeit = <Uhrzeit>

PROMPT $N$G
Ausgabe: A:> (aktuelles Laufwerk A:)

**SET - Umgebungsstring (environment) ändern**

**Syntax** : SET
SET variable=
SET variable=wert

**Funktion** : Mit diesem Kommando kann der Umgebungsstring angezeigt bzw. verändert werden. Wird für eine Variable kein Wert angegeben, so wird die Variable gelöscht.

Der Umgebungsstring wird jedem aufgerufenen Programm zur Verfügung gestellt (vgl. Prozeßverwaltung im Abschn. 4.4.4.). Die Eintragungen bestehen immer aus einem Variablennamen (Großbuchstaben) und einem Wert. Beim Eintragen wird nach einem evtl. schon vorhandenen Parameter mit gleichem Namen gesucht und ggf. der Wert ersetzt. Durch das Kommando PATH wird ebenfalls ein Eintrag im Umgebungsstring vorgenommen. Der Wert wird bei SET mit angezeigt. Die Größe des Umgebungsstrings wird beim Start von COMMAND festgelegt. In Stapeldateien gelten besondere Festlegungen für die Parameterangabe (vgl. Abschn. 7.).

**Beispiele:**
```
SET COMSPEC=C:\COMMAND.COM   Suchpfad für COMMAND
SET LIB=D:\LIB               Suchpfad für LINK
SET          aktuellen Umgebungsstring anzeigen
```

**COMMAND - neuen Kommandointerpreter laden**

**Syntax** : COMMAND [d:][path] [ctty] /Optionen

**Optionen :**

| | |
|---|---|
| ctty | Standardeingabegerät (vgl. Kommando CTTY |
| /E:nnnnn | Größe des Environmentstring (128..32768) |
| /D | keine Abfrage für Datum und Uhrzeit |
| /F | bei Fehler Rückkehr ins rufende Programm |
| /P | Kommando EXIT wird nicht akzeptiert |
| /C string | Ausführung des mit string angegebenen Kommandos und automatische Beendigung |

**Funktion** : Es wird eine neue Kopie des mit COMSPEC im Environmentstring angegebenen Kommandointerpreters geladen und gestartet. Eine Beendigung ist mit dem Kommando EXIT möglich. Mit den Optionen ist der Aufruf aus einem Anwendungsprogramm heraus möglich. Dadurch können auch die im Kommandointerpreter enthaltenen internen Kommandos genutzt werden.

Durch den Start eines zusätzlichen Kommandointerpreters wird eine neue Ebene der Kommandointerpretation aufgemacht. Die Umgebungsparameter können dabei neu eingestellt werden. Der Aufruf von COMMAND erfolgt z.B. bei der Nutzung des DIR-Kommandos innerhalb eines Programms. Für diesen Aufruf wird die DOS-Funktion EXEC (4Bh) bzw. eine Funktion des verwendeten Compilers (bei Turbo-Pascal 4.0 ist das ebenfalls die Prozedur EXEC) genutzt.

**Beispiele:**
```
COMMAND /c dir b:
COMMAND c:\ COM1 /e:1024        (Bedienung von COM1)
```

| **EXIT - Kommandointerpreter beenden** |
|---|
| **Syntax** : EXIT |
| **Funktion** : Der aktuelle Kommandointerpreter wird beendet, und es erfolgt eine Rückkehr zur vorhergehenden Ebene. Wurde COMMAND mit der Option /P gestartet, erfolgt keine Reaktion. |

## 6.3. Geräte- und Datenträgerverwaltung

Die im Abschn. 2. beschriebenen Geräte werden vom Anwender als logische Geräte angesprochen. Es gelten folgende Gerätenamen (reservierte Bezeichner):

**CON:** Bedienkonsole (Tastatur und Bildschirm)
**COM1:** 1. serielle (asynchrone) Schnittstelle
**COM2:** 2. serielle (asynchrone) Schnittstelle
**LPT1:** 1. parallele Schnittstelle (Drucker)
**LPT2:** 2. parallele Schnittstelle (Drucker)
**LPT3:** 3. parallele Schnittstelle (Drucker)
**NUL:** Nullgerät (Papierkorb) für Testzwecke
**AUX:** analog COM1
**PRN:** analog LPT1.

Die logischen Gerätenamen beziehen sich auf das Gerät als Einheit und können daher nicht als Dateinamen verwendet werden. Der Doppelpunkt kann fehlen. Zusätzlich sind für Diskettenlaufwerke und Festplatten die Buchstaben A...Z als Gerätenamen festgelegt. Zur Unterscheidung zu einem Dateinamen ist der Doppelpunkt immer erforderlich.

**A:** 1. Diskettenlaufwerk
**B:** 2. Diskettenlaufwerk
**C:** 1. Festplatte

Ist nur ein Diskettenlaufwerk vorhanden, so wird für Laufwerk B: ein 2. Laufwerk auf dem gleichen Kanal wie Laufwerk A: simuliert. Ein beim Kopieren evtl. erforderlicher Diskettenwechsel wird angefordert.

Für die genannten Geräte sind Kommandos verfügbar, um eine effektive Arbeit mit DOS zu sichern. Diese Kommandos werden in den folgenden Abschnitten nach Geräten geordnet beschrieben.

### 6.3.1. Kommandos zur Tastatureinstellung

Die Funktion der Tastatur ist in den vorangegangenen Abschnitten bereits ausführlich erläutert worden (Abschnitte 2.4., 4.3.3., 4.4.1.). Auf der Kommandoebene sind einige Kommandos vorhanden, um eine Anpassung an nationale Besonderheiten vorzunehmen. In Tafel 6.4 sind die Tastaturkommandos aufgeführt.

**Tafel 6.4. Tastaturkommandos**

| Tastaturkommandos | |
|---|---|
| **KEYB** | Tastaturtreiber für nationale Tastenbelegung |
| **ANSI.SYS** | Definition von Tastaturmakros |

| **KEYB - Tastaturtreiber für nationale Tastenbelegung** | |
|---|---|
| **Syntax** | : KEYB nn |
| **Optionen** : | |
| nn | Landescode (GR deutsche Tastatur)<br>(UK englische Tastatur) |
| **Funktion** | : Mit diesem Kommando wird ein Tastaturtreiber geladen, der die nationalen Besonderheiten der verschiedenen Tastaturen berücksichtigt. |

Bei diesem Kommando ist zu beachten, daß ab Version 3.3 eine andere Behandlung der Tastaturtreiber erfolgt. Es ist nur noch ein gemeinsames Programm KEYB vorhanden. Durch die Parameterangabe <nn> wird lediglich eine spezielle Tabelle ausgewählt. Vorher war für jede Tastaturvariante ein eigenes Programm vorhanden (z.B. KEYBGR). Als Standard ist die amerikanische Tastenbelegung eingestellt. Zwischen der Standardeinstellung und der nationalen Variante kann duch <Ctrl-Alt-F1> (Standard) und <Ctrl-Alt-F2> (nationale Variante) während der Arbeit gewechselt werden. Das ist manchmal sinnvoll, wenn spezielle Zeichen auf der Tastatur nicht vereinbart sind und man sich die Doppelbelegung gemerkt hat.

| **ANSI.SYS - Definition von Tastaturmakros** | |
|---|---|
| **Syntax** | : Nur als Treiber in der Datei CONFIG.SYS |
| **Funktion** | : Wenn dieser Treiber installiert ist, kann man durch Angabe von ESC-Sequenzen Tasten mit einer neuen Funktion belegen. |

Die Einbindung des ANSI-Treibers ist im Abschn. 4.2. beschrieben. Werden ESC-Sequenzen zum Bildschirm gesendet, so analysiert ANSI.SYS die Zeichenfolgen und definiert die Belegung von Tasten entsprechend der ESC-Sequenz neu. Dazu wird die DOS-Funktion zur Tastatureingabe modifiziert. Die Tastaturmakros funktionieren daher auch nur, wenn die Eingabe mit DOS-Funktionen erfolgt. Die ESC-Sequenzen des ANSI-Treibers sind im Anhang aufgeführt. Es ist zu beachten, daß der erweiterte Scancode als ASCII-Zeichen anzugeben ist. Die Ausgabe der ESC-Sequenzen ist entweder von einem Programm, durch das Promptzeichen, ein ECHO in einer Stapeldatei oder relativ umständlich direkt über die Tastatur möglich. Der ursprüngliche Wert einer Taste wird wiederhergestellt, wenn zweimal der gleiche Wert angegeben wird.

```
Beispiele: (Format:  ESC[<Scancode>;"<string>"p)
Taste F1 als DIR     write(27'[0059;"dir"p');   (Pascal)
Taste a  als b       write(27'[30;"b"p');
```

### 6.3.2. Kommandos zur Bildschirmeinstellung

Es gibt eine Vielzahl von Bildschirmparametern, die z.T. auch durch Kommandos eingestellt werden können (Tafel 6.5).

**Tafel 6.5. Kommandos zur Bildschirmeinstellung**

| Kommandos zur Bildschirmeinstellung | |
|---|---|
| **MODE** | Bildschirmmodus einstellen |
| **CLS** | Bildschirm löschen |
| **GRAFTABL** | Tabelle für Graphikzeichen |
| **ANSI.SYS** | ANSI-Bildschirmtreiber |

Die Einstellung erfolgt entweder durch Aufnahme der Kommandos in die Datei AUTOEXEC.BAT oder speziell für ein Programm. Beim Experimentieren kann es vorkommen, daß der Bildschirm wegen falscher Einstellung nichts mehr anzeigt. Durch Eingabe der entsprechenden Kommandos kann in der Regel die Einstellung wieder korrigiert werden, ohne daß der Rechner neu gestartet wird.

**MODE - Bildschirmmodus einstellen**

**Syntax** : MODE [display],shift[,T][scroll]

**Optionen** :

| | |
|---|---|
| display | MONO<br>40\|80\|BW40\|BW80\|CO40\|CO80 |
| shift | R\|L (Verschiebung rechts bzw. links) |
| T | Testmuster für Verschiebung (Abfrage Y/N) |

**Funktion** : Mit diesem Kommando wird der Bildschirm auf einen gewünschten Modus eingestellt.

Die Bildschirmmodi sind bereits im Abschn. 4. beschrieben worden. Für Herculesadapter ist bei einigen Versionen von MS-DOS eine Einstellung der Seitenanzahl durch Angabe von MODE HGC,HALF (eine Seite) bzw. MODE HGC,FULL (zwei Seiten) möglich. Bei der Installation von Monochromadapter und Color/Graphik-Adapter in einem Gerät ist durch das MODE-Kommando ein Umschalten zwischen diesen beiden Adaptern möglich.

**CLS - Bildschirm löschen**

**Syntax** : CLS

**Funktion** : Der Bildschirm wird mit der Hintergrundfarbe gelöscht und das Promptzeichen in der linken oberen Ecke ausgegeben.

**GRAFTABL - Tabelle für Graphikzeichen laden**

**Syntax** : GRAFTABL [nnn]
GRAFTABL ?
GRAFTABL /STA

**Optionen** :

| | |
|---|---|
| nnn | 437=USA, 850=mehrsprachig, u.a. |
| ? | Liste der Optionen, aktueller Status |
| /STA | Anzeige des aktiven Zeichensatzes |

**Funktion** : Mit diesem Kommando wird die Tabelle für die Zeichen des erweiterten ASCII-Codes geladen (Graphikzeichen 128...255)

Die mit GRAFTABL bereitgestellte Tabelle realisiert die im Anhang angegebenen Graphikzeichen auf dem Bildschirm im Graphikmodus. Die Standardtabelle enthält nur die ersten 128 Zeichen des

ASCII-Codes. Die Tabelle wird nur einmal geladen,;beim wiederholten Aufruf von GRAFTABL wird ein Hinweis ausgegeben, daß die erweiterte Zeichenroutine bereits aktiv ist. Die Optionen sind erst ab Version 3.3 gültig.

| **ANSI.SYS - ANSI-Bildschirmtreiber** |
|---|
| **Syntax** : nur als Treiber in der Datei CONFIG.SYS |
| **Funktion** : Wenn dieser Treiber installiert ist, kann man durch Ausgabe von ESC-Sequenzen einige Funktionen zur Bildschirmsteuerung aufrufen. |

Die Einbindung des Treibers ist im Abschn. 4.2. beschrieben. Werden ESC-Sequenzen zum Bildschirm (Standardausgabegerät) gesendet, so analysiert ANSI.SYS die Zeichenfolgen und ruft Funktionen zur Cursorsteuerung, zum Löschen oder zur Moduseinstellung auf. Die Steuerung über ESC-Sequenzen funktioniert nur, wenn zur Ausgabe DOS-Funktionen benutzt werden. Eine Tabelle mit zulässigen ESC-Sequenzen ist im Anhang enthalten. Im Abschn. 11. sind Beispiele zur Cursorsteuerung mit Turbo-Pascal enthalten. ANSI.SYS ist sowohl für die Bildschirmsteuerung als auch für die Tastatureingabe (Abschn. 6.3.2.) verwendbar. Durch die Nutzung von ANSI.SYS wird die Bildschirmausgabe merklich langsamer, da alle ausgegebenen Zeichen gefiltert werden. Die Bildschirmausgabe wird zwar an die Möglichkeiten anderer Terminaltypen angepaßt und dadurch die Portabilität der Software verbessert, eine effektive Windowtechnik ist so aber z.B. auf einem "langsamen" XT nicht realisierbar.

**Beispiele:**
```
ESC[J ESC[10;30f Ausschrift
(Löschen des Bildschirms und Auschrift in
Zeile 10, Spalte 30)

PROMPT $e[J Kommando:
(Löschen des Bildschirms, Ausgabe von Kommando:
in linker oberer Ecke bei jeder Kommandoeingabe)
```

### 6.3.3. Kommandos zur Einstellung der COM-Schnittstelle

Die beiden seriellen Schnittstellen sind nach dem Systemstart mit Standardparameterwerten eingestellt. Die Verwendung als Gerät unter DOS ist nur wenig unterstützt. Es existiert lediglich ein Kommando MODE zur Neufestlegung der für eine asynchrone Datenübertragung notwendigen Parameter (vgl. Abschn. 5.5.1.). Zur Nutzung der Schnittstellen im Netzbetrieb sind weitere Kommandos

erforderlich, die eine sichere Datenübertragung gestatten.

| **MODE - Parameter für serielle Schnittstelle einstellen** | |
|---|---|
| **Syntax** | : MODE port:[baudrate],[parity],[data],[stop] |
| **Optionen** : | |
| port | COM1, COM2 (serielles Port) |
| baudrate | 110, 150, 300, 600, 1200, 2400, 4800, 9600 |
| parity | e (even), o (odd), n (none) (Parität) |
| data | 7, 8 (Datenbit) |
| stop | 1, 2 (Stopbit) |
| **Funktion** | : Mit diesem Kommando kann die Parametereinstellung für eine der seriellen Schnittstellen COM1 oder COM2 durchgeführt werden. |

Um eine Übertragung über die serielle Schnittstelle zu realisieren, müssen die Parameter auf beiden Seiten der Übertragungsstrecke identisch eingestellt sein. Es ist sinnvoll, die Schnittstelle immer auf die Werte einzustellen, mit denen die Übertragung stattfinden soll, und sich nicht auf Standardeinstellungen zu verlassen. Es besteht keine Möglichkeit, die eingestellten Werte abzufragen. Als Standardeinstellung werden folgende Werte angenommen:

gerade Parität (even parity)
Datenwortlänge 7 Bit
2 Stopbit.

### 6.3.4. Kommandos zur Druckereinstellung

Der Standarddrucker (logisches Gerät PRN:) wird im Normalfall an die parallele Schnittstelle LPT1: (Centronics-Port) angeschlossen. Diese Zuordnung kann mit dem Kommando MODE verändert werden. Es ist außerdem möglich, die Zeichenzahl in einer Zeile und die Anzahl Zeilen auf einer Seite einzustellen.

Im BIOS ist eine Funktion zur direkten Ausgabe (hardcopy) des Bildschirminhalts definiert, die über die Tastenkombination <Shift-PrtSc> aufgerufen wird. Die Ausgabe arbeitet standardmäßig nur im Textmodus. Für den Graphikmodus muß ein Treiberprogramm geladen werden, das den Standardtreiber für den entsprechenden BIOS-Interrupt (05h) ersetzt. In Tafel 6.6 sind die für den Drucker gültigen Kommandos zusammengefaßt. Das Kommando PRINT installiert einen Druckerspooler, der einen eingeschränkten Pa-

rallelbetrieb von Drucken und Arbeit am Bildschirm ermöglicht.

**Tafel 6.6. Kommandos zur Druckereinstellung**

| Kommandos zur Druckereinstellung | |
|---|---|
| **MODE** | Druckerparameter einstellen |
| **GRAPHICS** | Druckertreiber für Bildschirmausdruck |
| **PRINT** | Druckerspooler |

**MODE - Druckerparameter einstellen**

**Syntax** : MODE LPT#:[=COM#]
MODE LPT#:[zeichen],[zeilen],[p]

**Optionen** :

| | |
|---|---|
| LPT# | LPT1, LPT2, LPT3 (physischer Drucker) |
| COM# | COM1, COM2 (serielles Port für Umlenkung) |
| zeichen | 80, 132 (Anzahl Zeichen in einer Zeile) |
| zeilen | 6, 8 (Anzahl Zeilen je Zoll) |
| p | Wiederholung, wenn Drucker nicht bereit |

**Funktion** : Mit der ersten Form des MODE-Kommandos wird die Druckausgabe auf eine serielle Schnittstelle umgelenkt. Die zweite Form legt Parameter zur Seitengestaltung fest. Werden keine Parameter angegeben, so wird auf die Standardwerte (80 Zeichen, 6 Zeilen je Zoll) eingestellt.

Bei der Umlenkung auf ein serielles Port ist vorher die Schnittstelle mit den erforderlichen Parametern einzustellen (vgl. Abschn. 6.3.3.). Wird in der ersten Form nur der Drucker angegeben, so wird die Umlenkung zurückgenommen. Mit dem Parameter p kann verhindert werden, daß eine Druckausgabe zum Programmabbruch führt, wenn der Drucker nicht bereit ist (z.B. Papierende, nicht eingeschaltet). Die Umlenkung des Druckers ist natürlich nur sinnvoll, wenn auch am seriellen Port ein Drucker angeschlossen ist. Die Druckerparameter für den seriellen Anschluß am Drucker müssen mit den Parametern im MODE-Kommando übereinstimmen.

```
Beispiele: MODE COM1:9600,e,8,1      (Einstellung COM1)
           MODE LPT1:=COM1:          (Umlenkung nach COM1)
           MODE LPT2:132             (Schmaldruck auf LPT2)
           MODE LPT1:,8              (8 Zeilen/Zoll auf LPT1)
```

**GRAPHICS - Druckertreiber für Bildschirmausdruck**

**Syntax** : GRAPHICS [drucker] /Optionen

**Optionen** :

| | |
|---|---|
| drucker | COLOR1, COLOR4, COLOR8, COMPACT, GRAPHICS |
| /R | Druck Schwarz auf Weiß |
| /B | Untergrund in Farbe (nur bei COLOR4, COLOR8) |

**Funktion** : Laden des Druckertreibers für Bildschirmausdruck (hardcopy) im Graphikmodus

Der Ausdruck wird nur beeinflußt, wenn der Bildschirm im Graphikmodus ist. Im hochauflösenden Modus (640*200) wird der Bildschirminhalt im Querformat gedruckt. Für weitere Druckertypen werden bei einigen Herstellern spezielle Druckertreiber mitgeliefert.

**PRINT - Druckerspooler** (1. Aufruf von PRINT)

**Syntax** : PRINT /Optionen

**Optionen** :

| | |
|---|---|
| /D:gerät | LPT1, LPT2, LPT3, COM1, COM2, AUX, PRN Ausgabegerät, Standard PRN |
| /B:n | Puffergröße, Standard n=512 |
| /Q:n | Anzahl Dateien in Warteschlange (1...32) Standard n=10 |
| /M:n | Anzahl Zeiteinheiten (clock ticks, 1...255) für PRINT, Standard n=2 |
| /S:n | Zeitscheibe (time slice value, 1...255) für PRINT, Standard n=8 |
| /U:n | Wartezeit bis Drucker bereit (1..255), Standard n=1 |

**Funktion** : Mit PRINT wird ein Druckerspooler installiert, der weitere Ausgaben mit PRINT bearbeitet.

Das Kommando PRINT wird für zwei Aufgaben eingesetzt. Beim 1. Aufruf wird der Druckerspooler installiert und bleibt speicherresident bis zum Neustart. Der Aufruf sollte daher möglichst gleich nach dem Start erfolgen (z.B. in der Datei AUTOEXEC.BAT), um eine Zersplitterung des Speichers zu vermeiden. Weitere Aufrufe von PRINT werden zur Ausgabe von Dateien und für die Drucksteuerung genutzt (vgl. Abschn. 6.5.).

**Beispiel:** PRINT /B:1024/D:LPT2/M:20/Q:32/U:10

Es existieren alternative Druckerspooler, die z.B. verschiedene Übertragungsprotokolle für serielle Drucker und auch den Erweiterungsspeicher (extended memory) des AT nutzen können.

### 6.3.5. Verwaltung von Disketten und Festplatten

Disketten und Festplatten werden durch blockorientierte Gerätetreiber verwaltet (Verwendung von Buchstaben A...Z als Gerätebezeichner). Bevor diese Datenträger als externe Speichermedien eingesetzt werden können, sind sie für DOS einzurichten. In den vorrangehenden Abschnitten waren bereits einige Informationen zur Struktur von Disketten und Festplatten enthalten. In diesem Abschnitt werden die Kommandos beschrieben, um von der Ebene des Kommandointerpreters aus die nötigen Arbeiten ausführen zu können (Tafel 6.7).

**Tafel 6.7. Kommandos zur Datenträgerverwaltung**

| Datenträgerverwaltung | |
|---|---|
| **d:** | Auswahl des aktuellen Gerätes (d=A...Z) |
| **ASSIGN** | Ersetzen von Laufwerksnamen |
| **FDISK** | Dienstprogramm für Festplatten |
| **FORMAT** | Formatierung von Datenträgern |
| **LABEL** | Eintragen eines Datenträgeretiketts |
| **VOL** | Datenträgeretikett anzeigen |
| **SYS** | DOS-Systemdateien kopieren |
| **CHKDSK** | Datenträger überprüfen |
| **RECOVER** | Behandlung von Sektorfehlern |
| **DISKCOPY** | Datenträger kopieren |
| **DISKCOMP** | Datenträger vergleichen |
| **BACKUP** | Sicherheitskopie erstellen |
| **RESTORE** | Sicherheitskopie zurückspeichern |

Der Kommandointerpreter kennt ein **aktuelles Laufwerk**, das für die Arbeit mit Datenträgern verwendet wird, wenn keine Angabe im Kommando erfolgt. Das aktuelle Laufwerk wird normalerweise im Promptzeichen angegeben. Soll das aktuelle Laufwerk geändert werden, so ist das neue aktuelle Laufwerk, gefolgt von einem Doppelpunkt, einzugeben.

```
Beispiel: A:<cr>  aktuelles Laufwerk Diskettenlaufwerk A
          C:<cr>  aktuelles Laufwerk Festplatte C
```

Als Laufwerknamen sind nur die Buchstaben A...Z zulässig. Die Anzahl der Laufwerke wird durch den Parameter LASTDRIVE in der Datei CONFIG.SYS festgelegt (vgl. Abschn. 4.).

| **ASSIGN - Ersetzen von Laufwerksnamen** |
|---|
| **Syntax** : ASSIGN [dalt=dneu]<br><br>**Optionen** :<br>dalt Laufwerksname, der ersetzt werden soll<br>dneu neuer Laufwerksname<br><br>**Funktion** : Der alte Laufwerksname wird auf den neuen Laufwerksnamen umgesetzt. Die Umsetzung ist für den Anwender transparent. Es können mehrere Ersetzungen in einem Kommando vorgenommen werden. Wird ein ASSIGN ohne Parameter angegeben, so werden die ursprünglichen Werte eingestellt. |

Das ASSIGN-Kommando ist manchmal sinnvoll, wenn eine Anwendung bestimmte Laufwerksbezeichnungen fest voraussetzt, die jedoch an einem anderen Gerät nicht vorhanden sind. Durch ASSIGN können die gewünschten Werte eingestellt werden.

**Beispiele:** ASSIGN A=C (alle Bezugnamen auf A: werden auf C: umgelenkt)

ASSIGN A=C B=D (mehrere Umlenkunkungen)

ASSIGN (Standardwerte einstellen)

| **FDISK - Dienstprogramm für Festplatten** |
|---|
| **Syntax** : FDISK<br><br>**Optionen** : alle Eingaben menügesteuert<br><br>**Funktion** : Eine physisch formatierte Festplatte wird in mehrere Bereiche (partitions) eingeteilt, die eingerichtet, gelöscht, aktiviert und deaktiviert werden können. In neueren Versionen ist auch die Zuweisung von Treibern zu Bereichen möglich. |

Nach dem Start meldet sich FDISK mit einem Menü, aus dem einzelnen Funktionen ausgewählt werden können. Der Funktionsumfang ist in den einzelnen DOS-Versionen unterschiedlich und wird z.T. durch Unterschiede im Leistungsvermögen der Festplatten verschiedener Hersteller begründet. Einige Verwaltungsprogramme ersetzen FDISK mit einem wesentlich umfangreicheren Funktionsangebot (z.B. SpeedStor von Storage Dimensions, das sowohl die

physische Formatierung, die Bereichseinteilung als auch die Bereitstellung spezieller Treiberprogramme für eine große Anzahl von Festplattentypen realisiert). Die Bedienung dieser zumeist menügesteuerten Programme ist relativ einfach. Man sollte sich nur im klaren darüber sein, daß alle Veränderungen auf der Festplatte zum Verlust von Daten führen können. Man sollte daher diese Programme nur zur Einrichtung auf dem Systemgerät belassen bzw. deren Verwendung und den Benutzerkreis stark einschränken. Einige Funktionen, die grundlegend für die Verwaltung sind, sollen hier aufgezählt werden.

**Funktion 1: DOS-Partition anlegen**
Es wird ein neuer Bereich im freien Speicherbereich angelegt. Der mögliche Umfang wird angezeigt. Standardmäßig können Bereiche nur bis zu einer Größe von 32 MByte angelegt werden. Größere Platten werden in mehrere logische Platten eingeteilt, die einzeln verwaltet werden.

**Funktion 2: aktive Partition wechseln**
Wenn das System von der Festplatte geladen werden soll, so sucht der Bootstraplader nach einer aktiven DOS-Partition. Eine der Partitionen muß als aktive Partition gekennzeichnet werden und das Betriebssystem enthalten.

**Funktion 3: DOS-Partition löschen**
Ein nicht mehr benötigter Bereich kann auf der Platte gelöscht und z.B. für ein anderes Betriebssystem genutzt werden. Die Daten in der gelöschten Partition sind nicht mehr verfügbar.

**Funktion 4: Partition anzeigen**
Mit dieser Funktion kann man sich ein Verzeichnis der bestehenden Bereiche ausgeben lassen. Es werden die Anfangs- und Endadressen und die Betriebssystemkennung ausgegeben.

**Funktion 5: nächste Festplatte anwählen**
Sind im System mehrere Festplatten installiert (die Information darüber werden vom Initialisierungsprogramm nach dem Systemstart ermittelt), so ist die Bearbeitung der nächsten Platte möglich.

Weitere Informationen zur Festplattenverwaltung sind den entsprechenden Dokumentationsunterlagen zu entnehmen. Nach der Einteilung in Partitionen werden Disketten und Festplattenbereiche mit den gleichen Kommandos (FORMAT, SYS usw.) behandelt. Die Ansteuerung erfolgt durch die zughörigen blockorientierten Treiberprogramme.

Einige Spezialprogramme (z.B. SpeedStor) enthalten alle Funktionen, die zur Festplattenverwaltung erforderlich sind. Dabei sind auch solche Funktionen realisiert, die ein Überschreiten der 32-MByte-Grenze zulassen. Bei der Anwendung sollte man auf evtl. vorhandene Unverträglichkeit mit der normalen Verwaltung durch FDSIK achten. Programme wie FDISK bzw. SpeedStor sollten nur dem Systemmanager zugänglich sein.

| **FORMAT - Formatierung von Datenträgern** | |
|---|---|
| **Syntax** | : FORMAT d: /Optionen |
| **Optionen** : | |
| d: | Laufwerk, das formatiert werden soll |
| /V | Datenträgeretikett eintragen |
| /4 | Format 360 KByte (Laufwerke mit 1.2 MByte) |
| /T:n | Anzahl Spuren auf dem Datenträger |
| /N:n | Anzahl Sektoren in einer Spur |
| /S | System übertragen |
| **Funktion** : | Der im angegebenen Laufwerk enthaltene Datenträger wird neu formatiert. Vor der Formatierung wird ein neuer Datenträger angefordert. Damit läßt sich ein versehentliches Formatieren abbrechen. |

Das FORMAT-Kommando erzeugt eine neue Sektoreinteilung auf Disketten, legt eine FAT (file allocation table) auf dem Datenträger bzw. der Partition an und erzeugt ein Inhaltsverzeichnis (directory). Die Optionen zur Sektor- und Spuranzahl (/S /N) sind nicht bei allen Versionen von MS-DOS gültig. Die Formatierung von Disketten mit 360 KByte auf HD-Laufwerken mit 1.2 MByte ist zwar möglich, das Lesen auf den entsprechenden Geräten mit 360 KByte verläuft jedoch nicht immer problemlos. Wird die Option /V angegeben, so wird ein Datenträgeretikett angefordert, das aus max. 11 Zeichen bestehen kann. Das Format entspricht den Angaben beim Kommando LABEL. Die Systemübertragung (Option /S) kann einzeln auch mit dem Kommando SYS durchgeführt werden. Nach Abschluß der Formatierung wird der Status des Datenträgers (belegter und freier Speicherplatz) und die Bereitschaft zur Formatierung eines weiteren Datenträgers bzw. Programmbeendigung ausgegeben.

**Beispiel:** FORMAT c:/S (Festplatte C: formatieren und Systemdateien übertragen)

| **LABEL - Datenträgeretikett eintragen** | |
|---|---|
| **Syntax** | : LABEL [d:][label] |
| **Optionen** : | |
| d: | Laufwerk des Datenträgers |
| label | Datenträgeretikett (max. 11 Zeichen) |
| **Funktion** : | Das mit der Option label angegebene Etikett wird im Wurzelverzeichnis des Datenträgers in Laufwerk d: eingetragen. Wird <label> nicht angegeben, so erfolgt eine Abfrage. |

Im Datenträgeretikett dürfen keine Sonderzeichen vorkommen. Die maximale Länge ist 11 Zeichen. Das Datenträgeretikett wird bei der Verzeichnisausgabe und mit dem Kommando VOL angezeigt. Wird bei der Abfrage keine Zeichenfolge angegeben, so wird das alte Etikett gelöscht.

**Beispiel:** LABEL D:HUEBENER

| **VOL - Datenträgeretikett anzeigen** |
|---|
| **Syntax** : VOL [d:] |
| **Optionen** :<br>d: Laufwerk (Standard: aktuelles Laufwerk) |
| **Funktion** : Das Datenträgeretikett des angegebenen Laufwerks wird angezeigt. |

Das Datenträgeretikett wird mit dem Kommando FORMAT oder LABEL eingetragen. Bei fehlender Laufwerksangabe wird das aktuelle Laufwerk verwendet.

| **SYS - Systemdateien übertragen** |
|---|
| **Syntax** : SYS d: |
| **Optionen** :<br>d: Laufwerk mit Systemdatenträger (Ziel) |
| **Funktion** : Die Systemdateien BIOS (IO.SYS bzw. IBMBIO) und DOS (MSDOS bzw. IBMDOS) werden auf das angegeben Laufwerk kopiert. |

Die Systemdateien für BIOS und DOS müssen am Anfang des Datenbereichs gespeichert sein. Ein Kopieren mit COPY ist nicht ohne besondere Vorkehrungen möglich, da die Dateien mit dem Attribut <System> gekennzeichnet sind und nicht kopiert werden. Sind weitere Dateien auf dem Datenträger vorhanden, so muß der Platz für die Systemdateien frei bzw. durch eine andere Kopie des Systems reserviert sein. Das Ersetzen einer älteren Version ist nur möglich, wenn die neue Version nicht mehr Speicherplatz erfordert (und das kommt selten vor). Die Übertragung eines Systems ist ebenfalls im Zusammenhang mit dem Formatieren möglich (Kommando FORMAT mit Option /S). Es ist zu beachten, daß der Kommandointerpreter COMMAND.COM nicht durch SYS kopiert wird.

**CHKDSK - Datenträger überprüfen**

**Syntax** : CHKDSK [d:] [datei] /Optionen

**Optionen** :

| | |
|---|---|
| d: | Laufwerk, das überprüft werden soll |
| datei | Name der zu überprüfenden Datei(en) |
| /F | Fehlerkorrektur (fix errors) |
| /V | Ablaufprotokoll (verbose) |

**Funktion** : Das Laufwerk d: wird auf Konsistenz der Dateien mit dem Inhaltsverzeichnis überprüft. Fehler werden bei /F automatisch korrigiert. Bei /V wird jede kontrollierte Datei in der Standardausgabe angezeigt.

Es kommt mitunter vor, daß Dateien angelegt werden, jedoch nicht im Inhaltsverzeichnis eingetragen sind. Durch CHKDSK werden über die FAT alle Dateien überprüft, und bei Fehlern wird unter einem von CHKDSK vergebenen Namen ein Verzeichniseintrag gebildet (FILEnnnn.CHK). Man kann diese Dateien dann kontrollieren und evtl. umbenennen bzw. löschen. Da die Ausgabe des Ablaufprotokolls auf das Standardausgabegerät erfolgt, ist die Umlenkung in eine Datei möglich. Am Ende des Programmlaufs wird ein Datenträgerstatus ausgegeben. Wird im Kommando eine Datei bzw. Dateigruppe (* oder ?) angegeben, so wird außerdem geprüft, ob der Speicherplatz für die Dateien zusammenhängend ist.

**RECOVER - Behandlung von Sektorfehlern**

**Syntax** : RECOVER [d:][datei]

**Optionen** :

| | |
|---|---|
| d: | Laufwerk |
| datei | Name der zu korrigierenden Datei(en) |

**Funktion** : Der Datenträger im angegebenen Laufwerk bzw. die angegebene Datei oder Dateigruppe werden sektorweise gelesen. Fehlerhafte Sektoren werden markiert und nicht mit in die neue Datei übertragen.

Die korrigierten Dateien sind unvollständig, lassen sich aber evtl. regenerieren, indem fehlende Stellen eingefügt werden. Die durch RECOVER angelegten Dateien werden mit dem Namen FILEnnnn.REC ins Wurzelverzeichnis eingetragen.

| **DISKCOPY - Datenträger kopieren** | |
|---|---|
| **Syntax** | : DISKCOPY [d1:] [d2:] /Optionen |
| **Optionen** : | |
| d1: | Quelldatenträger |
| d2: | Zieldatenträger |
| /1 | nur eine Seite kopieren |
| **Funktion** | : Der Datenträger in Laufwerk d1: wird spurweise nach Laufwerk d2: kopiert. Bei Angabe von /1 wird nur eine Diskettenseite kopiert. |

DISKCOPY realisiert zusätzlich zum Kopieren auch ein Formatieren, wenn der Zieldatenträger neu ist bzw. für ein anderes Format eingerichtet ist. Die Laufwerkstypen müssen übereinstimmen. Wird kein Laufwerk angegeben, so erfolgt eine Abfrage. Wird nur ein Laufwerk angegeben, so wird die Quelldiskette gelesen und im gleichen Laufwerk die Zieldiskette geschrieben. DISKCOPY fordert den Wechsel der Datenträger an. Mit einem Programmlauf können mehrere Kopiervorgänge durchführt werden.

| **DISKCOMP - Datenträger vergleichen** | |
|---|---|
| **Syntax** | : DISKCOMP [d1:] [d2:] /Optionen |
| **Optionen** : | |
| d1: | Laufwerk 1 |
| d2: | Laufwerk 2 |
| /1 | nur eine Seite vergleichen |
| /8 | nur 8 Sektoren je Spur vergleichen |
| **Funktion** | : Die Datenträger in Laufwerk d1: und d2: werden spurweise verglichen. Bei Abweichun- erfolgt eine Fehlermeldung. |

Wie beim Kommando DISKCOPY müssen die Datenträgertypen übereinstimmen. Wird nur ein Laufwerk angegeben, so werden beide Datenträger im gleichen Laufwerk gelesen. Der Datenträgerwechsel wird vom Programm angefordert. Fehlt die Laufwerksangabe, so erfolgt eine Abfrage. Mit einem Programmlauf können mehrere Datenträger verglichen werden. Sollte bei der Arbeit mit zwei Diskettenlaufwerken die Systemdiskette entfernt worden sein, so wird nach dem Programmlauf der Diskettenwechsel angefordert. Einzelne Dateien werden mit dem Kommando COMP verglichen.

**BACKUP - Sicherheitskopie erstellen**

**Syntax** : BACKUP [d1:][datei] [d2:] /Optionen

**Optionen** :

| | |
|---|---|
| d1: | Quellaufwerk |
| datei | Dateien (Pfad, Dateiname) |
| d2: | Ziellaufwerk |
| /A | Hinzufügen von Dateien (append) |
| /M | nur modifizierte Dateien sichern (modified) |
| /S | Dateien aus Unterverzeichnissen sichern (subdirectories) |
| /F | Formatierung des Zieldatenträgers |
| /D:datum | Sichern ab <datum> (mm-tt-jj) |
| /T:zeit | Sichern ab <zeit> (hh:mm:ss) |
| /L:log | Anlegen einer Datei mit dem Namen <log> |

**Funktion** : Erstellen einer Sicherheitskopie von Laufwerk d1: auf Diskette oder Festplatte d2: im speziellen BACKUP-Format. Auswahl von Dateien mit Namen oder Optionen ist möglich.

Mit BACKUP ist das Sichern von Datenträgern oder ausgewählten Dateien eines Datenträgers möglich. Zusammen mit dem Kommando RESTORE ist eine relativ schnelle Sicherung und Wiederherstellung realisierbar. BACKUP verwendet ein eigenes Dateiformat. Dadurch ist auch die Sicherung sehr groβer Dateien z.B. auf mehreren Disketten möglich. Bei Angabe der Option /L wird eine Verzeichnisdatei erstellt, die eine schnelle Orientierung über den Sicherungslauf gestattet.

Es existieren Programme, die die gleichen Funktionen wie BACKUP/RESTORE anbieten, z.T. jedoch wesentlich schneller arbeiten und eine leichtere Bedienung gestatten (z.B. FASTBACK, SAFEGUARD, TURBO-BACKUP, PC-BACKUP). Manchmal ist es angebracht, sich eine komprimierte Archivdatei von Softwarepaketen anzulegen und nur die Arbeitsversionen von der Festplatte mit BACKUP zu sichern. Neuere Diskettenformate machen es möglich, auch gröβere Dateien im DOS-Format auf Diskette zu archivieren. Da die Kopiergeschwindigkeit ebenfalls gröβer geworden ist, ist das eine sinnvolle Alternative zum Sichern mit BACKUP.

**Beispiele:** BACKUP text.doc b: /a
(Sichern der Datei text.doc auf B:, die Sicherheitskopie wird zu den bestehenden Dateien hinzugefügt)

BACKUP c: a:*.* /s/m
(Sichern aller modifizierten Dateien von c: nach a:)

**RESTORE - Sicherheitskopien zurückspeichern**

**Syntax** : RESTORE [d1:] [d2:][datei] /Optionen

**Optionen** :

| | |
|---|---|
| d1: | Quellaufwerk (Sicherungskopien) |
| d2: | Ziellaufwerk |
| datei | Dateien, die wiederhergestellt werden sollen |
| /P | Bestätigung bei Rückspeichern (prompt) |
| /S | Unterverzeichnisse wiederherstellen |
| /B:datum | nur Dateien vor <datum> (before) |
| /A:datum | nur Dateien nach <datum> (after) |
| /E:zeit | nur Dateien vor <zeit> (earlier) |
| /L:zeit | nur Dateien nach <zeit> (later) |
| /M | nur modifizierte Dateien |
| /N | nur Dateien die fehlen (non existent) |

**Funktion** : Die auf dem Laufwerk d1: gespeicherten Sicherheitskopien werden auf das Laufwerk d2: zurückgespeichert. Mit Dateinamen und Optionen ist eine Auswahl möglich.

Die Funktionen sind komplementär zu BACKUP. Es kann nur das Dateiformat von BACKUP verwendet werden. Die Systemdateien des BIOS und DOS können nicht mit RESTORE zurückgespeichert werden (vgl. Kommando SYS).

**Beispiel:** RESTORE a: c:*.pas /m/s
(Zurückspeichern aller modifizierten Pascal-Dateien von a: nach c:, vorm Speichern wird eine Bestätigung abgefragt)

## 6.4. Kommandos zur Verzeichnisverwaltung

Das leistungsfähige Dateisystem von MS-DOS verwendet eine baumartige Verzeichnisstruktur, die durch den Kommandointerpreter mit einigen Kommandos zum Einrichten, Löschen, Wechseln, Ändern und Anzeigen unterstützt wird. Die verfügbaren Kommandos sind in Tafel 6.8 zusammengefaßt und werden anschließend einzeln beschrieben. Informationen zur Baumstruktur sind im Abschn. 5. enthalten. Der **Zugriffsweg** vom Gerät bis zur Datei über verschiedene Indexstufen heißt **Pfad** (path). Der Übergang von einer Indexstufe zur nächsten wird durch das Zeichen '\' gekennzeichnet. Der Pfadname darf eine Gesamtlänge von 63 Zeichen haben. Bei der Formatierung wird die erste Indexstufe angelegt, die als **Wurzelverzeichnis** (root directory) bezeichnet und mit d:\ gekennzeichnet wird. Im Verzeichnis sind Einträge enthalten, die entweder auf Dateien oder Unterverzeichnisse verweisen. Der Verzeichniseintrag, der auf das übergeordnete **Elternverzeichnis** (parent directory) verweist, wird durch zwei Punkte (..) gekennzeichnet. Der Kommandointerpreter kennt ein **aktuelles Verzeichnis**, das immer dann verwendet wird, wenn keine Angabe in einem Kommando enthalten ist.

**Beispiele:**
```
D:\                     (Wurzelverzeichnis auf D:)
C:brief.doc             (Datei im aktuellen Verzeichnis)
D:\Index1\Index2\Datei.ext
                        (zwei Indexstufen auf D:)
```

**Tafel 6.8. Kommandos zur Verzeichnisverwaltung**

| Verzeichnisverwaltung | |
|---|---|
| **ATTRIB** | Dateiattribute im Verzeichnis ändern |
| **MKDIR** (**MD**) | Unterverzeichnis anlegen |
| **CHDIR** (**CD**) | aktuelles Verzeichnis einstellen |
| **RMDIR** (**RD**) | Unterverzeichnis entfernen |
| **DIR** | Inhaltsverzeichnis ausgeben |
| **TREE** | Verzeichnisstruktur ausgeben |
| **JOIN** | Laufwerk in Verzeichnisstruktur eingliedern |
| **SUBST** | Verzeichnis als Laufwerk vereinbaren |
| **APPEND** | Suchpfad für Datendateien festlegen |
| **PATH** | Suchpfad für Programme festlegen |

**ATTRIB - Dateiattribute im Verzeichnis ändern**

**Syntax** : ATTRIB [+|-r][+|-a] [d:][pfad]dateiname /S

**Optionen** :

| | |
|---|---|
| +r | Datei kann nur gelesen werden (read only) |
| -r | Schreibschutz aufheben |
| +a | Archivattribut setzen |
| -a | Archivattribut löschen |
| /S | Kommando gilt für alle Unterverzeichnisse |

**Funktion** : Die Dateiattribute <nur lesen> und <archiv> werden gesetzt (+) oder gelöscht (-). Die anderen Dateiattribute können mit diesem Kommando nicht verändert werden.

Das Dateiattribut <nur lesen> kann als Schutz gegen versehentliches Löschen benutzt werden (Schreibschutz). Das Archivattribut wird von einigen Programmen (BACKUP, XCOPY, RESTORE) beim Kopieren getestet und kann dadurch zur Dateiauswahl verwendet werden. Bei jeder Dateiveränderung wird das Archivattribut automatisch gesetzt. Zum Setzen der anderen Dateiattribute kann man z.B. das Programm FA der Norton-Utilities benutzen oder sich ein eigenes Programm schreiben und die entsprechende DOS-Funktion verwenden.

**Beispiele:**
```
ATTRIB +a *.*      (Archivattribut setzen)
ATTRIB -a *.bak    (Archivattribut selektiv setzen)
XCOPY  *.* a: /a   (alle Dateien mit Archivattribut
                    kopieren)
```

**MKDIR - Unterverzeichnis anlegen**
**MD**

**Syntax** : MKDIR [d:]pfad

**Optionen** :

| | |
|---|---|
| d: | Laufwerk |
| pfad | Verzeichnisname |

**Funktion** : Auf dem Laufwerk <d>: wird das mit <pfad> angegebene Unterverzeichnis angelegt.

Der Pfadname kann ein vollständiger Zugriffspfad sein (Angabe aller Indexstufen) oder wird im aktuellen Verzeichnis angelegt. Es ist zu berücksichtigen, daß jedes Verzeichnis einen Cluster (z.B. 4 KByte bei Festplatten) beansprucht, auch wenn keine Dateien darin verzeichnet sind. Der Kommandointerpreter verwaltet für jeden Datenträger ein aktuelles Verzeichnis.

**Beispiele:** MD NEU (Anlegen im aktuellen Verzeichnis)
MKDIR D:\VW\V1 (vollständiger Pfadname)
MD ..\NV (Verzeichnis auf gleicher Stufe)

**CHDIR - aktuelles Verzeichnis wechseln**
**CD**

**Syntax** : CHDIR [d:][pfad]

**Optionen** :
d: Laufwerk
pfad neues aktuelles Verzeichnis

**Funktion** : Auf dem Laufwerk <d:> wird das neue aktuelle Verzeichnis <pfad> eingestellt. Ist kein Laufwerk angegeben, so gilt das aktuelle Laufwerk. Fehlt die Angabe <pfad>, so wird das aktuelle Verzeichnis angezeigt.

Für jedes Laufwerk kann ein aktuelles Verzeichnis eingestellt werden. Beim Laufwerkswechsel wird dann sofort das entsprechende Verzeichnis eingestellt.

**Beispiele:** CD D:\ (Wurzelverzeichnis auf D: einstellen)
CD .. (Elternverzeichnis einstellen)
CD D:\MSTOOLS (Verzeichnis MSTOOLS auf D:)
CD (aktuelles Verzeichnis anzeigen)

**RMDIR - Unterverzeichnis entfernen**
**RD**

**Syntax** : RMDIR [d:]pfad

**Optionen** :
d: Laufwerk
pfad Unterverzeichnis, das entfernt werden soll

**Funktion** : Das mit <pfad> angegeben Unterverzeichnis auf Laufwerk <d:> wird entfernt.

Das Unterverzeichnis muß bis auf die Einträge für das Elternverzeichnis (..) und den Verweis auf das Verzeichnis selbst (.) leer sein.

**Beispiele:** DEL D:\MSTOOLS*.* (Dateien löschen)
RD D:\MSTOOLS (Verzeichnis MSTOOLS auf D: löschen)

**DIR - Inhaltsverzeichnis ausgeben**

**Syntax** : DIR [d:][pfad][datei] /Optionen

**Optionen** :

| | |
|---|---|
| d: | Laufwerk |
| pfad | Verzeichnis |
| datei | Dateiname (*, ? erlaubt) |
| /P | Stop nach jeder Bildschirmseite (pause) |
| /W | breite Ausgabe, nur Dateiname (wide) |

**Funktion** : Die durch <d:><pfad><datei> ausgewählten Dateien werden als Liste auf dem Standardausgabegerät ausgeben. Die Ausgabeform wird wird durch /W und /P gesteuert.

Das DIR-Kommando ist eines der am häufigsten verwendeten Kommandos, weil man sich damit einen Dateiüberblick verschaffen kann. Nach der Dateiliste wird der noch verfügbare Platz auf dem Datenträger ausgegeben. Durch Umlenkung der Ausgabe in eine Datei oder auf einen Drucker lassen sich aktuelle Verzeichnisse speichern.

**Beispiele:**

```
DIR              (aktuelles Inhaltsverzeichnis)
DIR C:           (aktuelles Inhaltserzeichnis von C:)
DIR *.PAS        (alle Pascal-Dateien im aktuellen
                  Verzeichnis)
DIR /W           (aktuelles Inhaltsverzeichnis in
                  breiter Ausgabeform)
DIR | SORT /+25 (Umlenkung und Sortierung nach
                  Dateidatum)
```

**TREE - Verzeichnisstruktur ausgeben**

**Syntax** : TREE [d:] /Optionen

**Optionen** :

| | |
|---|---|
| d: | Laufwerk |
| /F | Anzeige von Dateien in Verzeichnissen (file) |

**Funktion** : Die Verzeichnisstruktur von Laufwerk <d:> wird auf dem Standardausgabegerät ausgegeben (Umlenkung ist möglich). Bei der Angabe von /F werden außerdem die Dateinamen angezeigt.

Mit TREE kann man sich eine vollständige Übersicht über Verzeichnisse und Dateien auf einem Datenträger verschaffen. Bei stark belegten Festplatten kann die Ausgabe sehr umfangreich sein. Die Umlenkung der Ausgabe in eine Datei bzw. zum Drucker ergibt eine brauchbare Dateiübersicht für Archivierungszwecke.

**Beispiele:** TREE /F (aktuelles Laufwerk mit Dateinamen)
TREE /F >prn (Ausgabe auf Drucker)
TREE | MORE (seitenweise Ausgabe)

**JOIN - Laufwerk in Verzeichnisstruktur eingliedern**

**Syntax** : JOIN [d1: d2:path]
JOIN d1: /D

**Optionen** :
d1: Laufwerk, das eingegliedert wird
d2:path Laufwerk und Pfad für Eingliederung
/D Eingliederung für <d1:> aufheben

**Funktion** : Das mit <d1:> angegebene Laufwerk wird im Wurzelverzeichnis von <d2:> mit dem Verzeichnisnamen <pfad> eingegliedert. Die Aufhebung der Eingliederung erfolgt mit /D.

Die Eingliederung kann nur in das Wurzelverzeichnis erfolgen. Alle Unterverzeichnisse des eingegliederten Laufwerks lassen sich mit über den neuen Zugriffspfad ansprechen. Das eingegliederte Laufwerk kann nicht angesprochen werden. Wird JOIN ohne Paramter aufgerufen, so erfolgt eine Anzeige der eingegliederten Laufwerke.

**Beispiele:** JOIN A: C:\DATEN (Laufwerk A: wird Unterverzeichnis \DATEN)
JOIN A: /D (A: wird wieder Laufwerk)
JOIN (Eingliederungen anzeigen)

**SUBST - Verzeichnis als Laufwerk vereinbaren**

**Syntax** : SUBST [d:] [pfad] /D

**Optionen** :
d: neue Laufwerksbezeichnung (virtuelles Gerät)
pfad Verzeichnis, das durch <d:> ersetzt wird
/D Vereinbarung wird aufgehoben

**Funktion** : Das mit <pfad> angegebene Verzeichnis wird als virtuelles Gerät vereinbart.
Wird die Option /D angegeben, so wird die Vereinbarung aufgehoben. Sind keine Parameter angegeben, so erfolgt eine Anzeige der gültigen Vereinbarungen.

Die Zuordnung gestattet die Ansteuerung eines Verzeichnisses als Laufwerk. Bei langen Verzeichnisnamen kann das eine Verringerung des Schreibaufwands bedeuten. SUBST kann als komplementäre

Funktion zu JOIN angesehen werden. Der Laufwerksbezeichner muß in dem durch LASTDRIVE angegebenen Bereich liegen (Standard ist A:...E:, vgl. Abschn. 4.). Das aktuelle bzw. in <d:> angegebene Laufwerk kann nicht als virtuelles Gerät angegeben werden. Gerätekommandos (z.B. FORMAT) und Verzeichniskommandos (z.B. CHDIR) sind für virtuelle Geräte nicht erlaubt. Wird das Wurzelverzeichnis eines Datenträgers als neues Laufwerk vereinbart, so entsteht der gleiche Effekt wie bei einem entsprechenden ASSIGN-Kommando.

```
Beispiele: SUBST E: D:\TEXTE\NEU (virtuelles Gerät E:)
           DIR E:                (Inhaltsverzeichnis von
                                  D:\TEXTE\NEU ausgeben)
           SUBST                 (Vereinbarungen anzeigen)
           SUBST E: /D           (Vereinbarung aufheben)
           SUBST A: C:\          (enspricht ASSIGN A=C)
```

**APPEND - Suchpfad für Datendateien festlegen**

**Syntax** : APPEND [d:][pfad][;...]

**Optionen** :

| | |
|---|---|
| d: | Laufwerk für Suchpfad |
| pfad | Verzeichnis für Suchpfad |

**Funktion** : Die mit Laufwerk und Pfad angegebenen Verzeichnisse (mehrere Angaben werden durch ';' getrennt) werden von DOS beim Eröffnen von Dateien durchsucht. Wird als Parameter nur ein ';' angegeben, so werden alle Suchpfade gelöscht. Bei fehlenden Parametern wird die aktuelle Einstellung angezeigt.

Standardmäßig wird beim Eröffnen einer Datei nur im aktuellen Verzeichnis gesucht. Durch APPEND kann die Suche auf andere Verzeichnisse ausgedehnt werden. Wird das Kommando ASSIGN benutzt, muß APPEND vorher angegeben werden. Für Programmdateien gilt das Kommando PATH. Es ist zu beachten, daß ein evtl. vorher eingestellter Suchpfad gelöscht wird. Bei der Definition eines Suchpfads besteht keine Möglichkeit, die vorher evtl. vorhandenen Einträge zu erhalten. Der Suchpfad muß immer vollständig eingegeben werden.

```
Beispiele: APPEND A:\TEXTE;B:\TEXTE1 (Suchen in A: und B:)
           APPEND ;                  (Suchpfad löschen)
           APPEND                    (Suchpfad anzeigen)
```

| **PATH - Suchpfad für Programmdateien festlegen** |
|---|
| **Syntax** : PATH [d:][pfad][;...] |
| **Optionen** :<br>d: Laufwerk für Suchpfad<br>pfad Verzeichnis für Suchpfad |
| **Funktion** : Die mit Laufwerk und Pfad angegebenen Verzeichnisse (mehrere Angaben werden durch ';' getrennt) werden beim Laden von Programmen durchsucht. Wird nur ein ';' angegeben, so werden alle Suchpfade gelöscht. Bei fehlenden Parametern wird die aktuelle Einstellung angezeigt. |

Der Suchpfad wird für Programme (*.COM, *.EXE) und Stapeldateien (*.BAT) benutzt. Standardmäßig wird beim Laden nur im aktuellen Verzeichnis nach Programmen gesucht. Mit PATH kann die Suche auf weitere Verzeichnisse ausgedehnt werden. Für Datendateien gilt das Kommando APPEND. Der Suchpfad wird in den Umgebungsstring eingetragen und bei SET mit angezeigt. Es ist zu beachten, daß ein evtl. vorher eingestellter Suchpfad gelöscht wird.

```
Beispiele: PATH \DOS   (aktuelles Laufwerk, Verzeichnis \DOS)
           PATH C:\    (Laufwerk C:, Wurzelverzeichnis)
           PATH ;      (Suche nur im aktuellen Verzeichnis)
           PATH        (aktuellen Suchpfad anzeigen)
```

## 6.5. Kommandos zur Dateiverwaltung

In diesem Abschnitt werden die Möglichkeiten von DOS zur Ein- und Ausgabe von Dateien und die Kommandos zur Dateiverwaltung zusammengefaßt. Im Abschn. 5. sind die internen Funktionen des DOS erläutert worden. Auf der Ebene des Kommandointerpreters stehen Kommandos zur Verfügung, die die wichtigsten Funktionen für den Nutzer zur Verfügung stellen. Wem die Möglichkeiten nicht ausreichen, der muß sich eigene Programme mit Hilfe der DOS-Funktionen erstellen.

Alle im PC gespeicherten Daten sind in Form von Dateien zusammengefaßt. Jede Datei (File) hat einen eindeutigen Namen und wird in das DOS-spezifische Dateisystem eingeordnet. MS-DOS verwendet eine UNIX-ähnliche Dateistruktur mit mehrstufigen, baumartigen Inhaltsverzeichnissen (Katalog, directory). Eine vollständige Dateibezeichnung wird im folgenden Beispiel gezeigt.

**Beispiel:** C:\Stufe1\Stufe2\Datei.ext

| | |
|---|---|
| C:\ | Gerät, Root-Verzeichnis |
| Stufe1\ | Indexstufe 1 |
| Stufe2\ | Indexstufe 2 |
| Datei | Dateiname, Hauptteil |
| .ext | Dateiname, Erweiterung |

Eine Gruppe von Dateien wird über ein gemeinsames Verzeichnis oder durch Angabe eines Namenmusters mit Stellvertreterzeichen identifiziert. Für ein beliebiges Zeichen in einem Namenteil steht das Zeichen '?', und für eine beliebige Anzahl steht das Zeichen '*'.

**Beispiel:**

| Dateiname | Gültig sind z.B. |
|---|---|
| Bericht?.doc | Bericht1.doc<br>Bericht2.doc |
| Be*.doc | Bericht.doc<br>Bereich1.doc<br>Bemerk.doc |
| *.* | alle Dateien im aktuellen Verzeichnis |

Als Dateierweiterung (max. 3 Zeichen) wählt man am besten eine Zeichenfolge, aus der sich der Dateityp erkennen läßt.

**Beispiel:**

| | |
|---|---|
| .pas | alle Pascal-Texte |
| .wsd | WordStar (Dokumente) |
| .txt | Texte allgemein |

Wird in einem Kommando ein Dateiname ohne Gerätekennzeichen angegeben, so wird das aktuelle Gerät angenommen. Einen Überblick über die in einem Verzeichnis gespeicherten Dateien kann man sich mit dem DIR-Kommando verschaffen. Die Kommandos zur Gerätebehandlung und zur Verzeichnisverwaltung sind in den Abschnitten 6.1. und 6.2. beschrieben.

Die Kommandos zur Dateiverwaltung und Ein- und Ausgabe sind in Tafel 6.9 zusammengefaßt und werden im Anschluß einzeln beschrieben.

Einige Hinweise sind zur Eingabe und Ausgabe von Dateien wichtig. Die **Umleitung der Eingabe** vom Standardeingabegerät ist bei fast allen Kommandos mit dem Umlenkungszeichen '<' möglich. Die **Umlenkung der Ausgabe** erfolgt entsprechend mit '>'. Das **Anfügen** an eine bereits bestehende Datei erfolgt mit dem Zeichen '>>'. Durch die Verbindung von Kommandos über Zwischendateien werden **Kommandoketten (pipes)** erzeugt, die an die Möglichkeiten von UNIX angelehnt sind. Die dazu erforderlichen **Filterprogramme** (FIND, SORT, MORE, TYPE, PRINT) werden durch das Zeichen '|' verkettet. Beispiele folgen bei den einzelnen Kommandos.

**Tafel 6.9. Kommandos zur Dateiverwaltung**

| Kommandos zur Dateiverwaltung | |
|---|---|
| < | Umlenkung der Standardeingabe |
| > | Umlenkung der Standardausgabe |
| >> | Umlenkung der Standardausgabe, Anfügen |
| \| | Kommandofilter (pipe, stdout wird stdin) |
| **FIND** | Suchen einer Zeichenkette (DOS-Filter) |
| **MORE** | Bildschirm seitenweise ausgeben (DOS-Filter) |
| **SORT** | Sätze sortieren (DOS-Filter) |
| **PRINT** | Ausgabe von Text auf den Drucker (stdprn) |
| **TYPE** | Ausgabe von Text auf den Bildschirm (stdout) |
| **COPY** | Kopieren von Dateien |
| **COMP** | Dateien vergleichen |
| **REName** | Datei umbenennen |
| **DELete** | Datei löschen |
| **XCOPY** | Dateien kopieren (erweiterte Version) |

**FIND - Suchen einer Zeichenkette**

**Syntax** : FIND /Optionen "string" [d:][pfad][datei]...

**Optionen** :

| | |
|---|---|
| /N | Ausgabe der Zeilennummer vor der Zeile |
| /C | nur Anzahl Zeilen mit "string" ausgeben |
| /V | Ausgabe der Zeilen ohne "string" |
| string | Zeichenkette, die gesucht werden soll |
| d: | Laufwerk |
| pfad | Verzeichnis |
| datei | Dateiname |

**Funktion** : In der angegebenen Datei ist nach der Zeichenkette <string> zu suchen. Alle Zeilen, in denen die Zeichenkette gefunden wird, werden ausgegeben. Die Ausgabe wird durch Optionen gesteuert. Ist keine Datei angegeben, so wird von der Standardeingabe gelesen. Die Ausgabe erfolgt auf das Standardausgabegerät. Umlenkung ist möglich.

Es können mehrere Dateien angegeben werden (Trenner ist ein Leerzeichen). Ersetzungszeichen (*, ?) sind nicht erlaubt. Die Großschreibung von Zeichen wird berücksichtigt.

**Beispiele:**
```
FIND "Datei" BUCH.TXT
FIND /N "wird" BUCH.TXT  (mit Zeilennummer)
DIR | FIND "TXT"    (TXT-Dateien im Verzeichnis)
FIND "MODE" AUTOEXEC.BAT AUTOEXEC1.BAT
                          (mehrere Dateien)
```

| **MORE - seitenweise Bildschirmausgabe** |
|---|
| **Syntax** : MORE |
| **Funktion** : Die Standardeingabe wird seitenweise auf dem Bildschirm ausgegeben. Umlenkung ist erlaubt. Die Ausgabe wird bei Tastendruck fortgesetzt. |

Bei der Ausgabe langer Dateien z.B. mit TYPE kann durch den Filter MORE der Bildschirm zum Lesen angehalten werden. Es erscheint eine Meldung am unteren Bildschirmrand, daß weitere Zeilen auszugeben sind. Die Ausgabe wird bei jeder beliebigen Taste fortgesetzt. Abbruch ist mit Ctrl-C möglich. Durch MORE wird eine temporäre Datei auf dem aktuellen Laufwerk angelegt. Ist der Datenträger voll oder schreibgeschützt, so wird MORE abgebrochen.

**Beispiele:** MORE <BUCH.TXT (Datei als Standardeingabe)

TYPE BUCH.TXT | MORE (pipe, Ausgabe von TYPE wird Eingabe von MORE)

| **SORT - Sortieren von Textdateien** |
|---|
| **Syntax** : SORT /Optionen |
| **Optionen** :<br>/R absteigende Sortierfolge<br>/+n Sortierfeld ab Spalte <n> (Standard n=1) |
| **Funktion** : Die Standardeingabe wird alphabetisch ab Spalte 1 (bzw. <n>) aufsteigend (bzw. absteigend bei /R) sortiert. Die Ausgabe erfolgt in die Standardausgabedatei. Umlenkungen für Ein- und Ausgabe sind erlaubt. |

Das SORT-Kommando unterscheidet nicht zwischen Klein- und Großbuchstaben. SORT wird meistens in Verbindung mit einer Pipe als Filter eingesetzt.

**Beispiele:** SORT <TEXT.TXT >TEXTS.TXT (Angabe von Dateinamen)

DIR | SORT /+14 | MORE (sortiertes Inhaltsverzeichnis, Ausgabe von DIR wird Eingabe von SORT, Ausgabe von SORT wird Eingabe von MORE)

**PRINT - Ausgabe von Text auf den Drucker (spooling)**

**Syntax** : PRINT [d:][pfad][dateiname] /Optionen

**Optionen :**

| | |
|---|---|
| d: | Laufwerk |
| pfad | Verzeichnis |
| dateiname | Druckdatei |
| /T | alle Druckaufträge löschen |
| /C | Löschmodus einschalten |
| /P | Druckmodus einschalten |

**Funktion** : Mit dieser Form des PRINT-Kommandos wird die eingerichtete Druckwarteschlange verwaltet (vgl. Abschn. 6.1.). Sind keine Parameter angegeben, wird der Status des Spoolers angezeigt. Optionen und Dateinamen können gemischt als Parameter angegeben werden.

Das PRINT-Kommando wird zum Drucken im Hintergrund (spooling) verwendet. Der Druck wird durch Optionen gesteuert, die zusammen mit Druckdateiname als Parameterfolge angegeben werden. Ist der Druckerspooler noch nicht aktiv, muß er vorher installiert werden. Die Installation sollte mit einem gesonderten PRINT-Kommando erfolgen (vgl. Abschn. 6.1.), um die Druckparameter korrekt einzustellen. Eine brauchbare Alternative zum PRINT-Kommando ist das Programm LP der Norton-Utilities, das jedoch nicht im Hintergrund druckt.

**Beispiele:**
```
PRINT BUCH.TXT              (Dateiausdruck)
PRINT                       (Statusanzeige)
PRINT /T                    (Löschen der Druckaufträge)
PRINT /C *.TXT /P BUCH.TXT  (Löschen TXT-Dateien,
                             (Druck der Datei BUCH.TXT)
```

**TYPE - Ausgabe von Text auf Standardausgabe (stdout)**

**Syntax** : TYPE [d:][pfad]dateiname

**Optionen :**

| | |
|---|---|
| d: | Laufwerk |
| pfad | Verzeichnis |
| dateiname | Dateiname |

**Funktion** : Die angegebene Datei wird in die Standardausgabe übertragen. Eine Umlenkung ist möglich. Der Dateiname darf keine Ersetzungszeichen enthalten.

Im Normalfall wird mit dem Kommando TYPE eine Datei auf dem

Bildschirm ausgegeben. Sinnvolle Ausgaben sind nur bei Textdateien (ASCII-Zeichen) zu erwarten, da Programmdateien spezielle Zeichen enthalten können, die die Ausgabe verhindern.

```
Beispiele: TYPE A:Beispiel.txt
           TYPE BUCH.TXT | MORE      (Einsatz als Pipe)
           TYPE BUCH.TXT >prn        (Einsatz zum Drucken)
```

**COPY - Kopieren von Dateien**

**Syntax** : COPY /Optionen quelle ziel

**Optionen** :

| | |
|---|---|
| quelle | Ursprungsdatei(en) (evtl. Gerät, Pfad) |
| ziel | Zieldateien (evtl. mit Gerät, Pfad) |
| /A | Kopieren von ASCII-Dateien (Standard) |
| /B | Kopieren von Binärdateien |
| /V | Verifizieren |

**Funktion** : Die mit <quelle> angegebenen Dateien oder Geräte werden auf die Dateien oder das Gerät <ziel> kopiert. Ist /V angegeben, so erfolgt ein Verifizieren.

Mit dem COPY-Kommando lassen sich Dateien innerhalb eines Verzeichnisses oder auch zwischen verschiedenen Verzeichnissen kopieren. Für Laufwerk und Verzeichnis werden bei fehlender Angabe die aktuellen Werte angenommen. Falls bei <ziel> kein Dateiname angegeben ist, wird der Name der Eingabedatei verwendet. Mehrere Dateien können zu einer großen zusammengefügt werden, indem sie bei <quelle>, durch '+' getrennt, aufgeführt werden. Binärdateien müssen mit der Option /B kopiert werden, weil sonst das Ctrl-Z als Dateiende interpretiert wird. Der Zieldateiname sollte nicht mit einem der Ursprungsdateien übereinstimmen, weil sonst evtl. vor dem Kopieren ein Überschreiben stattfindet. Wird als Quelle oder Ziel ein Gerät angegeben, so können die Doppelpunkte auch fehlen (reservierte Namen). Für die Dateizugriffe werden die DOS-Funktionen ab Version 2.0 genutzt.

```
Beispiele: COPY TEXT.DOC TEXT.BAK   (Kopie im Verzeichnis)
           COPY D:TEXT.DOC C:       (Kopie auf anderes Gerät)
           COPY TEIL1.DOC+TEIL2.DOC GESAMT.DOC
                                    (Dateien zusammenfügen)
           COPY *.TXT GESAMT.DOC    (Dateien zusammenfügen)
           COPY *.LST prn:          (Ausgabe auf Gerät)
           COPY /B *.EXE D:         (Programme kopieren)
           COPY CON PRN             (Schreiben auf Drucker)
```

**COMP - Dateien vergleichen**

**Syntax** : COMP [datei1 datei2]

**Optionen** :
datei1 1. Datei zum Vergleich
datei2 2. Datei zum Vergleich

**Funktion** : Die beiden Dateien werden miteinander verglichen (gleiche Länge ist Bedingung). Wenn mehr als 10 Vergleichsfehler auftreten, wird der Vergleich abgebrochen. Anstelle der Dateien können auch Dateigruppen angegeben werden. Wird das Kommando ohne Parameter eingegeben, erfolgt eine Abfrage der Dateinamen.

Für den Dateivergleich existieren mehrere Programme (z.B. FC mit weiteren Möglichkeiten zur Steuerung des Vergleichs). Sollen ganze Datenträger gleicher Struktur miteinander verglichen werden, so ist das Kommando DISKCOMP zu verwenden. Beim Vergleich werden Fehler durch den relativen Abstand zum Dateianfang angegeben (Offset in Byte). Die Differenzmeldung kann in eine Datei umgelenkt werden.

**Beispiele:**
```
COMP a:\*.txt d:\texte\*.txt (Vergleich mehrerer
                              Dateien)
COMP       (Dateinamen interaktiv angeben)
COMP TEXT1 TEXT2 >DIFF
```

**REName - Datei umbenennen**

**Syntax** : REN [d:][pfad]datei1 datei2

**Optionen** :
d: Laufwerk
pfad Verzeichnis
datei1 alter Dateiname
datei2 neuer Dateiname

**Funktion** : Einzelne Dateien oder auch Dateigruppen wermit einem neuen Namen ins Verzeichnis eingetragen. Der Wechsel des Verzeichnisses ist nicht möglich.

In den Namen sind Ersetzungszeichen möglich (*, ?); dadurch können mehrere Dateien in einem Kommando angesprochen werden. Durch das Umbenennen wird nur der Name im Inhaltsverzeichnis geändert. Als Kommandoname kann auch RENAME angegeben werden.

**Beispiele:**
```
REN Datei.alt *.neu  (Namensteile mit * bleiben)
REN *.TXT *.LST      (Umbenennen ganzer Gruppen)
```

**DEL - Löschen von Dateien**

**Syntax** : DEL [d:][pfad]datei

**Optionen** :

| | |
|---|---|
| d: | Laufwerk |
| pfad | Verzeichnis |
| datei | Datei(gruppe) |

**Funktion** : Alle bezeichneten Dateien werden gelöscht. Wird ein ganzes Verzeichnis gelöscht, so erfolgt eine Rückfrage. Ersetzungszeichen sind erlaubt.

Beim Löschen wird der Verzeichniseintrag als frei gekennzeichnet und der Speicherplatz in der FAT freigemeldet. Fehlt die Dateiangabe, wird ein ganzes Verzeichnis gelöscht. Zur Vermeidung versehentlichen Löschens wird dabei eine Bestätigung verlangt. Als Kommandoname kann auch ERASE angegeben werden.

**Beispiele:**
```
DEL text.doc
DEL text.*
DEL d:
DEL *.*
```

**XCOPY - erweitertes Kopieren von Dateien**

**Syntax** : XCOPY quelle ziel /Optionen

**Optionen** :

| | |
|---|---|
| quelle | Ursprungsdatei(en) (Laufwerk, Pfad, Datei) |
| ziel | Zieldatei(en) (Laufwerk, Pfad, Datei) |
| /A | Kopieren aller Dateien mit Archivattribut (Archivbit wird nicht verändert) |
| /D:<datum> | Kopieren aller Dateien ab <datum> |
| /E | Unterverzeichnisse kopieren, auch wenn leer |
| /M | Kopieren aller Dateien mit Archivattribut (Archivbit wird zurückgesetzt) |
| /P | Bestätigung bei jeder Datei mit Y/N (j/n) |
| /S | Kopieren der Dateien und Unterverzeichnisse |
| /V | Verifizieren |
| /W | Pause zum Diskettenwechsel vorm Kopieren |

**Funktion** : Alle mit <quelle> angegebenen Dateien werden nach <ziel> kopiert. Mit dem Kommando können im Unterschied zu COPY auch Unterverzeichnisse und ganze Datenträger logisch kopiert werden. Der Kopiervorgang läuft über den Speicher ab. Der Rückkehrcode kann zur Fehlerauswertung genutzt werden.

Das Kommando ist erst seit der Version 3.2 verfügbar und erledigt viele Funktionen besser als das Kommando COPY. COPY ist

jedoch ein internes Kommando und kann damit schneller aufgerufen werden. Zum Kopieren ganzer Datenträger ist das Kommando DISKCOPY besser geeignet. Bei der Dateiauswahl anhand des Datums ist die im System verwendete Form zu nutzen.

**Beispiele:** XCOPY A: B: /S /E (kopiert ganzen Datenträger logisch mit Verzeichnissen)
XCOPY *.TXT C:\ /S

## 6.6. Fehlerbehandlung

Die Fehlerbehandlung in MS-DOS ist relativ einfach. Es werden **Gerätefehler** und **Fehlermeldungen der DOS-Kommandos** unterschieden. Einige DOS-Kommandos liefern einen Fehlercode zurück, der z.B. in Kommandodateien zur gezielten Fehlerbehandlung genutzt werden kann. Bei Gerätefehlern wird eine kurze Mitteilung über die Art des Fehlers, die Zugriffsart (Lesen, Schreiben) und mögliche Reaktionen ausgegeben. Die Ausschriften erfolgen in englisch oder deutsch und sind selbsterklärend. Als Reaktionen werden folgende Alternativen angeboten:

- A [bort] Abbruch des laufenden Programms
- R [etry] Schreiben/Lesen wiederholen
- I [gnore] Fehler ignorieren.

Bei einigen Fehlern ist sofort zu erkennen, daß jeder weitere Versuch zum gleichen Fehler führt (z.B. Schreibschutz nicht entfernt). Beim Lesen wertvoller Daten sollte man auf jeden Fall mehrfach wiederholen, bevor man zu weiteren Maßnahmen greift. Das trifft auch zu, wenn man z.B. Disketten auf einem 40-Spur-Laufwerk lesen will, die auf einem 80-Spur-Laufwerk im 40-Spur-Modus geschrieben wurden (Übernahme von AT nach XT o.ä). In einigen Fällen läßt sich ein Fehler auch durch neues Einlegen der Diskette und weiteren Leseversuch vermeiden, da die Diskette dabei neu zentriert wird. Das Ignorieren führt in jedem Fall zu unvollständigen Ergebnissen.

Die Fehlermeldungen der DOS-Kommandos sind ebenfalls selbsterklärend und führen in den meisten Fällen zum Programmabbruch. Wird eine Bestätigung für eine Aktion gefordert, so ist mit 'Y' (Yes, ja) oder 'N' (No, nein) zu antworten. Eine Erklärung der Fehlerausschriften ist im Handbuch zu MS-DOS zu finden.

Werden DOS-Funktionen aus eigenen Programmen heraus aufgerufen, so erfolgt eine Standardfehlerbehandlung, die durch eigene Fehlerbehandlungsroutinen ersetzt werden kann. Auf die Programmierung von DOS-Funktionen wird im Abschn. 5. näher eingegangen. Eine Fehlerliste dazu ist im Anhang zu finden.

# 7. Stapelverarbeitung

Beim Einsatz des PC mit häufig wiederkehrenden Befehlsabläufen ist die Anwendung der Stapelverarbeitung (batch), die durch den Kommandointerpreter COMMAND.COM realisiert wird, sinnvoll.

**Vorteile der Stapelverarbeitung**

- Abarbeitung einer Befehlsfolge ohne Nutzereingriff
- getestete Kommandofolge ist gespeichert (keine Fehler)
- Abkürzung häufig benutzter Kommandos
- Parametrisierung von Kommandos

Die Befehlsfolgen werden in einer Textdatei (ASCII) mit der Dateierweiterung .BAT gespeichert und wie ein Programm gestartet. Der Kommandointerpreter liest diese Datei und aktiviert die entsprechenden Programme. Zur effektiven Kommandoprogrammierung sind eine Reihe interner DOS-Kommandos verfügbar, die einen automatischen Ablauf und eine Parametrisierung gestatten. Sie sind in Tafel 7.1 zusammengefaßt. Eine Kommadoprozedur wird mit einem normalen Texteditor erstellt. Eine einfache Möglichkeit zur Textaufnahme besteht auch in der Nutzung des COPY-Kommandos (COPY CON batch.BAT). Innerhalb einer Kommandoprozedur dürfen alle DOS-Kommandos zusätzlich zu denen aus Tafel 7.1 benutzt werden. Zur effektiven Arbeit mit Kommandoprozeduren wurde die Möglichkeit zur **Ersetzung von Parametern** geschaffen. Die beim Aufruf angegebenen Parameter ersetzen intern mit %1 ... %9 bezeichnete Parameter. Der Parameter %0 ersetzt den Kommandonamen. Der **Abbruch** einer Kommandofolge ist durch Eingabe von Ctrl-C nach einer Bestätigung möglich. Die notwendige Bestätigung schützt gegen die versehentliche Unterbrechung. Bei Abbruch sind die Ergebnisse der Kommandofolge in der Regel unvollständig.

Beim Start des Kommandointerpreters wird automatisch nach einer Kommandodatei mit dem Namen AUTOEXEC.BAT gesucht. In dieser Datei kann z.B. die Einstellung gewünschter Systemparameter (aktuelles Verzeichnis, Datum, Uhrzeit u.a.) oder der Start eines Anwendungsprogramms (z.B. für das Büro) erfolgen.

Im Abschn. 7.1. werden die Kommandos zur Stapelverarbeitung beschrieben, und im Abschn. 7.2. folgen einige Hinweise zur Gestaltung von Kommandofolgen und Beispiele.

## 7.1. Kommandos zur Stapelverarbeitung

In diesem Abschnitt werden die verfügbaren Kommandos zur Stapelverarbeitung einzeln beschrieben. Die Kommandosprache ist

zwar relativ einfach, es lassen sich jedoch leistungsfähige Kommandoprozeduren damit realisieren, die bei häufigem Einsatz eine gute Unterstützung leisten. Da der regelmäßige Einsatz ausgetesteter Kommandofolgen meistens durch Anwender erfolgt, die nicht unbedingt mit der Programmierung vertraut sind, sollte man die Ablauffolge nicht zu kompliziert gestalten und eine Reihe von (abschaltbaren) Kommentaren einbauen.

**Tafel 7.1. Kommandos zur Stapelverarbeitung (batch)**

| Kommandos zur Stapelverarbeitung | |
|---|---|
| **ECHO** | Steuerung der Kommandoanzeige |
| **CALL** | Kommandounterprogramm aufrufen |
| **FOR** | Kommandowiederholung |
| **GOTO** | Sprung zu einer Marke |
| **IF** | bedingter Sprung |
| **PAUSE** | Unterbrechung |
| **REM** | Kommentarzeile |
| **SHIFT** | Parameterbehandlung |

**ECHO - Steuerung der Kommandoanzeige**

**Syntax** : ECHO [meldung]
ECHO ON
ECHO OFF

**Optionen** :
meldung Ausgabe des Textes <meldung> auf Stdout
ON Echomodus anschalten
OFF Echomodus abschalten

**Funktion** : Mit ECHO wird die Anzeige der in der Prozedur vorhandenen Kommandos an- oder abgeschaltet. Außerdem besteht die Möglichkeit zur Ausgabe von Text auf dem Bildschirm.

Im eingeschalteten Echomodus wird jedes Kommando angezeigt, und man hat eine Kontrolle über die intern ablaufenden Kommandos. Die Einstellung ECHO ON ist zum Test einer Prozedur zu empfehlen. Wird das Kommando ohne Parameter aufgerufen, so wird der aktuelle ECHO-Modus angezeigt.

**Beispiele:** ECHO Test aller DOS-Funktionen (Textausgabe)

ECHO OFF (kein Echo)

**CALL - Aufruf Kommandounterprogramm**

**Syntax** : CALL batch [parm]

**Optionen** :
batch Kommandounterprogramm (.BAT)
parm Aufrufparameter

**Funktion** : Der Kommandointerpreter verzweigt in das Kommandounterprogramm. Das rufende Programm wird nach Beendigung nach dem CALL fortgesetzt.

Mit dem CALL-Kommando lassen sich modular aufgebaute Kommandoprogramme aufbauen. Eine andere Möglichkeit zum Start von Kommandoprogrammen besteht darin, einen neuen Kommandointerpreter zu laden (COMMAND ...) und ein Kommando zur Abarbeitung zu übergeben. Es besteht auch die Möglichkeit, als letzte Anweisung in einer Kommandoprozedur eine weitere aufzurufen (Ketten von Kommandoprozeduren). Die Variante mit CALL, die mit der Version 3.3 eingeführt wurde, ist insgesamt für diesen Zweck besser geeignet.

**Beispiele:** CALL Kopieren a:*.* c:\ (Aufruf mit Parametern)
CALL autoexec

**FOR - Kommandowiederholung**

**Syntax** : FOR %%string IN (menge) DO kommando %%string

**Optionen** :
string Laufvariable für Elemente aus <menge> bzw. Parameter für <kommando>
menge Wertebereich für <string>
kommando Kommando mit Parameter <string>

**Funktion** : Der Laufvariablen werden nacheinander die in <menge> angegebenen Werte zugewiesen und das Kommando mit dem Parameter aufgerufen. Die %%-Zeichen dienen zur Unterscheidung von Aufrufparametern (die Angabe von 0...9 ist daher in <string> nicht gestattet).

Als Elemente der Menge können sowohl Dateinamen als auch Parametervariablen (%0...%9) stehen. Ersetzungszeichen sind erlaubt. Das FOR-Kommando ist auch im normalen Kommandomodus (außerhalb von Kommandoprozeduren) erlaubt; dabei ist zur Kennzeichnung der Laufvariablen jedoch nur ein %-Zeichen anzugeben.

**Beispiele:**

```
FOR %%a in (*.txt) DO TYPE %%a (TYPE *.txt)
FOR %%c in (%1 %2) DO COPY %%c (Parameter als Menge)
```

**GOTO - Sprung zu einer Marke**

**Syntax** : GOTO marke

**Optionen** :
marke Sprungziel, definiert als :marke

**Funktion** : Mit GOTO wird ein Sprung zu einer Marke ausgeführt. Die Marke <marke> steht am Beginn einer Zeile innerhalb der Kommandofolge und beginnt mit einem Doppelpunkt (:marke).

Die Marke muß allein in einer Zeile stehen, da eine mit einem Doppelpunkt beginnende Zeile bei der Kommandoabarbeitung nicht berücksichtigt wird. Leerzeichen innerhalb der Marke sind erlaubt. GOTO wird oft zusammen mit dem IF-Kommando eingesetzt.

**Beispiel:**
```
:Start
ECHO GOTO-Schleife Abbruch mit Ctrl-C
GOTO :Start
```

**IF - bedingte Ausführung von Kommandos**

**Syntax** : IF [NOT] bedingung kommando

**Optionen** :

| | | |
|---|---|---|
| bedingung | string1==string2 | (Vergleich von Strings) |
| | ERRORLEVEL zahl | (Test Returncode) |
| | EXIST dateiname | (Dateitest) |

**Funktion** : Das Kommando wird ausgeführt, wenn die angegebene Bedingung erfüllt ist. Als Kommandos sind auch Stapelkommandos zugelassen.

Mit diesem Kommando lassen sich relativ einfach leistungsfähige Kommandoprozeduren erzeugen. Der Returncode wird nicht durch alle Kommandos gesetzt. Im Zweifelsfall sollte man experimentieren. Eine Null bedeutet normalerweise eine fehlerfreie Bearbeitung.

**Beispiel:**
```
:Start
ECHO OFF
FORMAT a: /s
IF ERRORLEVEL 0 GOTO Ende
ECHO Formatierungsfehler, Wiederholung
GOTO Start
:Ende
```

| **PAUSE - Unterbrechung** |
|---|
| **Syntax** : PAUSE [meldung] |
| **Optionen** :<br>meldung Text, der bei Unterbrechung ausgegeben wird |
| **Funktion** : Die Kommandoabarbeitung wird unterbrochen, die evtl. vorhandene Meldung ausgegeben und eine Anforderung zur Fortsetzung durch Betätigung einer Taste ausgeschrieben. |

Dieses Kommando wird benutzt, um notwendige Bedienhandlungen anzufordern (z.B. Diskette wechseln), bei dem die Verarbeitung angehalten werden muß. Ein Abbruch der Kommandobearbeitung ist mit Ctrl-C möglich. Die Meldung wird nur ausgegeben, wenn ECHO ON gilt.

**Beispiel:** PAUSE Bitte neue Diskette in A: einlegen

| **REM - Kommentarzeile** |
|---|
| **Syntax** : REM [meldung] |
| **Optionen** :<br>meldung ein Kommentar, der im Modus ECHO ON auf dem Terminal erscheint |
| **Funktion** : Das Kommando dient zur Aufnahme von Kommentaren in die Kommandofolge, die im ECHO-Modus angezeigt werden. |

Das Kommando kann, außer zur Aufnahme von Kommentaren, zur Ergänzung des PAUSE-Kommandos bei langen Anweisungen genutzt werden.

**Beispiel:**
```
REM Kommentar 1
REM Kommentar 2
PAUSE Handlungsanweisung
```

| **SHIFT - Parameterbehandlung** |
|---|
| **Syntax** : SHIFT |
| **Funktion** : Normalerweise können 10 Parameter %0...%9 an Kommandoprozeduren übergeben werden. Mit SHIFT wird die aktuelle Parameterliste um eine Position nach links verschoben. Der Parameter %0 ist dann z.B. nicht mehr verfügbar, und %9 wird durch einen weiteren Parameter ersetzt. |

Das SHIFT-Kommando kann auch verwendet werden, wenn weniger als 10 Parameter übergeben werden.

**Beispiel:**
```
:loop     (DELETE Datei1.*  Datei2.*  Datei3.*)
DEL %1
SHIFT
IF "%1" == "" GOTO :end
DEL %1
GOTO loop
:end
```

## 7.2. Gestaltung von Kommandofolgen

Die Kommandoprozeduren sind eine wirkungsvolle Möglichkeit zur Gestaltung von häufig wiederholten Kommandofolgen. Die Kommandosprache gestattet eine einfache Form der Programmierung, die etwa mit BASIC zu vergleichen ist. Die Erstellung von Kommandoprozeduren erfolgt wie bei einem Programm. Folgende Hinweise sollten dabei beachtet werden:

- Eine Kommandoprozedur hat die Dateierweiterung BAT.
- Beim Aufruf wird der Prozedurname ohne Erweiterung angegeben. Kommandoprozeduren werden wie einfache Kommandos behandelt.
- Treten Kommandos und Prozeduren mit dem gleichen Namenshauptteil auf, so erfolgt die Suche in der Reihenfolge COM-Datei, EXE-Datei, BAT-Datei. Ein eindeutiger Aufruf ist durch Angabe der Erweiterung möglich.
- Das letzte Kommando in einer Prozedur kann ein weiterer Prozedurname sein. Es entsteht eine Kette von Prozeduren.
- Wird ein Datenträger während der Abarbeitung einer Prozedur entfernt, so erfolgt eine Aufforderung, wenn der Datenträger wieder benötigt wird.
- Lokale Parameter werden mit %0...%9 bezeichnet, globale Parameter werden im Umgebungsstring eingetragen und mit %name% aufgerufen.
- Umlenkungen der Standardeingabe und -ausgabe sind innerhalb einer Kommandoprozedur erlaubt.
- Das Setzen des Umgebungsstrings mit SET ist gültig für alle folgenden Kommandos innerhalb der Prozedur.
  erreichbar sein (d.h. im aktuellen Verzeichnis stehen, mit dem PATH-Kommando eingestellt oder mit Laufwerk und Verzeichnis aufgerufen werden).
- Die Textaufnahme erfolgt mit einem Texteditor (z.B. EDLIN, WordStar, MS-Word, Editor von Turbo-Pascal) oder mit dem COPY-Kommando. Zur Korrektur ist ein Editor erforderlich.
- Zum Test sollte mit ECHO ON gearbeitet werden.

**Beispiel 1:** Aufbau einer Kommandoprozedur AUTOEXEC.BAT

```
AUTOEXEC.BAT                    Bemerkung
-------------------------------------------------------------
ECHO off                        ECHO ausschalten
CLS                             Bildschirm löschen
GRAFTABL                        Graphikzeichen laden
KEYB gr                         deutscher Tastaturtreiber
PATH c:\;c:\dos;c:\tools;       Suchpfad für Programme
DATE                            Datum erfragen
TIME                            Uhrzeit einstellen
PROMPT $p$g                     Prompt Laufwerk, Verzeichnis>
VER                             aktuelle Versionsnummer
ECHO on                         ECHO-Modus einschalten
```

Aufruf: AUTOEXEC

**Beispiel 2:** Kopieren mehrerer Dateien nach Laufwerk A:

```
COPYTOA.BAT                       Bemerkung
-------------------------------------------------------------
ECHO off                          ECHO ausschalten
:neu                              Marke für Schleife
COPY %1 a:                        Kopieren
SHIFT                             nächster Parameter (Datei)
IF "%1" == "" GOTO ende           Ende, wenn alle Dateien kopiert
GOTO neu                          Wiederholung
:ende                             Endemarke
ECHO on                           ECHO-Modus einschalten
ECHO Dateien kopiert              Endemeldung
```

Aufruf: COPYTOA Datei1.dat Datei2.dat Datei3.dat

**Beispiel 3:** Kontrolliertes Formatieren

Die Kommandodatei FORMAT.BAT wird statt des Formatierprogramms aufgerufen, das in FORMATA.COM umgenannt wird, um das versehentliche Formatieren der Festplatte zu verhindern.

```
FORMAT.BAT                      Bemerkung
-------------------------------------------------------------
ECHO off                        ECHO ausschalten
FORMAT a:                       Formatierung Laufwerk a:
ECHO on                         ECHO-Modus einschalten
```

Aufruf: FORMAT (Parameter werden ignoriert)

**Beispiel 4:** Setzen und Anzeigen Umgebungsstring

```
SETLIB.BAT                      Bemerkung
-------------------------------------------------------------
ECHO off                        ECHO ausschalten
SET %1
ECHO - Globale Prameter -       Titelzeile
SET                             Umgebungsstring anzeigen
ECHO on
```

Aufruf: SETLIB c:\tools\lib

# 8. Werkzeuge zur Softwareentwicklung

Die Entwicklung eigener Programme wird durch Werkzeuge (Hilfsmittel, tools) zur Programmentwicklung unter MS-DOS sehr gut unterstützt. Im Abschn. 8.1. werden die wesentlichen Arbeitsschritte zur Programmerstellung kurz beschrieben und im Abschn. 8.2. die dazu vorhandenen Standardwerkzeuge erläutert. Im Abschn. 8.3. wird auf das Programmiersystem für Turbo-Pascal etwas näher eingegangen, weil die Beispiele in diesem Buch in Turbo-Pascal formuliert sind und die Sprache auch zu einer Art Standard für PC geworden ist.

## 8.1. Entwicklungsetappen eines Programms

Der Weg von der Problemstellung bis zu einem ausführbaren Programm ist manchmal sehr langwierig. Mit dem PC hat man jedoch ein sehr gutes Hilfsmittel zur Programmentwicklung, das stets zur Verfügung steht (bzw. stehen sollte). Es kann in diesem Buch keine Einführung in die Grundlagen der Softwaretechnologie erfolgen; es werden lediglich die wichtigsten Begriffe, Arbeitsschritte und die dabei entstehenden Dateien zusammenfassend dargestellt.

Ein Programm durchläuft mehrere Phasen, für die jeweils ein Verwaltungs- und Transformationsprogramm zuständig ist (Bild 8.1). Der in einer Programmiersprache formulierte **Quelltext** (source program) wird mit einem **Editor** in das Dateisystem aufgenommen und liegt danach als Textdatei in einem Verzeichnis vor. Der Editor übernimmt auch die Korrekturfunktionen. Durch einen **Assembler oder Compiler** wird der Quelltext in das Objektmodulformat (Maschinenkode) übertragen. Sind im Quelltext Fehler enthalten, so erfolgt eine Anzeige und anschließende Fehlerkorrektur. Fertige Module können in eine **Modulbibliothek** eingetragen werden, die durch den **Bibliothekar** (**Librarian**) verwaltet wird. Mehrere Module werden vom **Linker** (**Programmverbinder**) zu einem **ablauffähigen Programm** zusammengefügt. Dabei werden allgemein verwendbare Moduln aus Bibliotheken entnommen. Die Bibliotheken sind Bestandteil eines Compilersystems (z.B. Fortran, Pascal, C). Das ablauffähige Programm enthält meistens in der Anfangsphase mehrere Fehler, die mit einem **Fehlersuchprogramm (Debugger)** bzw. Testläufe gefunden werden. Der Zyklus wird so lange wiederholt, bis ein fehlerfreies Programm vorliegt. Zur Verwaltung der fertigen Programme gibt es dann evtl. weitere Transformationsprogramme. Durch die Ladefunktion (EXEC) des Betriebssystems wird das Pro-

gramm in den Speicher geladen und gestartet.

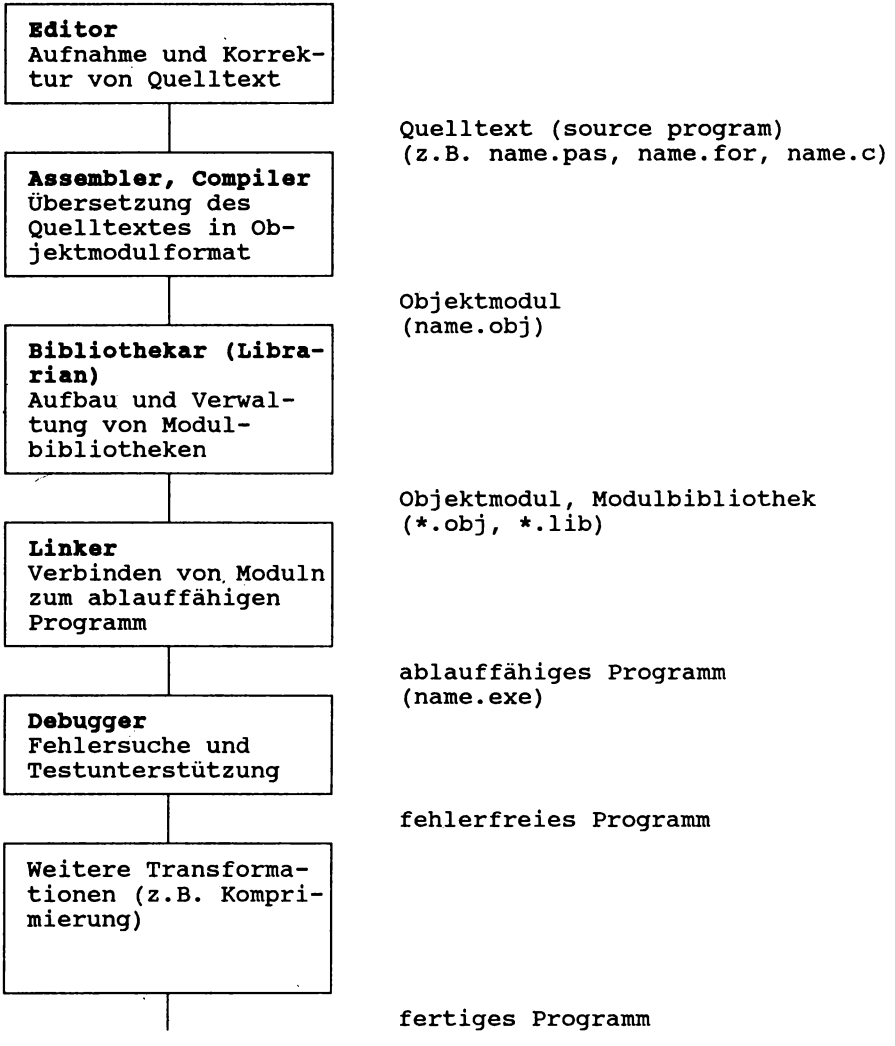



**Bild 8.1. Phasen der Programmentwicklung**

Für alle Phasen gibt es unter DOS eine Fülle von Programmen, so daß dem Softwareentwickler die Wahl nicht leicht gemacht wird. Von der Firma Microsoft, dem Entwickler des MS-DOS, werden für fast alle Programmiersprachen gute Compilersysteme angeboten. Auch für die anderen Phasen der Programmentwicklung gibt es ausgebaute Werkzeuge (tools). Neben diesen Produkten aus einem Softwarehaus gibt es spezialisierte Produkte anderer Hersteller, die z.T. gute Alternativen darstellen. Im folgenden Abschnitt wird

auf die einzelnen Standardwerkzeuge etwas näher eingegangen, wobei besonders auf die Verbindung zu den Funktionen des DOS Wert gelegt wird.

## 8.2. Standardwerkzeuge

In diesem Abschnitt werden die Standardwerkzeuge zur Unterstützung der im Abschn. 8.1. genannten Phasen der Programmentwicklung beschrieben (Tafel 8.1). Eine Vollständigkeit wird dabei nicht angestrebt; sie ist im Rahmen dieses Buches auch gar nicht möglich. Bei der Entwicklung immer neuer Versionen von Werkzeugen ist die Tendenz zu erkennen, alle Werkzeuge mit ähnlicher Bedienungsform zu realieren und damit die Hemmschwelle zu verringern, sich der Werkzeuge auch zu bedienen.

**Tafel 8.1. Standardwerkzeuge zur Programmentwicklung**

| Standardwerkzeuge | |
|---|---|
| **EDLIN** | Zeileneditor |
| **MASM** | Makroassembler |
| **C** | Compiler für Sprache C |
| **FORTRAN** | Compiler für Sprache FORTRAN |
| **PASCAL** | Compiler für Sprache Pascal |
| **LIB** | Bibliotheksverwalter |
| **LINK** | Programmverbinder (Linker) |
| **DEBUG** | einfaches Fehlersuchprogramm |
| **CodeView** | komfortables Fehlersuchprogramm (Symbole) |
| **MAKE** | Modulverwaltung |
| **CREF** | Erzeugen einer Querverweisliste |
| **MAPSYM** | Erzeugen einer Symboltabelle |
| **EXE2BIN** | Umwandlung EXE-Format in COM-Format |
| **EXEMOD** | Modifikation des EXE-Vorspanns |
| **EXEPACK** | EXE-Datei komprimieren |
| **ERROUT** | Standardfehlerdatei (stderr) umlenken |

### 8.2.1. Editor (Verwaltung von Programmtexten)

Der Editor gehört zu den allgemeinen Werkzeugen, die nicht an eine Programmiersprache gebunden sind. Der Quelltext wird als ASCII-Datei erfaßt und in einem Verzeichnis abgelegt. In Tafel 8.2 sind einige Editoren aufgeführt, die zur Verwaltung von Programmtexten geeignet sind.

**Tafel 8.2. Editoren zur Programmtextverwaltung**

| Editoren zur Programmtextverwaltung | |
|---|---|
| **EDLIN** | einfacher Zeileneditor für MS-DOS (Microsoft) |
| **WORD** | allgemeines Textsystem (Microsoft) (vgl. Abschn. 10.1.2.) |
| **WS** | WordStar - allgemeines Textsystem (MicroPro) (vgl. Abschn. 10.1.1.) |
| **NE** | Norton Editor (Peter Norton) |
| **EDIT** | Bildschirmeditor (IBM) |
| **MS** | MicroStar - Editor Toolbox zu Turbo-Pascal (Borland) |

Neben diesen speziellen Textprogrammen gibt es noch eine Reihe weiterer Programme, die u.a. auch für die Programmtextverwaltung geeignet sind. Die integrierten Softwarepakete sind meist viel zu aufwendig für die Programmtextverwaltung, und die in Nutzerinterfaceprogramme eingebauten Editoren (z.B. Norton Commander) sind meistens nur für kurze Texte geeignet. Einige Compilersysteme (z.B. Turbo-Compiler von Borland und die neuen Quick-Compiler von Microsoft) enthalten eingebaute Editoren, die natürlich genutzt werden sollten. Die in Tafel 8.2 genannten Editoren enthalten meistens eine **Hilfsfunktion**, mit der man sich leicht in die Nutzung einüben kann. Schnelligkeit beim Editieren erwirbt man nur durch häufige Benutzung. Da der Editor EDLIN mit dem DOS ausgeliefert wird, sollen die Kommandos hier kurz angeführt werden, obwohl er wegen seiner wenig komfortablen Unterstützung nur sehr selten angwendet wird. Bei der Auswahl seines Editors sollte man darauf achten, daß man mehrere Fenster verwalten kann. Das macht z.B. der Norton Editor NE.

**EDLIN - Zeileneditor**

**Syntax** : EDLIN [d:][path]datei /Option

**Optionen** :

| | |
|---|---|
| d: | Laufwerk (Standard: aktuelles Laufwerk) |
| path | Verzeichnis (Standard: aktuelles Verz.) |
| datei | Datei, die korrigiert werden soll |
| /B | Bearbeitung einer Binärdatei |

**Funktion** : zeilenorientierte Editierkommandos, Zeilennumern werden absolut oder relativ an gegeben. Die aktuelle Zeile wird durch einen Punkt angegeben.

```
Kommandos:
[n]A                  n Zeilen aus Datei anfügen (append)
[von],[bis],nach,[n]C n Zeilen kopieren           (copy)
[von][,bis]D          Zeile(n) löschen          (delete)
[num]                 Zeile <num> zum Editieren  (edit)
E                     Ende mit Sichern            (end)
[num]I                Einfügen vor Zeile <num> (insert)
                      Abschluß mit <Ctrl-C>
[von][,bis]L          Zeilen ausgeben             (list)
[von],+num,nachM      Zeilen verschieben          (move)
[von][,num]P          Blättern seitenweise        (page)
Q                     Ende ohne Sichern           (quit)
[von][,bis][?]R       Text ersetzen            (replace)
   text1<Ctrl-Z>text2
[von][,bis][?]Stext   Text suchen               (search)
[nach]Tdateiname      Datei einfügen           (transfer)
[n]W                n Zeilen in Datei schreiben (write)
```

Mit EDLIN lassen sich alle wichtigen Editierfunktionen ausführen. Es ist zu beachten, das die Zeilennummern bei Einfügeoperationen verändert werden. Zum Editieren des Textes innerhalb einer Zeile können die für MS-DOS gültigen Editier- und Funktionstasten verwendet werden (vgl. Anhang).

### 8.2.2. Assembler und Compiler

Für alle gängigen Programmiersprachen stehen Interpreter oder Compiler unter DOS zur Verfügung. Mit dem Betriebssystem wird davon jedoch nur ein BASIC-Interpreter ausgeliefert. Bei einigen PC-Typen ist ein einfacher BASIC-Interpreter sogar im ROM enthalten und wird bei fehlendem System automatisch gestartet. Obwohl BASIC wegen seiner einfachen Syntax und Handhabung noch eine weite Verbreitung hat und auch durch neue interaktive Compiler (z.B. Turbo-BASIC von Borland, QuickBasic von Microsoft) noch weiter unterstützt wird, wird es doch zunehmend durch andere Sprachen (z.B. Turbo-Pascal) verdrängt. In diesem Buch wird daher auf BASIC nicht weiter eingegangen.

In Tafel 8.3 sind einige Sprachen genannt, die eine wichtige Rolle bei der Programmentwicklung mit und für den PC spielen. Auf die Sprachen MASM, PASCAL, C und FORTRAN soll in den folgenden Abschnitten etwas näher eingegangen werden. Bei den Compilern, die durch IBM zum PC-DOS angeboten werden, handelt es sich meist um geringfügig modifierte Versionen der MS-Compiler. Die Bibliotheken nutzen vielfach die BIOS-Schnittstelle, so daß die Programme dann nur auf kompatiblen PC laufen. Zu beachten ist bei den meisten Compilern die Unterstützung mehrerer Speichermodelle des 8086 (large, medium, small, compact), die einen Einfluß auf die Laufzeiteffizienz des erzeugten Programms haben.

**Tafel 8.3. Sprachen zur PC-Programmentwicklung**

| | |
|---|---|
| **MASM** | Makroassembler |
| **PASCAL** | allg. Programmentwicklung |
| **MODULA-2** | modulare Systemprogrammierung |
| **C** | hardwarenahe Systemprogrammierung |
| **COBOL** | kaufmännische Probleme |
| **FORTRAN** | wiss.-techn. Berechnungen |
| **LISP** | Probleme der künstlichen Intelligenz |
| **PROLOG** | logische Probleme der künstlichen Inteligenz |

Für die Assemblerprogrammierung wird im wesentlichen **MASM** von Microsoft eingesetzt. Einige andere Implementierungen (z.B. TurboEDITASM von Speedware) haben nur geringe Verbreitung gefunden. MASM wird im Abschn. 8.2. kurz beschrieben.

**PASCAL** liegt in verschiedenen Implementationen für MS-DOS vor. Das unter CP/M (8 und 16 Bit) sehr bekannte System PASCAL/MT+ hat durch Turbo-Pascal an Bedeutung verloren. Ebenso hat das umfangreiche System Microsoft-PASCAL Schwierigkeiten, sich gegen Turbo-Pascal zu behaupten. MS-PASCAL bietet jedoch eine Reihe von Vorzügen, wenn es um die Kompatibilität mit den anderen Sprachsystèmen von Microsoft geht. Da das UNIT-Konzept realisiert ist, können auch umfangreiche Systementwicklungen in MS-PASCAL realisiert werden. Eindeutiger Standard unter den PASCAL-Systemen für den PC ist Turbo-Pascal, das besonders durch die Neuerungen der Version 4.0 weiter an Bedeutung gewinnen wird. Turbo-Pascal wird im Abschn. 8.3. gesondert beschrieben.

**MODULA-2** wird bisher nicht als Programmiersystem von Microsoft angeboten. Es existieren jedoch leistungsfähige Entwicklungssysteme, wie z.B. M2SDS von Logitech, die auch auf anderen Rechnern zur Verfügung stehen.

Die Sprache **C** wird zunehmend als Systemimplementierungssprache auch auf PC eingesetzt. Im Abschn. 8.2. wird der C-Compiler von Microsoft kurz beschrieben. Turbo-C von Borland ist ein sehr schneller interaktiver C-Compiler, der die gleiche Bedienung wie Turbo-Pascal und Turbo-Basic hat.

**COBOL** wird besonders in den USA zur Programmierung kaufmännischer Probleme eingesetzt. Für die gleichen Zwecke werden in zunehmendem Maße auch Datenverwaltungspakete, wie z.B. dBASE, eingesetzt, die zusätzlich noch die Dateiverwaltung im interaktiven Modus unterstützen. Auf COBOL soll hier nicht weiter eingegangen werden.

**FORTRAN** als eine der ältesten Rechnersprachen ist immer noch weit verbreitet, obwohl wesentlich modernere Sprachen für die wiss.-techn. Berechnungen verfügbar sind. Die FORTRAN-Implementierung von Microsoft wird im Abschn. 8.2. kurz beschrieben.

**LISP** als typische Sprache der künstlichen Intelligenz ist

auch auf PC verwendbar. Auf Grund der geforderten Rechenleistung ist ein Einsatz nur auf PC der AT-Klasse sinnvoll. Eine Implementierung ist z.B. Golden COMMON LISP von Gold Hill mit sehr gut ausgebauter Programmierumgebung. Auf LISP wird hier nicht weiter eingegangen.

**PROLOG** wird zunehmend neben LISP als Sprache der künstlichen Intelligenz genutzt (z.B. für die Entwicklung von Expertssystemen). Leistungsfähige Implementierungen sind z.B. Arity PROLOG mit Entwicklungspaket, LPA microPROLOG und Turbo-PROLOG mit Toolbox von Borland. Auf PROLOG wird hier nicht weiter eingegangen.

### 8.2.2.1. MASM – Macroassembler

Der von Microsoft angebotene Makroassembler MASM ist das Standardwerkzeug für maschinennahe Programmierung. Alle BIOS- und DOS-Funktionen sind als Assembleraufruf definiert. Obwohl die Bedeutung der Assemblerprogrammierung abnimmt, weil die meisten Probleme auch in einer höheren Programmiersprache formuliert und effektiv bearbeitet werden können, ist doch immer noch in bestimmten Situationen die Erstellung kleinerer Module in MASM erforderlich.

**MASM - Makroassembler**

**Syntax** : MASM [src],[obj],[lst],[crf] /Optionen [;]

**Optionen :**

| | |
|---|---|
| src | Programm in Quelltext (.asm) |
| obj | Programm im Objektformat (.obj) |
| lst | Übersetzungsprotokoll (.lst) |
| crf | Querverweisliste (.crf) |
| /A | Segmetliste in alphabetischer Reihenfolge |
| /D | Fehlernachweis für Phase 1 und Phase 2 |
| /E | Code für Emulator (80x87) generieren |
| /R | Code für Coprozessor (80x87) generieren |
| /ML | Unterschied bei Groß-/Kleinschreibung |
| /MX | Kleinbuchstaben bleiben erhalten |
| /MU | Umsetzung Klein- nach Großbuchstaben |
| ... | weitere s. Dokumentation |

**Funktion** : Der Quelltext wird in Objektformat umgesetzt und wahlweise ein Protokoll und eine Querweisliste erstellt. Die Übersetzung kann durch Schalter gesteuert werden.

Die Parameterangaben beim Aufruf können auch fehlen. Der Assembler arbeitet dann im interaktiven Modus und erfragt die

erforderlichen Angaben. Wird die Kommandozeile mit einem ';' abgeschlossen, so werden die fehlenden Angaben durch Standardwerte ersetzt. Die Assemblersprache bietet neben den Prozessorbefehlen eine reiche Auswahl an Kommandos, die die Codegenerierung steuern. Zur Beschreibung dieser Eigenschaften muß auf die Dokumentation zu MASM verwiesen werden.

**Beispiel:** Programm hallo.asm zur Ausgabe von 'MASM - Hallo'

```
type hallo.asm                    (Quelltext anzeigen)
        name    hallo
stack   segment stack
        dw      20 dup(?)               ;Platz fuer stack
stack   ends
;
;       Main - Hauptprogramm
;
main    segment 'CODE'          ;Hauptprogramm
        assume  cs:main, ss:stack
start:  call    far ptr conout  ;Ausgabe auf Console
        mov     ah,4ch
        int     21h             ;Programmende
main    ends
;
;       Unterprogramm Conout - Ausgabe auf Console
;
outseg  segment 'CODE'
        assume  cs:outseg, ds:outseg
msg     db      'MASM - Hallo$'
conout  proc    far
        mov     ax,outseg
        mov     ds,ax           ;ds laden
        mov     dx,offset msg
        mov     ah,9            ;DOS-Funktion String-Ausgabe
        int     21h
        ret
conout  endp
outseg  ends
        end     start

MASM hallo,hallo;               (assemblieren)
LINK hallo                      (linken)
hallo                           (aufrufen)
```

### 8.2.2.2. C-Compiler (Microsoft)

Es existieren sehr viele C-Compiler für MS-DOS. Da Microsoft zur Programmentwicklung den eigenen optimierenden C-Compiler einsetzt, wird er besonders gut gepflegt und kann als Standard für andere Systeme angesehen werden. Die Versionen werden zunehmend komplexer und umfangreicher. Durch geeignete Zusatzprogramme und gut ausgebaute Bibliotheken bleibt das C-System jedoch handhabbar und liefert einen guten Programmcode. Ab Version 5.0 wird ein sehr komfortables Entwicklungssystem QuickC mitgeliefert, das sich an den Eigenschaften des Turbo-C (Borland) orientiert, durch

die Kompatibilität zum "richtigen" C-Compiler von Microsoft jedoch wesentliche Vorzüge aufweist. Für den C-Programmierer bietet das C-System von Microsoft eine sehr gute Basis zur Programmentwicklung. Der C-Compiler ist ein Mehrpaßcompiler. Das Aufrufen der einzelnen Komponenten und des Linkers wird durch ein Treiberprogramm CL organisiert. Die Programmtestung auf Quelltextebene erfolgt mit dem Fehlersuchprogramm CodeView.

**C-Compiler (Microsoft)**

**Syntax** : CL [/Optionen src]... /link lib /Linkopt

**Optionen :**

| | |
|---|---|
| /An | Speichermodell (n=M,L,H Standard /AL) |
| /Fn datei | Dateisteuerung (vgl. /help) |
| /FPn | Code für Coprozessor (Standard /FPi87) |
| /Gn | Steuerung der Codegenerierung |
| /I dir | Angabe von INCLUDE-Verzeichnissen |
| /c | kein automatischer LINK-Aufruf |
| ... | diverse andere Schalter (vgl. Dokumentation) |
| /help | Anzeige möglicher Schalter |
| /link | Kennung für nachfolgende LINK-Optionen |
| src | Name des Quellprogramms (name.c) |
| lib | LINK-Bibliotheken |
| /Linkopt | LINK-Optionen (vgl. Programm LINK) |

**Funktion** : CL arbeitet als Steuerprogramm. Nach der Parameteranalyse werden intern die einzelnen Compilerpässe gestartet und der Linker aufgerufen. Für die normale Arbeit sind alle Schalter auf sinnvolle Standardwerte gesetzt.

Der Compiler wird mit einem Programm SETUP installiert, d.h., es werden die Bibliotheken angepaßt, das Speichermodell ausgewählt, die Unterstützung des Coprozessors eingestellt und das C-System in den entsprechenden Verzeichnissen eingerichtet. Zum Editieren sollte man den interaktiven Compiler QuickC verwenden, der mit dem optimierenden C-Compiler kompatibel ist. Mit QuickC kann man vollständig interaktiv arbeiten. Die Arbeit mit der Maus wird effektiv unterstützt. Umfangreiche Hilfefenster ersparen häufiges Nachschlagen in der Dokumentation. Die Arbeit mit Bibliotheken ist in QuickC ebenfalls möglich. Weitere Informationen dazu sind in der umfangreichen Dokumentation enthalten. Der automatische Aufruf des Linkers kann mit dem Schalter /c unterbunden werden. Der Linker kann dann separat aufgerufen werden, um die gewünschte Programmstruktur zu erzeugen.

Die Arbeit mit CL wird erleichtert, wenn man die Möglichkeit von Umgebungsvariablen für Dateien ausnutzt. Standardmäßig wird das aktuelle Verzeichnis verwendet.

Folgende Namen sind definiert:

```
PATH      Compilerdateien und Dienstprogramme
INCLUDE   INCLUDE-Verzeichnis
TMP       temporäre Dateien
LIB       Bibliotheken für LINK.
```

**Beispiel:** Programm hallo.c zur Ausgabe von C - Hallo

```
type hallo.c              (Quelltext anzeigen)
     #include <stdio.h>
     main()
     {
     printf("C - Hallo\n");
     }

CL hallo.c                (compilieren, linken)
hallo                     (aufrufen)
```

### 8.2.2.3. FORTRAN-Compiler (Microsoft)

Für FORTRAN stehen ebenfalls eine Reihe von Compilern zur Verfügung (z.B. F77 von Digital Research, Professional FORTRAN von IBM, MS-FORTRAN von Microsoft). Hier soll auf den optimierenden FORTRAN-Compiler von Microsoft eingegangen werden. Der Sprachumfang (Version 4.0) enthält den Standard FORTRAN-77 und einige Erweiterungen. Die Verträglichkeit mit den Compilern für andere Sprachsysteme von Microsoft (PASCAL, C, MASM) ist sehr gut, so daß die Anwendung des MS-FORTRAN einige Vorteile gegenüber anderen FORTRAN-Compilern hat. Die Nutzung des Mehrpaßcompilers wird (ähnlich wie bei C) durch ein Treiberprogramm FL unterstützt. Auch die Schalter sind weitgehend ähnlich.

Der Compiler wird mit einem Programm SETUP installiert, d.h., es werden die Bibliotheken angepaßt, das Speichermodell ausgewählt, die Unterstützung des Coprozessors eingestellt und das FORTRAN-System in den entsprechenden Verzeichnissen eingerichtet. MS-FORTRAN unterstützt den symbolischen Debugger Code-View, mit dem Testung und Fehlersuche auf Quelltextniveau möglich sind. Weitere Informationen dazu sind in der umfangreichen Dokumentation enthalten. Der automatische Aufruf des Linkers kann mit dem Schalter /c unterbunden werden. Der Linker kann dann separat aufgerufen werden, um die gewünschte Programmstruktur zu erzeugen.

Die Arbeit mit FL wird erleichtert, wenn man die Möglichkeit von Umgebungsvariablen für Dateien ausnutzt. Standardmäßig wird das aktuelle Verzeichnis verwendet. Folgende Namen sind definiert:

```
PATH      Compilerdateien und Dienstprogramme
INCLUDE   INCLUDE-Verzeichnis
TMP       temporäre Dateien
LIB       Bibliotheken für LINK.
```

**FORTRAN-Compiler (Microsoft)**

```
Syntax   : FL [/Optionen src]...  /link lib /Linkopt

Optionen :
/4..        diverse Schalter (Standardwerte aktiv)
            z.B. /4I4 heißt INTEGER*4
/An         Speichermodell (n=M,L,H Standard /AL)
/Fn datei   Dateisteuerung (vgl. /help)          )
/FPn        Code für Coprozessor (Standard /FPi87)
/Gn         Steuerung der Codegenerierung
/I dir      Angabe von INCLUDE-Verzeichnissen
/c          kein automatischer LINK-Aufruf
...         diverse andere Schalter (vgl. Dokumentation)
/help       Anzeige möglicher Schalter
/link       Kennung für nachfolgende LINK-Optionen
src         Name des Quellprogramms (name.for)
lib         LINK-Bibliotheken
/Linkopt    LINK-Optionen (vgl. Programm LINK)

Funktion : FL arbeitet als Steuerprogramm. Nach der Pa-
           rameteranalyse werden intern die einzelnen
           Compilerpässe gestartet und der Linker auf-
           gerufen. Für die normale Arbeit sind alle
           Schalter auf sinnvolle Standardwerte ge-
           setzt.
```

**Beispiel:** Programm hallo.for (Ausgabe von FORTRAN - Hallo)

```
type hallo.for                (Quelltext anzeigen)

program hallo
write(*,*) 'FORTRAN - Hallo'
end

FL hallo.for                  (compilieren, linken)
hallo                         (aufrufen)
```

### 8.2.3. LIB – Bibliotheksverwaltung

Alle MS-Compiler (weitgehend auch die Compiler anderer Softwarehäuser) nutzen zur Ausgabe das Intel-Objektformat, das durch eine Folge von Sätzen mit festgelegten Satztypen gekennzeichnet ist. Auf die speziellen Satztypen soll hier nicht näher eingegangen werden, da man bei der Programmierung damit nicht unmittelbar

zu tun hat. Es enthält alle für den Linker erforderlichen Informationen, um Programme aus mehreren Modulen aufzubauen. Durch den Bibilotheksverwalter **LIB** können solche Module in Bibliotheken zusammengefaßt werden. Alle vorgefertigten Bausteine der Compilersysteme sind in Bibliotheken abgelegt und stehen meistens nur im Objektformat zur Verfügung. Die Nutzung des Programms LIB ist relativ einfach und erfordert daher keine besonderen Erläuterungen. Wie die anderen Programmierwerkzeuge kennt auch LIB einen interaktiven Modus, in dem die erforderlichen Informationen abgefragt werden, und einen Modus, bei dem alle Informationen in der Kommandozeile stehen. Bei umfangreichen Bibliotheken kann die Zeit zur Bibliotheksverwaltung (Umkopieren, Querverweisliste) relativ lang werden.

**LIB - Verwaltung von Objektmodulbibliotheken**

**Syntax** : LIB (interaktiver Modus)
LIB @antwort (Antwortdatei)
LIB [lib1] /Option [op],[list],[lib2][;]

**Optionen :**

| | |
|---|---|
| @antwort | Antworten stehen in der Datei <antwort> |
| lib1 | Modulbibliothek (Eingabe) |
| lib2 | Modulbibliothek (Ausgabe) |
| /Pagesize:n | Seitenlänge der Bibliothek (pagesize) |
| list | Inhaltsverzeichnis (list file) |
| op | Kommandofolge |

**Funktion** : Die Eingabemodulbibliothek <lib1> wird eröffnet und daraus die neue Modulbibliothek <lib2> erzeugt. Beim Kopieren werden die Kommandos berücksichtigt und die entsprechenden Module behandelt. Falls die Datei noch nicht existiert, wird eine neue angelegt. Das Inhaltsverzeichnis enthält Angaben zu enthaltenen Modulen und externen Refeferenzen (cross referenz). Fortsetzungszeilen können mit '&' gebildet werden. Wird die Eingabe mit ';' abgeschlossen, so werden für alle fehlenden Angaben Standardwerte angenommen.

**Kommandos:**

| | | |
|---|---|---|
| +mod | Modul anfügen | (append) |
| -mod | Modul löschen | (delete) |
| *mod | Objektmodul <mod>.obj kopieren | (copy) |
| -+mod | Modul ersetzen | (replace) |
| -*mod | Modul kopieren und löschen | (move) |

Wenn die neue Bibliothek mit dem gleichen Namen angelegt wird, so wird die alte Bibliothek umbenannt und bleibt erhalten (Dateierweiterung .BAK). Erfolgt keine Veränderung der Bibliothek, so wird auch keine neue Bibliothek angelegt.

**Beispiele:** LIB pascal,pascal.lst (Inhaltsverzeichnis)
LIB privat +neu1+neu2 (Hinzufügen)

### 8.2.4. LINK – Programmverbinder (Linker)

Zu MASM und den Compilersystemen von Microsoft wird der Overlay-Linker **LINK** bereitgestellt, der die Verbindung von separat übersetzten Modulen realisiert. Der Verbindungsprozeß ist auf Grund der Struktur des 8086/80286 etwas kompliziert. Zur Darstellung der Konzepte ist hier nicht genügend Platz. Die Anordnung der Segmente kann durch Parameter weitgehend gesteuert werden. Von den Compilern werden die wichtigsten Angaben vorbereitet. Zum Verständnis der Arbeitsweise reicht es meistens aus, sich auf die einzubindenden Module, die verwendeten Bibliotheken und die genutzten Verzeichnisse zu konzentrieren. Eine ausreichende Übersicht zur Segmentanordnung gibt die Speicherliste (MAP-Datei). Vom Linker wird eine temporäre Datei angelegt, wenn der interne Arbeitsspeicher nicht ausreicht. Von LINK kann nur das EXE-Format erzeugt werden. Eine weitere Behandlung der fertigen Programme ist durch Zusatzwerkzeuge (vgl. Abschn. 8.2.6.) möglich.

**LINK - Programmverbinder**

**Syntax** : LINK (interaktiver Modus)
LINK @antwort (Antwortdatei)
LINK [obj],[exe],[map],[lib] /Optionen[;]

**Optionen :**

| | |
|---|---|
| @antwort | Antworten stehen in Datei <antwort> |
| obj | Liste von Objektmodulen (.obj) |
| exe | Name der EXE-Datei (Ausgabe) |
| map | MAP-Datei (Speicherzuordnungsliste, Symbole) |
| lib | Bibliotheken (.lib) |
| /COdeview | EXE-Datei enthält Informationen für CodeView |
| /DOsseg | Segmentsortierung nach DOS-Standard |
| /Informati | Ausgabe Modulname beim Linken |
| /Linenumbe | Zeilennummern aus Quelldatei einfügen |
| /Map | MAP-Datei enthält globale Symbole |
| /NODefault | keine Standardbibliothekssuche |
| /Pause | Pause bevor EXE-Datei ausgegeben wird |
| /SEgment:n | Anzahl Segmente (Standard n=128) |
| /STack:n | Stackgröße in Byte |
| ... | diverse andere Schalter (vgl. Dokumentation) |
| /HElp | Ausgabe der möglichen /Optionen |

**Funktion** : Das LINK-Programm bindet mehrere Objektmodule zu einem ausführbaren Programm. Module müssen im MS-Objektformat vorliegen.

Werden mehrere Module oder Bibliotheken angegeben, so sind die Namen durch Leerzeichen oder '+' zu trennen. Wird LINK ohne Parameter aufgerufen, so fragt das Programm nach den Dateinamen. Die Antworten können auch in einer Datei stehen, die als @datei angegeben wird. Steht nach den Parametern ein Semikolon, werden alle fehlenden Parameter durch Standardwerte ersetzt. Für die Suche nach Bibliotheken gilt eine Standardreihenfolge, die auch abgeschaltet werden kann (/NOD). Die Compiler können Suchpfade im Objektmodul eintragen (z.B. für die Compilerbibliotheken). Es gilt folgende Reihenfolge:

- aktuelles Laufwerk und Verzeichnis
- Bibliotheken aus Kommandozeile
- Bibliotheken, die mit LIB-Umgebungsvariablen angegeben.

**Beispiele:**
```
LINK /help                          (Anzeige /Optionen)
LINK test;                          (Standardparameter)
LINK                                (interaktiver Modus)
LINK prog/map;                      (Symbolverzeichnis)
LINK mod1+mod2;                     (mehrere Module)
LINK mod,mod,mod,\lib\slibc+\lib\paslib
                                    (vollständige Angabe)
```

### 8.2.5. Fehlersuchprogramme – DEBUG, CodeView

Wer keine Fehler macht, braucht auch kein Fehlersuchprogramm. Manchmal hat man aber doch hartnäckige Fehler, die man nur mit großem Aufwand findet. Für diese Zwecke wurden spezielle Fehlersuchprogramme geschaffen, mit denen man den Maschinencode (Programm DEBUG) oder auch den Ablauf im Quellprogramm (SYMDEB, CodeView) verfolgen kann. Die wesentliche Funktion eines Fehlersuchprogramms besteht darin, das zu testende Programm unter der Aufsicht des Debuggers ablaufen zu lassen. Dazu können Haltepunkte gesetzt werden, Variablen überprüft und gesetzt und das Programm modifiziert werden. Während mit dem Programm DEBUG von Microsoft nur eine Fehlersuche auf Maschinencodeebene möglich ist, die für Programmierer einer Hochsprache eine "Zumutung" ist, gestatten die Programme SYMDEB und CodeView eine gezielte Fehlersuche auf Quelltextebene. Alle Variablen sind mit ihren im Programm verwendeten Namen ansprechbar. Da DEBUG immer noch als Standardwerkzeug mit MS-DOS ausgeliefert wird, werden hier die Kommandos kurz aufgeführt. Der neue Debugger Codeview enthält alle Funktionen des symbolischen Debuggers SYMDEB. Hier werden darum nur noch die Kommandos von CodeView aufgeführt.

Mit DEBUG können nicht nur Programme, sondern auch andere Dateien gelesen und verändert werden (z.B. ist das Lesen eines Sektors direkt möglich, ohne das Dateisystem zu nutzen). Eine

effektive Anwendung von DEBUG setzt einige Kenntnisse der Assemblerprogrammierung voraus.

**DEBUG - Standarddebugger**

```
Syntax   : DEBUG [d:][pfad][datei] [parm]

Optionen :
d:         Laufwerk
pfad       Verzeichnis
datei      Datei, die geladen werden soll
parm       Parameter für <datei>
Funktion : DEBUG wird geladen und lädt wiederum die
           angegebene Datei. Im interaktiven Modus wer-
           dann die folgenden Kommandos akzeptiert.

Kommandos:
A [adr]     Assemblieren ab <adr>              (assemble)
C ber adr   Vergleich Bereich <ber> mit <adr>  (compare)
D [ber]     Ausgabe Bereich <ber>                 (dump)
E adr [lst] Dateneingabe (hex oder string)       (enter)
F ber lst   Bereich füllen                        (fill)
G [=adr [adr..]] Starten und Unterbrechung          (go)
H num num   hexadezimale Arithmetik                (hex)
I port      Port lesen                           (input)
L [adr[d:rec rec]] Programm oder Sektor laden     (load)
M ber adr   Daten kopieren von <ber> nach <adr>   (move)
N datei parm Dateinamen mit Parametern angeben    (name)
O port dat  Port ausgeben                       (output)
Q           DEBUG beenden                         (quit)
R [regs]    Register ausgeben                 (register)
S ber lst   Suchen                             (search)
T [=adr][num] Befehlsverfolgung                  (trace)
U [ber]     Disassemblieren von <ber>        (unassemble)
W [adr[d:rec rec]] Programm/Sektor schreiben     (write)
```

Die Arbeit mit DEBUG kann recht mühsam werden. Es existieren weitere Debugger, die die gleichen Funktionen erledigen, jedoch einfacher in der Handhabung sind. Die Funktionen zur Änderung von Sektoren werden besser mit Dienstprogrammen für Datenträger erledigt (z.B. PCTOOLS von Central Point Software oder den Norton Utilities).

Zur symbolischen Fehlersuche ist besser der symbolische Debugger CodeView geeignet. Voraussetzung für eine effektive Anwendung ist dabei jedoch die Verfügbarkeit von Quelltexten und Symboltabellen. Die Compiler von Microsoft sind für die Anwendung des CdeView geeignet.

| CODEVIEW - symbolischer Debugger | |
|---|---|
| **Syntax** | : CV /Optionen exedatei [parm] |
| **Optionen** : | |
| /B | CGA in BW-Modus |
| /Ckommand | Kommandofolge <kommand> beim Start ausführen |
| /F | Bildschirmumschaltung (flipping video pages) |
| /M | keine Mausunterstützung |
| /T | Start im sequentiellen Modus |
| /S | Bildschirmumschaltung (saving buffer) |
| /W | Start in Windowmodus |
| /43 | EGA-Modus mit 43 Zeilen |
| exedatei | Programm, das getestet werden soll |
| parm | Parameter für <exedatei> |
| **Funktion** : | CodeView lädt das angegebene Programm und nimmt dann interaktiv eine Vielzahl von Kommandos entgegen. Durch umfangreiche HELP-Information ist eine gute Unterstützung am Bildschirm gesichert. |

Die interaktiven Kommandos können hier nicht im einzelnen beschrieben werden (vgl. Dokumentation oder HELP im interaktiven Windowmodus).

### 8.2.6. Programmverwaltung (MAKE, EXE2BIN, EXEMOD)

Von Microsoft werden einige Werkzeuge zur Programmverwaltung angeboten, die in diesem Abschnitt kurz beschrieben werden sollen. Das Programm **MAKE** ist ein Werkzeug zur Verwaltung von größeren Projekten, die aus vielen Modulen, Programmen und Bibliotheken bestehen.

| MAKE - Projektverwaltung | |
|---|---|
| **Syntax** | : MAKE /Optionen [makro] datei |
| **Optionen** : | |
| /D | Bearbeitungsstand für Dateien ausgeben |
| /I | Returncode von Programmen nicht beachten |
| /N | Kommandoanzeige, keine Ausführung |
| /S | kein Protokoll (silent mode) |
| makro | Makrodefinition (name=wert) |
| datei | MAKE-Datei mit Abhängigkeiten und Kommandos |
| **Funktion** : | Prüfung von Abhängigkeiten zwischen Dateien. Bei Versionsunterschieden werden die angegebenen Kommandos ausgeführt. |

Anhand des Datums wird kontrolliert, ob für eine Verwaltungseinheit alle Bestandteile auf dem aktuellen Stand sind. In einer MAKE-Datei wird angegeben, wie die Verwaltungseinheiten auf den aktuellen Stand gebracht werden. Durch MAKE werden anhand dieser Datei alle notwendigen Arbeitsschritte durchgeführt. Zur Vereinfachung und Einsparung von Schreibaufwand können Makros definiert werden, die innerhalb der MAKE-Datei mit $(name) aufgerufen werden.

Eine MAKE-Datei hat folgenden Aufbau (die Folgen können mehrfach auftreten):

```
ausgabe:eingabe[,eingabe...] #kommentar
#kommentar
      kommando  #kommentar
     [kommando] #kommentar

spezielle Makros $*   (Hauptname der Ausgabedatei)
                 $@   (Name der Ausgabedatei)
                 $**  (Liste der Eingabedateien)
```

Das Programm **EXE2BIN** wandelt das EXE-Format in das kompaktere COM-Format um. Die Umwandlung ist nicht in allen Fällen möglich und auch nur für sehr häufig benutzte Programme zu empfehlen. Im Abschn. 5.6. sind weitere Informationen zu den Programmformaten zu finden.

| **EXE2BIN - Umwandlung von Programmformaten** |
|---|
| **Syntax** : EXE2BIN datei1 [datei2] |
| **Funktion** : Die Datei <datei1> (.exe) wird in <datei2> umgewandelt. |

In der Regel können nur mit MASM erstellte Programme in das COM-Format umgeformt werden. Ist <datei2> nicht angegeben, so wird der gleiche Hauptname wie bei <datei1> verwendet und die Erweiterung .COM angefügt.

Die Programme **EXEMOD** und **EXEPACK** verwalten den EXE-Vorspann bzw. den EXE-Programmteil. Die Funktion EXEPACK (Aufruf: EXEPACK <eingabe> <ausgabe>) kann auch mit dem Linker (/EXEPACK) durchgeführt werden. Es ist dabei nur zu beachten, daß gepackte EXE-Dateien nicht mehr mit CodeView bearbeitet werden können. Mit EXEMOD werden einige Felder im Vorspann auf neue Werte gesetzt. Außerdem können die aktuellen Werte angezeigt werden.

| **EXEMOD - Vorspann einer EXE-Datei verändern** |
|---|
| **Syntax** : EXEMOD exe /Optionen<br><br>**Optionen** :<br>exe EXE-Datei<br>/H Vorspann anzeigen<br>/STACK hex Stackwert verändern (SP)<br>/MIN hex minimale Paragraphenzuweisung (MINALLOC)<br>/MAX hex maximale Paragraphenzuweisung (MAXALLOC)<br><br>**Funktion** : Die mit Optionen angegebenen Felder werden verändert. |

## 8.3. Turbo-Pascal

Der Compiler Turbo-Pascal der Firma Borland stellt wohl die am meisten genutzte Programmierumgebung für PC dar. Der Sprachumfang der Version 3.0, der sowohl auf 8-Bit-Rechnern als auch auf 16-Bit-Rechnern verfügbar ist, ist eine Art Industriestandard, der sich am ANSI-Standard orientiert, jedoch einige Abweichungen und Erweiterungen aufweist. Mit der Version 4.0, die 1987 auf den Markt kam, wurde eine Reihe von Erweiterungen eingeführt, die Turbo-Pascal noch attraktiver machen. Zum Sprachumfang der Version 3.0 sei auf die Literatur verwiesen. Zur Version 4.0 wird ein exzellentes Handbuch geliefert, das alle Eigenschaften von Turbo-Pascal ausführlich beschreibt und die Handhabung erläutert. In Version 4.0 sind folgende Neuerungen enthalten:

- einheitliche Benutzeroberfläche mit anderen Compilern von Borland (Turbo-C, Turbo-Prolog, Turbo-Basic)
- ausgebautes HILFE-System
- verbesserte Editorfunktionen
- Compilierung mit 2- bis 3facher Geschwindigkeit (27000 Zeilen je Minute auf PC/AT mit 8 MHz)
- optimierter Code (kompakt und schnell)
- eingebauter "intelligenter" Linker (externe OBJ-Module sind möglich, die z.B. mit MASM oder Turbo C erstellt wurden)
- EXE-Format, keine Begrenzung mehr auf 64 KByte
- modulare Programmentwicklung durch UNIT-Konzept (separat übersetzte Module)
- Projektverwaltung mit MAKE
- Standard-UNITS: System, DOS, Graph, CRT, Printer
- INCLUDE-Dateien bis zu acht Ebenen
- neue Datentypen (Integer: LongInt, ShortInt, Word
  Real : Simple, Double, Extended, Comp)
- neue Standardprozeduren INC, DEC

- Kompatibilität zum ANSI-Standard
- Unterstützung der Coprozessoren 80x87
- bedingte Compilierung
- Zusatzprogramme zum Übergang von Version 3.0 nach 4.0
- Compiler in 2 Versionen: integrierte Entwicklungsumgebung und Kommandozeilenversion
- diverse Zusatzprogramme (TPUMOVER, GREP, TOUCH, TINST, BINOBJ).

Die interaktive Version wird mit "turbo" gestartet und steht dann nach dem Laden eines Pascalprogramms mit einem **Editor** (Kommandos sind weitgehend kompatibel zu WordStar) zur Verfügung. Das **HILFE-System** wird über die Funktionstaste <F1> aufgerufen. Neben ausführlichen Kommandoübersichten wird auch eine Hilfe zur Syntax von Pascal angeboten (Tasten <Ctrl-F1>). Die HILFE-Fenster werden bis zu einer Tiefe von 20 aufgehoben und können mit <Alt-F1> zurückgeholt werden. Das Menüsystem ist selbsterklärend. Beim Start sucht Turbo nach einer Konfigurationsdatei (TURBO.TP), in der Standardwerte für die einzelnen Optionen angegeben werden können (vgl. Kommandozeilenversion TPC). Die Arbeit mit einer Maus wird nicht unterstützt. Am unteren Bildschirmrand wird eine Zeile mit möglichen Funktionstasten angezeigt. Alle Menüpunkte können auch durch Angabe des ersten Buchstaben erreicht werden (bei der Hauptmenüleiste am oberen Rand ist ein <Alt> voranzustellen). Interessanteste Neuerung ist das **UNIT-Konzept**, das eine Überschreitung der 64-KByte-Grenze gestattet. Die Units (Module) werden separat in eine spezielle OBJ-Datei übersetzt (TPU-Format), die in das normale OBJ-Format von DOS mit dem Programm BIN2OBJ umgewandelt werden kann. OBJ-Dateien können mit der Direktive {$L datei} in ein Programm eingebunden werden. Dadurch sind auch längere MASM-Programme leicht in ein Pascalprogramm zu integrieren. Zu Turbo-Pascal gehören mehrere Standardunits, die in einer Bibliothek TURBO.TPL gespeichert sind. Die Bibliothek kann mit dem Dienstprogramm TPUMOVER verwaltet werden. Eigene Units können entweder separat in einem Unitverzeichnis stehen oder mit in die Standardbibliothek aufgenommen werden. Die Beispielroutinen im Abschn. 11. zum Aufruf von BIOS- und DOS-Funktionen sind z.T. als Unit geschrieben. Der Vorteil einer Unit gegenüber einer INCLUDE-Variante besteht darin, daß die ausgetesteten Programmteile nicht immer wieder übersetzt werden müssen. Außerdem ist man nicht gezwungen, Programmbausteine in Quelltextform weiterzugeben. Zur **Projektverwaltung** wird ein eigenes Programm MAKE angeboten, das weitgehend wie MAKE von Microsoft arbeitet. Durch Auswertung des Datums von verwendeten Units kann eine weitgehende Automatisierung der Übersetzung aller beteiligten Bestandteile eines Programms erreicht werden. Zur **Fehlersuche** werden die Informationen zu definierten Symbolen in einer speziellen TPM-Datei abgelegt, die mit dem Programm TPMAP in die MAP-Datei umgewandelt werden. Die Datei wird dann zusammen

mit den Programmen MAPSYM und SYMDEB von Microsoft zur Fehlersuche eingesetzt. Dadurch ist auch für Turbo-Pascal ein Debugger verwendbar. Durch Zusatzpakete (z.B. von Lauer&Wallwitz) wird wie bei der Version 3.0 außerdem auch ein Turbo-Debugger bereitgestellt.

Die **Kommandozeilenversion** wird mit TPC aufgerufen. Alle im interaktiven Modus einstellbaren Menüpunkte lassen sich hier durch Optionen auswählen.

**TPC - Kommandozeilenversion von Turbo-Pascal**

**Syntax** : TPC dateiname /Optionen

**Optionen :**

| | |
|---|---|
| dateiname | Hauptname des Übersetzungsmoduls |
| /$B+ /$B- | Options/Compiler/Boolean evaluation (On/Off) |
| /$D+ /$D- | Options/Compiler/Debug information (On/Off) |
| /$F+ /$F- | Options/Compiler/Force far calls (On/Off) |
| /$I+ /$I- | Options/Compiler/I/O checking (On/Off) |
| /$L+ /$L- | Options/Compiler/Link buffer (Memory/Disk) |
| /$Ms,m,n | Options/Compiler/Memory size (stack,min,max) |
| /$N+ /$N- | Options/Compiler/Numeric processing (HW/SW) |
| /$R+ /$R- | Options/Compiler/Range checking (On/Off) |
| /$S+ /$S- | Options/Compiler/Stack checking (On/Off) |
| /$T+ /$T- | Options/Compiler/Turbo Pascal map (On/Off) |
| /$V+ /$V- | Options/Compiler/string check (strict/relax) |
| /B | Compile/Build |
| /Fseg:ofs | Compile/Find error |
| /M | Compile/Make |
| /Q | (Quiet mode) |
| /Rparam | Parameter für Run (Memory) |
| /Xparam | Parameter für Run (Disk) |
| /Ddefine | Options/Compiler/Conditional defines |
| /Epfad | Options/Directories/Executable directory |
| /Opfad | Options/Directories/Object directories |
| /Tpfad | Options/Directories/Turbo directory |
| /Upfad | Options/Directories/Unit directories |

**Funktion** : Die angegebene Datei wird übersetzt. Ausgabe ist entweder ein Programm oder eine Unit. Die Übersetzung wird durch Optionen gesteuert.

Die Optionen können auch durch ein Minuszeichen gekennzeichnet werden. Mehrere Optionen werden durch Komma getrennt. Beim Start sucht TPC nach einer Konfigurationsdatei (TPC.CFG).

**Beispiel:** tpc biosbsp.pas /$T+,$U\unitdir

Zu Turbo-Pascal gibt es eine Vielzahl von Toolboxen, die hier nur in einer Auswahl aufgezählt werden können (Tafel 8.4).

**Tafel 8.4. Toolboxen für Turbo-Pascal**

| Toolboxen für Turbo-Pascal |
|---|
| **Turbo Tutor** (Borland)<br>Unterstützung der Einarbeitung, Beispielprogramme |
| **Turbo Database Toolbox** (Borland)<br>Routinen für indexsequentielle Datenbank (B-Tree), Sortierprogramme |
| **Turbo Numerical Methods (Mathe) Toolbox** (Borland)<br>Routinensammlung für mathematische Grundfunktionen (Lösung linearer Gleichungssysteme, Interpolation, Differentation, Integration, Matrizen, Fourier-Transformation u.a.) |
| **Turbo Editor** (Borland)<br>leistungsfähiger Editor mit allen Routinen im Quelltext (Routinen zur Menügestaltung, Windowtechnik, Textverarbeitung, Ansteuerung des Bildschirmspeichers, Spooler, Uhr u.a.) |
| **Turbo Graphix Toolbox** (Borland)<br>Graphikpaket für alle Standardbildschirmadapter (graphische Grundfunktionen, Verwendung von Weltkoordinaten, Graphikfenster, Präsentationsgraphik, Kurvendarstellungen) |
| **Turbo-Machine** (Lauer&Wallwitz)<br>DOS-Funktionen, Interruptbehandlung, Windows, Menüs, EMS-Behandlung, Tastaturmakros u.a. |
| **Turbo Lader/Complex** (Lauer&Wallwitz)<br>Routinensammlung mit DOS-Funktionen, umfangreichen mathematischen Verfahren, Menüunterstützung u.a. |
| **Turbo Talk** (Lauer&Wallwitz)<br>ausgebaute Unterstützung für serielle Schnittstellen (gepufferte Datenübertragung, Interruptssteuerung) |
| **Turbo Power Tools** (Blaise, Turbo Power Software)<br>Routinensammlung, ähnlich Turbo-Lader |
| **TurboWINDOW/Pascal** (Metagraphics)<br>Windowmanagement unter Turbo-Pascal |

Tafel 8.4 enthält nur eine kleine Auswahl der zu Turbo-Pascal passenden Softwarepakete. Das Angebot wird sicher mit der neuen Version 4.0 noch zunehmen. Die Toolboxen der Version 3.0 wurden als Includedateien genutzt, wobei der Quelltext immer mit übersetzt wurde. Der Umfang der Pakete war dadurch beschränkt. Durch Version 4.0 ist die Nutzung der Toolboxen in Form separat übersetzter Units (Module) möglich. Die Programmierumgebung verbessert sich dadurch erheblich.

# 9. Alternative Benutzerschnittstellen

Der Kommandointerpreter COMMAND realisiert eine einfache zeichenorientierte Benutzerschnittstelle. Alle Kommandos müssen als Text eingegeben werden. Neben dieser einfachen Form existieren weitere Möglichkeiten, dem System Kommandos zu übermitteln. Die Einführung von Bildschirmen mit Graphikfähigkeit machte auch die Nutzung von leistungsfähigen Windowsystemen möglich, die zunehmend an Bedeutung gewinnen. Im Zusammenhang mit einer Maus ist die Auswahl von Punkten eines Menüs sehr einfach möglich. Auch ohne eine Maus kann mit den Cursorsteuertasten leicht eine Auswahl getroffen werden. Weite Verbreitung haben die Windowsysteme MS-Windows von Microsoft und GEM von Digital Research gefunden. Das von IBM angebotene System TopView bietet einen ähnlichen Funktionsumfang, ist jedoch nicht so weit verbreitet. Das System MS-Windows wird mit dem neuen IBM-System PS/2 in etwas modifizierter Form als Presentation Manager die alternative Benutzerschnittstelle für PC-Systeme werden. Im Abschn. 9.1. werden einige Hinweise zur Nutzung des MS-Windows gegeben. Im Abschn. 9.2. folgen ebensolche Hinweise zu GEM, und im Abschn. 9.3. werden einige einfachere Menüsysteme charakterisiert.

## 9.1. MS-Windows

WINDOWS von Microsoft ist als echte Ergänzung zum MS-DOS konzipiert. Durch die Möglichkeit, mehrere Anwendungsprogramme (Applikationen) gleichzeitig in den Speicher zu laden und ablaufen zu lassen, kann man eine Integration unterschiedlicher Komponenten unter einem einheitlichen Nutzerinterface erreichen. Der Aufwand zu dieser Integration ist natürlich wesentlich größer als bei echten integrierten Paketen. Bei ausreichender Speicherkapazität und Rechenleistung spielt das aber keine große Rolle. Durch WINDOWS erhält der PC begrenzte Möglichkeiten zum Multitasking. Durch den wesentlich höheren Organisationsaufwand gegenüber COMMAND ist ein sinnvolles Arbeiten eigentlich erst bei PC/XT mit einer höheren Taktfrequenz bzw. PC/AT sinnvoll. Wesentliche Bestandteile von WINDOWS sind

- **KERNEL** **Kern für Multitasking (Prozeßverwaltung)**
- **GDI** **Graphikschnittstelle (Graphics Device Interface)**
- **Desktop** **MS-DOS Executive**
- **Utilities** **WINDOWS-Applikationen (Spooler, Notizblock, Karteikasten, Terminalemulator, Ablage, Uhr, Rechner, Kalender, Reversi, WRITE, PAINT)**
- **PIF-Editor** **Editor für PIF-Dateien (program information file).**

Neben diesen Hauptbestandteilen gibt es eine ganze Anzahl von Programmen von Microsoft, die für die Arbeit mit WINDOWS vorbereitet sind. Durch die PIF-Dateien werden die Programmanforderungen (Programmname, Titel, Parameter, Verzeichnis, Speicherplatz, Bildschirmansteuerung u.a.) spezifiziert. Fehlende Angaben werden durch Standardwerte ersetzt. Durch die PIF-Datei werden alle Programme für WINDOWS geeignet. WINDOWS muß installiert werden mit dem Programm SETUP, das die speziellen Anforderungen für den konkreten Einsatz einstellt. Der Aufruf erfolgt mit WIN aus dem WINDOWS-Verzeichnis. Es meldet sich die Komponente Desktop mit der MS-DOS-Executive. Von hier aus lassen sich beliebige Programme starten oder Utilities aufrufen. Die Bedienung erfolgt entweder über die Tastatur oder mit der Maus. Die Mausbedienung ist wesentlich einfacher und dem Fenstersystem angepaßt. Für die Tastaturbedienung sind die Tasten <Return>, <Tab>, <Alt-Leertaste>, <Ctrl-d> (d=Laufwerk), <Leertaste> und die Cursortasten wichtig. Die Bedienung erfordert etwas Eingewöhnung. Viele Parameter von WINDOWS lassen sich interaktiv einstellen und dem persönlichen Bedarf anpassen. Der Datenaustausch zwischen verschiedenen Programmen ist problemlos möglich.

Die Fenstergröße einer Anwendung ist frei wählbar. In Version 1 dürfen Fenster sich nicht überlappen; dadurch ist die Anzahl gleichzeitig sichtbarer Fenster beschränkt. WINDOWS nutzt Speicherplatz oberhalb 640 KByte effektiv aus. Die Anpassung an verschieden Graphikkarten und Drucker wird durch Treiberprogramme realisiert.

Eine wirkungsvolle Unterstützung stellen die zahlreichen Utilities dar. Die Bedienung ist leicht aus den Menüs zu erkennen. Zur Entwicklung von WINDOWS-Applikationen wird das **Microsoft WINDOWS TOOLKIT** angeboten, das folgende Möglichkeiten enthält:

- Werzeuge für die Erstellung von Applikationen mit graphischer Benutzeroberfläche
- Unterstützung des Multitasking von WINDOWS
- Datenaustausch für Text- und Graphikinformationen
- gemeinsame Nutzung von Systemressourcen (Speicher, Bildschirm, Tastatur und Maus)
- Zugriff auf das GDI für Graphik und Text
- Nutzung virtueller Geräteschnittstellen (Treiberprogramme)
- viele Beispielprogramme in MS-C und MS-PASCAL.

Für die Erstellung von WINDOWS-Applikationen sind eine Reihe von Arbeitsschritten erforderlich. Die Generierung der gewünschten Piktogramme, Positionsmarken der Maus, und anderer Felder erfolgt mit einem speziellen Editor **IconEdit**. Menüs und Dialogmasken werden mit einem weiteren Editor **DlgEdit** entworfen. Die Programmentwicklung erfolgt mit den bereitgestellten Prozeduren

in einer Programmiersprache (Pascal, C, FORTRAN, MASM). Für das Linken existiert ein spezieller Linker **LINK4**. Das Übersetzen und Linken der Ressourcedateien erfolgt mit dem Ressourcencompiler **RC**.

## 9.2. GEM

GEM (graphics environment manager) ist ein Programmpaket von Digital Research, das bereits eine weite Verbreitung gefunden hat. Es existiert nicht nur als PC-Version, sondern ist auch als Benutzerschnittstelle auf Computern mit dem Prozessor M68000 (z.B. Atari ST u.ä.) anzutreffen. GEM besteht aus mehreren Teilkomponenten, die alle aufeinander abgestimmt sind. Die grundlegenden Bestandteile sind

- **GEM VDI** **GEM-Graphikschnittstelle (virtual device interface)**
- **GEM AES** **GEM-Routinen (application environment service)**
- **GEM Desktop** **GEM-Benutzerschnittstelle.**

Das **GEM VDI** ist eine spezielle Realisierung des als Graphikinterface standardisierten VDI (bzw. CGI) und aus dem Vorgängerprodukt GSX (graphic system extension für CP/M und MS-DOS) der gleichen Firma hervorgegangen. Die Graphikschnittstelle wird im Abschn. 10.5.1. etwas näher erläutert.

**GEM AES** (application environment service) ist eine Sammlung von Routinen, um graphische Eingaben und Systemdienste in einheitlicher GEM-Philosophie unter Nutzung der Graphikschnittstelle GEM VDI zu unterstützen. Das AES besteht aus 5 Komponenten, die speziellen Aufgabenkomplexen zugeordnet sind und beim Laden von GEM im Speicher oberhalb von GEMVDI abgelegt werden:

- Unterprogrammbibliothek
- Kern und Dispatcher für begrenztes Multitasking
- Shell
- Puffer für ACC-Dateien (desk accessory buffer)
- Puffer für Menüdaten und Warntafeln (menu/alert buffer).

Anwendungsprogramme, die das GEM verwenden, heißen Applikationen und tragen die Dateikennzeichnung APP. Eine typische GEM-Applikation ist das Programm GEM-Desktop, das eine alternative Benutzerschnittstelle zur Systembedienung darstellt.

Die **Unterprogrammbibliothek** liefert Unterstützung zur Windowtechnik, Überwachung der Mausbedienung, Ausgabe von System- und Fehlermeldungen und Darstellung von Objekten auf dem Bildschirm. Die Bibliotheksroutinen sind speicherresident und werden erst bei Beendigung von GEM entfernt.

Der **Kern für begrenztes Multitasking** steuert die quasiparal-

lele Bearbeitung einer primären Applikation (z.B. GEM-Desktop oder WordStar), mehrerer als Overlay geladener Hintergrundprozesse (z.B. desk accessories) und des Bildschirmverwalters (screen manager). Die Prozesse werden durch den **Dispatcher** über eine Ready-Liste (lauffähige Prozesse) und Nonready-Liste (Prozesse, die auf ein Ereignis warten, wie z.B. Tastatureingabe, Zeitintervall, Mausbewegung oder eine Botschaft) verwaltet und die CPU-Zeit auf die Prozesse verteilt. Das Umschalten erfolgt bei jedem Aufruf von GEM-Routinen.

Die **Shell** startet ein Anwendungsprogramm und stellt die notwendige Gerätetreiberumgebung her (zeichenorientierte Arbeit mit DOS-Aufrufen bzw. graphikorientierte Arbeit mit GEM VDI).

Im **Puffer für ACC-Dateien** werden die Hintergrundprogramme (desk accessories) abgelegt. Die Anzahl der ACC-Dateien ist vom verwendeten Betriebssystem abhängig (für DOS sind 3 ACC-Programme zugelassen).

Im **Puffer für Menüdaten und Warntafeln** werden die beim Ausgeben von Menüs und anderen zeitweise benötigten Zusatzwindows überlagerten Teile des Bildschirms aufbewahrt und beim Schließen der Zusatzwindows zurückgespeichert.

Eine typische GEM-Anwendung realisiert folgende Schritte:

- Initialisieren der Applikation
- Ermittlung der Bildschirmdaten
- Laden der Resourcendatei (RSC), wie
  Texte, Icons, Menüs, Dialoge, Formulare
- Öffnen eines Fensters (window)
- Ausgabe des Menübalkens oder Start des Anwenderdialogs
- Abfrage der Ereigniswarteschlange
- Auswertung des Ereignisses
- Ausführung notwendiger Programmschritte
- Wiederholung Ereignisbearbeitung bis Endebedingung
- Programmbeendigung.

Zur Entwicklung von GEM-Applikationen wurde das **GEM-Programmer Toolkit** entwickelt. Kern des Toolkit (Werkzeugkasten) ist das GEM-Programm RCS.APP, das die interaktive Erstellung der benötigten Graphikelemente einer Applikation unterstützt. Die erzeugte Datei enthält alle Dialogfenster, Warntafeln, Hilfefenster, Fehlermeldungen und Menüs.

Die in einem Baukasten enthaltenen Graphikelemente (Icons) können mit dem ICON-Editor an die Wünsche des Nutzers angepaßt werden (pixelweise Editierung).

Mit dem RCS werden die einzelnen Bausteine so dargestellt, wie sie später bei der Programmabarbeitung aussehen. Mit diesem Verfahren kann man erheblich Programmieraufwand einsparen. Außerdem wird die Bedienung von GEM-Programmen einheitlich gestaltet.

Die Systemdienste von GEM werden nach Funktionen in Biblio-

theken zusammengefaßt und mit einem Standardinterface angesprochen.

| GEM AES Funktionsgruppen | |
|---|---|
| APPL | Applikationsbibliothek (applikation library) |
| EVNT | Ereignisbibliothek (event library) |
| MENU | Menübibliothek (menu library) |
| OBJC | Objektbibliothek (object library) |
| FORM | Formularbibliothek (form library) |
| GRAF | Graphikbibliothek (graphics library) |
| SCRP | Datenaustauschbibliothek (scrap library) |
| FSEL | Dateiauswahlbibliothek (file select library) |
| WIND | Windowbibliothek (window library) |
| RSRC | Resourcenbibliothek (resource library) |
| SHEL | Shellbibliothek (shell library) |

| GEM AES Interface | |
|---|---|
| Global Array | GEM-Systemparameter |
| Parameter Block | Adressen der Parameterfelder |
| Control Array | Längen der Parameterfelder |
| int_in | Eingabeparameterfeld (integer) |
| int_out | Ausgabeparameterfeld (integer) |
| addr_in | Eingabeparameterfeld (Adressen) |
| addr_out | Ausgabeparameterfeld (Adressen) |

Die Parameter in den einzelnen Feldern sind beim Aufruf einzutragen bzw. werden von GEM gesetzt. Die Verbindung zu GEM erfolgt wie beim VDI durch ein Softwareinterrupt. In den Feldern **parameter block** und **control array** stehen folgende Angaben:

```
params[0] = Adresse control array
params[1] = Adresse global array
params[2] = Adresse int_in
params[3] = Adresse int_out
params[4] = Adresse addr_in
params[5] = Adresse addr_out

control[0] = Funktionsnummer
control[1] = Anzahl Integer (2 Byte) in int_in
control[2] = Anzahl Integer (2 Byte) in int_out
control[3] = Anzahl Adressen (4 Byte) in addr_in
control[4] = Anzahl Adressen (4 Byte) in addr_out.
```

Zur Einbindung in Applikationen, die in einer höheren Programmiersprache geschrieben werden, existieren entsprechende

Spracheinbindungen. Im GEM-Programmer Toolkit ist das MASM-Interface und eine C-Einbindung mit Beispielprogrammen enthalten.

Wenn man mit der GEM-Philosophie vertraut ist, lassen sich mit dem Toolkit sehr schnell leistungsfähige interaktive Programme erstellen.

## 9.3. Sonstige Schnittstellen

Neben den Systemen GEM und WINDOWS existieren wesentlich kleinere Benutzerschnittstellen, die ebenfalls wesentliche Erleichterung bei der Bedienung bringen. Die Bedienung ist meistens relativ einfach zu erlernen und wird durch Hilfsfunktionen unterstützt. Als Beispiele ohne weitere Funktionserläuterungen seien hier die folgenden genannt:

**Norton Commander** (Peter Norton)
Ein residentes Programm, das mausgesteuert arbeitet und sehr effektiv die Auswahl von Dateien für Verwaltungsaufgaben auf DOS-Ebene (Kopieren, Umbenennen, Löschen u.a.) gestattet. Die Kommandoeingabeform von DOS bleibt voll erhalten. Ein kleiner Editor (Tastenbelegung wie bei WordStar) gestattet z.B. schnelle Korrekturen an Systemdateien (CONFIG.SYS, AUTOEXEC.BAT u.a.). Die Arbeit mit zwei Fenstern erleichtert die Kopiervorgänge. Die Möglichkeit einer eigenen Menüdatei unterstützt auch die PC-Anwendung für ungeübte Bediener.

**QDos** (Gazelle Systems)
Ein leistungsfähiges Programm, das sich zwischen COMMAND und Nutzer stellt. Mit den Cursortasten ist die Menüauswahl aller wichtigen Aufgaben zur Dateiverwaltung möglich.

**Xtree** (Executive Systems)
Ähnlich wie bei den anderen Benutzerschnittstellen (DOS-Shells) sind sehr schnell über Menüs alle Dateiverwaltungsaufgaben durchführbar. Wie der Name andeutet, wird durch graphische Darstellung der Verzeichnisstruktur eine gute Übersicht erreicht.

**DOS-Manager** (Microsoft)
Ein kleines, menügesteuertes Bedienungsprogramm, das ähnlich wie bei den anderen menügesteuerten Programmen eine schnelle Funktionsauswahl gestattet.

# 10. Anwendungsprogramme

In diesem Abschnitt werden einige sehr häufig eingesetzte Anwendungsprogramme übersichtsartig dargestellt. Es kann natürlich nur eine kleine Auswahl sein, da für jedes der Programme eine eigene, meistens relativ umfangreiche Dokumentation existiert und der Platz hier beschränkt ist. Die meisten Programme enthalten Funktionen zur Anwendungsunterstützung (Hilfefunktion), die bei der Arbeit am Bildschirm aufgerufen werden können. Bei Problemen vermitteln auch die Bücher zu diesem Themenkreis (vgl. Literaturverzeichnis und weitere Titel in der Reihe Technische Informatik) viele Informationen. Für alle Programmpakete gelten die folgenden Hinweise:

Die Originaldisketten sollten gegen versehentliches Überschreiben durch Überkleben der Schreibschutzkerbe gesichert und vor der Benutzung dupliziert werden (z.B. mit DISKCOPY, COPYIIPC). Läßt sich eine Diskette nicht kopieren, so ist evtl. ein Kopierschutz installiert. In solchen Fällen existiert manchmal ein spezielles Programm zum Erstellen einer Sicherheitskopie auf den Programmdisketten. Ist ein Duplizieren der Originaldisketten nicht möglich, so ist bei fast allen Paketen ein Installationsprogramm vorhanden, um sich eine Arbeitsversion auf Diskette oder Festplatte zu erzeugen. Die installierten Programme können in der Regel gesichert werden (z.B. mit BACKUP). Die Installation setzt eine Kenntnis der speziellen Rechnerbedingungen voraus. In den Programmdokumentationen sind dazu umfangreiche Hinweise enthalten.

Durch zunehmende Komplexität und Leistungsfähigkeit der Programme nimmt der Umfang auf externen Speichermedien ständig zu, so daß oftmals sehr viele Disketten benötigt werden. Wenn man mit den einzelnen Programmen vertraut ist, so läßt sich der Umfang der Programme auf Sicherungsdatenträgern durch Einsatz von Archivierungsprogrammen (z.B. ARC, PKARC) drastisch reduzieren. Da das Komprimieren und Entkomprimieren relativ zeitaufwendig ist, muß man genau überlegen, ob sich der Aufwand lohnt. Das ist z.B. davon abhängig, wie oft man die Daten benötigt.

## 10.1. Textverarbeitung

Eine klassische Anwendung der PC-Technik ist die Textverarbeitung. Aus einfachen Editoren für Programme und Texte haben sich sehr leistungsfähige Textverarbeitungssysteme herausgebildet, die heute bereits die Leistung professioneller Satzsysteme erreichen (desktop publishing). Neben der Erfasssung und Verwal-

tung von Texteinheiten wird zunehmend auch das Mischen von Text und Graphik möglich.

Einige wichtige Begriffe, die im Zusammenhang mit Textsystemen auftreten, sollen hier kurz erläutert werden. Man unterscheidet **zeichenorientierte Textverarbeitung** (Bildschirm im Textmodus) und **graphikorientierte Textverarbeitung** (Bildschirm im Graphikmodus). Einfache Textsysteme arbeiten meist im Textmodus. Die Darstellung der Texte erfolgt auch auf XT-Systemen mit ausreichender Geschwindigkeit. Ein klassischer Vertreter dieser Gruppe ist das Programm WordStar. Spezielle Effekte der Textdarstellung werden durch Befehle im Text gesteuert und nur teilweise auf dem Bildschirm sichtbar. Bei der Gestaltung komplizierter Texte ist es manchmal sehr wichtig, die Druckform direkt auf dem Bildschirm anzuzeigen. Dieses Prinzip heißt **WYSIWYG** (what you see is what you get). Obwohl auch WordStar diese Eigenschaft für sich beansprucht, ist eine echte Anzeige nur mit einem Bildschirm im Graphikmodus möglich. Die Bildschirmausgabe ist dabei jedoch erheblich langsamer und setzt für eine zufriedenstellende Arbeit eigentlich einen PC/AT bzw. einen PC/XT mit erhöhter Taktfrequenz voraus. **Textsysteme**, bei denen im Graphikmodus gearbeitet wird, sind z.B. **MS-WORD**, CHIWRITER und die Textkomponenten von Frame-Work II, Open Access II. Wichtig für die Druckausgabe ist die Steuerung des **Seitenformats**. Die Ausgabe einfacher Texte ist meistens problemlos und unterscheidet sich nicht sonderlich von einem normalen Dateiausdruck. In den Textsystemen sind dazu Standardwerte eingestellt, die evtl. an einen speziellen Drucker anzupassen sind. Vielfach wird die Druckeranpassung durch spezielle Druckertreiber realisiert, die die Besonderheiten des Druckers voll ausnutzen. Die Auswahl des Treibers erfolgt vielfach über den Druckernamen. Wenn ein Druckertyp nicht unterstützt wird, so ist ein kompatibler Typ auszuwählen bzw. ein eigener Druckertreiber zu konfigurieren oder zu schreiben. Sehr oft sind Drucker kompatibel zu EPSON-Druckern oder realisieren eine Ausgabe analog zum Graphics Printer von IBM.

Als Beispiele wurden in diesem Abschnitt die Programme WordStar (Micropro) und MS-Word (Microsoft) ausgewählt. Es existieren sehr viele Textverarbeitungssysteme, die im Funktionumfang ähnlich leistungsfähig sind, auf Spezialgebieten sogar manchmal wesentlich besser. Einige Beispiele sind im Abschn. 3.4. aufgeführt.

### 10.1.1. WordStar

WordStar (WS) ist ein relativ einfaches Textsystem, das zur Erfassung und Verwaltung von Programmtexten und Dokumententexten ausgelegt ist. Die Steuerung erfolgt über Kommandos, die in einer Menüstruktur zur Verfügung stehen bzw. als Punktbefehle in den

Text eingefügt werden. Die Punktbefehle beginnen mit einem Punkt in Spalte 1 der Zeile. Den hierarchischen Menüaufbau sollte man sich beim Lernen der Steuerkommandos klarmachen, weil sonst die Tastenbelegung für die Befehlskodes etwas umständlich erscheint. Im Anhang ist eine Tabelle mit allen Steuerbefehlen von WordStar enthalten. Hier sollen nur die einzelnen Funktionen kurz erläutert werden. Die Darstellung orientiert sich an dem Leistungsumfang der Version 4. Die wesentlichen Funktionen von WordStar sind

- Installation aller Programmfunktionen
- Druckeransteuerung über Druckertreiber
- Bildschirmdarstellung mit eingeschränktem WYSIWYG
- Menüunterstützung und Hilfsfunktionen
- Nutzung des Dateisystems von MS-DOS 2.0 (Gerät, Pfad, Datei)
- Zugriff zum Kommandointerpreter
- komfortable Textbehandlung (Zeichen, Wörter, Absätze, Blöcke)
- umfangreiche Möglichkeiten zur Seitengestaltung
- Unterstützung der Silbentrennung und Wortkorrektur
- Formatierung
- Definition von Makros
- Rechnerfunktion
- Verwendung von Sonderzeichen (z.B. für Liniengraphik)
- Serienbrieffunktion (MailMerge)
- Zusatzpakete (StarAddress, WordFinder, StarPublisher).

Die **Installation** erfolgt mit Hilfe der speziellen Programme WINSTALL bzw. WSCHANGE, die durch Menüs leicht zu bedienen sind. Es ist zu beachten, daß manche Änderungen nicht zurückgestellt werden können. Man sollte zum Experimentieren eine WS-Version mit neuem Namen erstellen. Wichtig für die Einarbeitung ist die Installation des richtigen Bildschirmmodus, da sonst keine Anzeigen von WS erfolgen können. Die Druckerinstallation kann zu einem späteren Zeitpunkt erfolgen, da alle Druckertreiber in einer großen Datei zusammengefaßt sind und beim Ausdrucken eine Auswahl erfolgen kann.

Der Start von WordStar erfolgt durch die Eingabe von WS, worauf sich WS mit dem **Startmenü** meldet.

```
┌----------------------S T A R T M E N Ü----------------------┐
| D Textdatei bearbeiten          N Programmdatei bearbeiten |
| P Datei drucken                 I Index anlegen            |
| O Datei kopieren                T Inhaltsverzeichn. anlegen|
| Y Datei löschen               Esc Makros                   |
| E Datei umbenennen              F Verzeichnis ausblenden   |
| C Dateischutz                   L Laufwerk wechseln        |
| M Mailmerge                     R Programm aufrufen        |
| J Hilfe                         X WordStar verlassen       |
└------------------------------------------------------------┘
```

Hier sind grundlegende Funktionen zur Dateiverwaltung zusammengefaßt. Die Funktionen Kopieren, Löschen, Umbenennen, Laufwerk wechseln und Programm aufrufen sind ohne weitere Erläuterungen verständlich. Von WS werden die erforderlichen Informationen angefordert. Die Unterbrechung einer Funktion kann immer mit ^U (Ctrl-U) erfolgen. Die weiteren Menüpunkte bringen ein weiteres Menü zur Anzeige. Die Anzeige der Menüs läßt sich durch Auswahl einer Hilfsstufe im **Hilfsmenü (^J)** an die Wünsche des Nutzers anpassen. Wichtig ist die Unterscheidung zur Bearbeitung einer Textdatei oder einer Programmdatei. Während Programmdateien durch zeilenorientierte Bearbeitung behandelt werden, wird bei Textdateien der volle Funktionsumfang von WS genutzt. Die Speicherung von Textdateien erfolgt mit einer eigenen internen Zeichendarstellung. Die Verwaltung von Texten und Programmen erfolgt unter Steuerung des **Hauptmenüs.**

```
┌------------------------------ H A U P T M E N Ü ------------------------------┐
| CURSOR              BILDSCHIRM            LÖSCHEN            SONSTIGES        |
| ^E Nach oben        ^W Zeile nach oben    <- Zeichen links   ^J Hilfe          |
| ^X Nach unten       ^Z Zeile nach unten   ^G Zeichen         ^V Einfügen aus   |
| ^S Zeichen links    ^R Seite nach oben    ^T Wort            ^B Formatieren    |
| ^D Zeichen rechts   ^C Seite nach unten   ^Y Zeile           ^N Zeile teilen   |
| ^A Wort links       UNTERMENÜS            ^U Undo            ^L Suchen/Ersetzen|
| ^F Wort rechts      ^O Bildschirm  ^P Drucksteuerzeichen       wiederholen     |
| ^I Tab              ^K Block/Datei ^Q Schnellmenü    ESC Makros                |
└-------------------------------------------------------------------------------┘
```

Die Steuerung der aktuellen Position im Text erfolgt durch einen Cursor, der mit den Cursorkommandos (Pfeiltasten oder spezielle Ctrl-Kombinationen) bewegt wird. Die Befehle im Hauptmenü sind zu Gruppen zusammengefaßt. Neben Cursorkommandos existieren Kommandos zum Bewegen des Textfensters auf dem Bildschirm, zum Löschen von Texteinheiten, zum Aufruf der Hilfsfunktion und zum Aufruf weiterer Menüs (Untermenü). Alle Befehle werden unter Verwendung der Ctrl-Taste (in den Beschreibungen durch ^ gekennzeichnet) eingegeben. Einige Funktionen lassen sich den Funktionstasten zuordnen, deren Bedeutung in den beiden untersten Zeilen auf dem Bildschirm angezeigt wird. Das Untermenü **Bildschirmdarstellung (^O)** stellt Befehle zur Textdarstellung bereit.

```
┌---------------- B I L D S C H I R M D A R S T E L L U N G ----------------┐
| ZEILENLINEAL              TEXTEINGABE  UND TEXTDARSTELLUNG                  |
| L Linken Rand setzen      I Tabulator setzen  S Zeilenabstand ändern        |
| R Rechten Rand setzen     N Tabulator löschen C Zeile zentrieren            |
| O Zeilenlineal kopieren   W Wortumbruch aus   D Trenn-  Steuerzeichen aus    |
| F Hauptlineal angleichen  J Blocksatz aus     H Trennhilfe ein              |
| X Rand lösen              E Trennvorschlag    P Datei nur lesen ein         |
| T Hauptlineal ausblenden  G linker Rand       B Markierung weicher          |
|                             vorübergehend       Leerschritte ein            |
└-------------------------------------------------------------------------------┘
```

Die Kommandos sind selbsterklärend. Das Kommando ^OD auf Hauptmenüebene wird benutzt, um die für den Ausdruck bestimmten Kommandos anzuzeigen oder auszublenden. Das ist manchmal wichtig, um das aktuelle Ausdruckformat festzustellen. Für die Erfassung von Tabellen sollte der Blocksatz und der Wortumbruch ausgeschaltet werden. Bei allen Bildschirmkommandos ist zu beachten, daß sie nur für die aktuelle Anzeige dienen. Feste Einstellungen sind mit den Punktkommandos im Text zu erreichen.

Wichtig für die Dateiverwaltung im Editiermodus ist das Menü der **Block- und Dateibefehle (^K)**.

```
┌---------------- B L O C K-  UND D A T E I B E F E H L E ----------------┐
|  SPEICHERN                      BLOCKBEFEHLE          DATEIBEFEHLE      |
|  S Speichern, weiterarbeiten    B Blockanfang         P Datei drucken   |
|  D Speichern, Datei verlassen   K Blockende           R Datei einlesen  |
|  X Speichern, WS verlassen      H Markierung aus      O Datei kopieren  |
|  Q Bearbeitung abbrechen        C Block kopieren      E Datei umbennen  |
|  BLOCKBEFEHLE                   V Block verschieben   J Datei löschen   |
|0-9 Lesezeichen setzen           W Block speichern     L Verzeichnis wechseln |
|  ' Groß- zu Kleinbuchstaben     Y Block löschen                         |
|  " Klein- zu Großbuchstaben     N Kolumne ein         F Programm aufrufen |
|  M Rechnen                      I Kolumne versetzen ein                 |
└-------------------------------------------------------------------------┘
```

Durch spezielle Kommandos lassen sich Textbereiche als Block kennzeichnen, der dann als Ganzes einer Operation unterworfen wird (z.B. Verschieben, Kopieren). Die Verbindung zum DOS wird in diesem Menü hergestellt (Dateiverwaltung, Aufruf Kommandointerpreter, Drucken). Die **Druckersteuerzeichen (^P)** werden durch ein eigenes Menü unterstützt. Der Druckvorgang wird gestartet entweder aus dem Startmenü, aus der Datei (Drucken erfolgt parallel zum Editieren!) oder vom Bildschirm. Als Ausgabemedium kann entweder der Drucker oder eine Datei gewählt werden. Eine spezielle Druckfunktion ist das verschachtelte Drucken (Mischdruck, Serienbrieffunktion, MailMerge). Dabei werden Informationen aus mehreren Dateien miteinander gemischt (vgl.Dokumentation).

```
┌---------------- D R U C K E R S T E U E R Z E I C H E N ----------------┐
|  TOGGLES                  EINZELBEFEHLE                                 |
|  B Fettdruck              H Zeichen überdrucken   L Seitenvorschub      |
|  D Doppeldruck            <┘ Zeile überdrucken    F Sonderzeichen       |
|  S Unterstreichen         O fester Leerschritt    G Sonderzeichen       |
|  T Hochstellen            C Druckpause einfügen   Q frei                |
|  V Tiefstellen            I Spaltentabulator      W frei                |
|  Y Kursiv/Farbe           @ absoluter Tabulator   E frei                |
|  X Durchstreichen         N Standardschreibdichte R frei                |
|  K Markierung für Index   A zweite Schreibdichte                        |
└-------------------------------------------------------------------------┘
```

Die unter TOGGLES zusammengefaßten Kommandos schalten einen Status um. Der auszuzeichnende Text wird durch entsprechende Kennungen eingeklammert.

Einige Befehle zur Beschleunigung der Bearbeitung sind im **Schnellmenü (^Q)** zusammengefaßt.

```
┌------------------------- S C H N E L L M E N Ü -------------------------┐
| CURSORBEWEGUNGEN                                    SUCHEN UND ERSETZEN |
| E Bildschirm oben    P letzte Cursor-Position       F Suchen            |
| X Bildschirm unten   V letztes Suchen/Ersetzen      A Suchen und Ersetzen |
| S Zeilenanfang       B Blockanfang                  G Zeichen vorwärts  |
| D Zeilenende         K Blockende                    H Zeichen rückwärts |
| R Dateianfang      0-9 Lesezeichen                  I Seite/Zeile suchen |
| C Dateiende          ? Zeichen zählen                                   |
| LÖSCHEN          BILDSCHIRMINHALT    KORREKTUREN        SONSTIGE        |
| T bis Zeichen    W nach unten rollen L Rest prüfen      U Datei formatieren|
| Y Zeile rechts   Z nach oben rollen  N Wort prüfen      Q Wiederholfunktion|
| - Zeile links                        O Wort eingeben    M Rechnen       |
└-------------------------------------------------------------------------┘
```

Die ESC-Taste führt zum Menü der **Makros**, mit dem sich Folgen von Kommandos zu einem Makro zusammenfassen lassen. In diesem Menü wird auch eine einfache Rechnerfunktion bereitgestellt.

Für die Ausgabe auf den Drucker existieren eine Reihe von Kommandos, die den **Satzspiegel** bestimmen. Mit den Standardeinstellungen kann man einfache Texte sinnvoll gestalten. Für spezielle Anforderungen sind folgende Festlegungen erforderlich:

- Seitenlänge (Anzahl Zeilen je Seite bei gegebener Zeilenhöhe)
- oberer Rand (Anzahl Zeilen bis Textanfang)
- unterer Rand (Anzahl Zeilen nach Textende)
- linker und rechter Rand (Anzahl Zeichen links bzw. rechts)
- Seitennumerierung (Standard unterhalb vom Text)
- Seitenumbruch
- Kopf- und Fußzeilen (Abstand vom Text)
- Zeilenhöhe
- Zeilenabstand
- Schriftdichte
- Einzüge (Absatzanfang).

Die für diese Festlegungen vorhandenen Punktbefehle sind im Anhang in einer Tabelle zu WordStar zu finden. Mit den Punktbefehlen wird eine Funktion bis zur Neufestlegung eingestellt. Da diese Befehle im Text stehen, sind sie auch beim erneuten Bearbeiten einer Datei wieder gültig.

Die Arbeit mit WordStar ist insgesamt relativ problemlos und hat dazu geführt, daß viele Nutzer bei diesem Programm geblieben sind, obwohl wesenlich leistungsfähigere Textsysteme existieren.

Auch das Programm WordStar 2000 der Firma MicroPro konnte sich dagegen nicht richtig durchsetzen. Das Textformat von WS wird von vielen Programmen als Eingabeformat akzeptiert. Es existieren auch Konvertierungsprogramme (z.B. WSCONVRT), um WS-Dateien in andere Formate umzusetzen. Für den Nutzer, der mit Compilern der Firma Borland (USA) arbeitet (z.B. Turbo-Pascal), ergibt sich außerdem der Vorteil gleicher Editorkommandos wie bei WordStar.

### 10.1.2. MS-Word

Das Programm WORD von Microsoft ist ein ausgereiftes Textsystem, das hohen Anforderungen gerecht wird. Es kann im Zeichenmodus (ablauffähig auf jedem Bildschirmtyp) und im Graphikmodus betrieben werden. Im Graphikmodus werden alle Textbesonderheiten (Schriftart, Größe, Abstand u.a.) direkt am Bildschirm dargestellt. Im Textmodus werden Auszeichnungen, z.B. durch Intensivdarstellung für Fettdruck, kenntlich gemacht. Für den Entwurf größerer Texte besteht die Möglichkeit, sich eine Gliederung anzufertigen und zu verwalten. Sehr viele Kommandos sind für Zeichen, Absatz, Bereich und Druckgestaltung vorhanden. Durch Makros lassen sich häufig wiederkehrende Eingabefolgen katalogisieren. Eine Einführung in die Arbeit mit WORD würde den Rahmen dieses Kapitels sprengen. Zur Einarbeitung ist entweder das Handbuch zu studieren, das Lernprogramm zu aktivieren oder schrittweise die HILFE-Funktion aus dem Befehlsmenü als Unterstützung beim Schreiben anzuwenden. Um den bereits erarbeiteten Text zu sichern, sollte man von Zeit zu Zeit den Befehl <Übertragen-Speichern> aufrufen.

Durch WORD wird die Arbeit mit einer Maus sehr gut unterstützt. Besonders zur Menüauswahl und Ausführung von Befehlen (z.B. Markierung von Textbereichen) ist die Mausanwendung zu empfehlen.

Für umfangreiche Texte ist die Möglichkeit zur gleichzeitigen Bearbeitung von bis zu 7 Fenstern sehr hilfreich. Mit der Funktionstaste <F1> kann zwischen den Fenstern gewählt werden. Mehrere Spalten sind problemlos möglich. Die Druckausgabe ist sowohl im Direktmodus als auch im Hintergrund möglich. Durch eine umfangreiche Treiberbibliothek sind sehr viele Druckertypen mit WORD nutzbar.

Einige Funktionen, die in der neuen Version 4.0 enthalten sind, werden im folgenden aufgezählt:

- Dateimanager
- Kennzeichnung von Korrekturen
- Gliederung und Inhaltsfunktion
- Druckformatsteuerung
- Einfügen von Kalkulationstabellen (z.B. Multiplan, Excel)

- Wortzählung
- Serienbrieffunktion
- Seitenwechselanzeige
- Darstellung im Text- und Graphikmodus
- Zoomfunktion (Erweiterung eines Fensters auf Bildschirmgröße)
- Darstellung von Linien und Einrahmungen
- Rechnerfunktionen
- Wörterbuch und Trennhilfe.

Die wichtigste Taste, um von der Textbearbeitung zum Menü zu gelangen, ist die Taste <Esc>, die auch zur Rückkehr zum Text genutzt wird. Die Hilfefunktion ruft man mit <Esc-H>.

## 10.2. Tabellenkalkulation

Die Tabellenkalkulation ist eine der ersten Anwendungen von PC gewesen. Einige Pakete sind schon seit der CP/M-Zeit auf 8-Bit-Rechnern bekannt. Das grundlegende Prinzip aller Tabellenkalkulationsprogramme ist gleich. Ein Arbeitsblatt mit einer maximalen Dimension (Zeilen/Spalten) wird zellenweise beschrieben, wobei in jeder Zelle entweder Zahlen, Zeichenketten oder Formeln stehen dürfen. Durch Angabe der Zeile und Spalte wird eine Zelle eindeutig bestimmt und kann als Operand in Formeln verwendet werden. Durch die Verbindung der Tabellenkalkulation mit graphischer Anzeige werden die Berechnungen sehr anschaulich und überzeugend. Bekannte Programme sind MS-Multiplan (Microsoft), Lotus 1-2-3 (Lotus), SuperCalc (Computer Associates), MS-Excel (Microsoft), Quattro (Borland), Javelin (Ashton Tate). Wer das Grundprinzip der Tabellenkalkulation begriffen hat, wird sich in diesen Programmen sehr schnell zurechtfinden, zumal durch HILFE-Funktionen alle Kommandos am Bildschirm erläutert werden.

## 10.3. Datenbanken

Das bekannteste Programm zur Verwaltung von Datenbanken ist mit Sicherheit das Programm dBase II bzw. dBASE III von Ashton-Tate. Es läuft standardmäßig vollständig menügesteuert, hat aber auch eine eingebaute Programmiersprache, mit der sich leicht eigene Programmabläufe programmieren lassen. Durch maskengesteuerte Eingabe von Datensätzen besteht die Möglichkeit der Eingabeüberwachung und Fehlerkorrektur. Durch die Bildung von Indexdateien lassen sich Datenbestände nach verschiedenen Felder sortiert bearbeiten und wird ein schneller Zugriff bei logischen Verküpfungen in einer Anfrage möglich. In der Plus-Version ist auch eine Nutzung im Rechnerverbund möglich.

Weitere Programme zur Verwaltung von Datenbanken sind z.B.

RBase (Microsoft), Reflex (Borland), GBase (SPI), Oracle (Kettler Consulting) und die Datenbankkomponenten in integrierten Paketen.

## 10.4. Rechnerkommunikation

Die zunehmende Verbreitung von PC führt dazu, daß auch Bestrebungen zur Arbeit im Rechnerverbund entstehen. Für die technische Realisierung von Rechnernetzen gibt es eine Reihe von Lösungen, z.B. IBM token ring, token bus (MAP), CSMA/CD (TOP), die im einzelnen hier nicht behandelt werden können. Zur Abwicklung der Übertragungsprotokolle gibt es ebenfalls verschiedene Lösungen, von denen nur einige direkt durch MS-DOS unterstützt werden. Für die meisten Zusatzpakete sind Spezialkarten erforderlich, die die Datenübertragung effektiv mit hoher Übertragungsrate unterstützen. Einige Möglichkeiten bieten jedoch auch die Standardschnittstellen COM1 und COM2 zur seriellen asynchronen Kommunikation. Damit sind

- Terminalemulation
- Dateitransfer

relativ einfach realisierbar. Als Beispiel soll hier das Programm Kermit genannt werden, das auf sehr vielen Rechnern zur Verfügung steht. Es wird ein zeichenweiser Transport von Dateien realisiert, der durch XON/XOFF gesteuert wird. Damit sind normalerweise alle Terminalleitungen eines Rechners zur Datenübertragung geeignet. Die Übertragungsparameter können beliebig eingestellt werden. Eine einfache Hilfefunktion (an allen Stellen erfolgt auf ? eine Information zu möglichen Eingaben) macht eine Dokumentation fast überflüssig. Kermit ist mit Quelltexten verfügbar und gestattet dadurch auch eine einfache Übernahme auf neue Rechner. Das Kermit-Protokoll wird zunehmend auch in anderen Kommunikationsprogrammen unterstützt (z.B. Open Access II). Auf weitere Netzsoftware, wie z.B. MS-Net, NET-BIOS, LAN-Manager, kann hier nicht eingegeangen werden.

## 10.5. Graphik

Die Graphikfähigkeit gehört heute zu den Standardeigenschaften eines PC. Gerade die graphische Präsentation von Rechnungsergebnissen ermöglicht eine effektive Datenauswertung und überzeugende Darstellung eigener Ergebnisse (z.B. für Werbungszwecke). Zwei wichtige Gruppen von Graphikprogrammen sind zu unterscheiden, die Grundgraphik zur Geräteansteuerung und die Anwendung dieser Grundgraphik zur Darstellung von Zeichnungen, Konstruktionen und Graphiken, die zusammen mit Text gedruckt werden. Die

dazu verfügbaren Programmpakete werden in den folgenden Abschnitten kurz erläutert. Weitere Pakete sind im Abschn. 3.4. genannt.

### 10.5.1. Grundgraphik

Zur Realisierung graphischer Probleme wurden große Anstrengungen unternommen, um zu standardisierten Schnittstellen zu gelangen, da die Zahl graphischer Geräte unübersehbar geworden ist. und Programme sonst nicht geräteunabhängig geschrieben werden können. Ausgehend von diesen Standards bzw. Standardentwürfen wurden durch viele bekannte Softwarehäuser Graphikpakete angeboten, die sich mehr oder weniger an diese Standards halten. In diesem Abschnitt soll auf folgende Systeme kurz eingegangen werden, wobei die Funktionen im einzelnen nicht erläutert werden können:

- GDI (Microsoft Windows Development Toolkit)
- VDI (IBM Graphics Development Toolkit)
- GEM VDI (Digital Research VDI für GEM).

Bei Anwendungsprogrammierung in einer höheren Sprache sollte eine Bibliothek verwendet werden, die in den entsprechenden Toolkits enthalten ist. Die fensterorientierte Arbeitsweise von Windows und GEM macht die Nutzung mehrerer virtueller Bildschirme möglich. Zwischen den einzelnen Fenstern kann beliebig umgeschaltet werden. Dadurch entstehen mehrere logische Ausgabekanäle in einem Programm. Die VDI-Schnittstelle ist vielfach als untere Ebene eines Graphiksystems anzusehen, auf der andere Programme aufbauen. Auch die graphisch arbeitenden Benutzerschnittstellen (Desktop, Oberflächen) nutzen diese Möglichkeit. Neben diesen Standardrealisierungen existieren eine Reihe weitere Programmpakete, z.B. zu Turbo-Pascal (Graphix Toolbox, TurboHALO), die eigene Bibliotheken verwenden. Mit den neuen Turbo-Compilern bietet Borland für mehrere Sprachen das BGI (Borland Graphics Interface) an. Auch in den neuen Compilern von Microsoft (C, QuickC) sind bereits Graphikfunktionen realisiert. Es gibt hier für den Anwender vielfältige Möglichkeiten zum Experimentieren; es ist letztlich aber doch sinnvoll, sich an eine Standardschnittstelle zu halten, die auch über mehrere Jahre verfügbar ist.

Das **Windows Development Toolkit** ist die Basis für viele unter Windows laufende Programme, wie z.B. MS-Paint, MS-Write und Aldus Pagemaker.

Im **Graphics Development Toolkit** (Professional Graphics Series) von IBM bzw. GSS wird die Treiberphilosophie von MS-DOS voll ausgenutzt. Alle verwendeten Gerätetreiber werden durch Angaben in der Datei CONFIG.SYS in das System eingebunden. Die

VDI-Schnittstelle ist von BASIC, C, Pascal, Fortran und Assembler aus aufrufbar. Das VDI wird als Basis für weitere Graphikpakete genutzt. Dazu gehören das Graphical Kernel System (GKS), das Plotting System, das Graphical File System (Bearbeitung eines Metafile) und das Programm Graphics Terminal Emulator.

Weite Verbreitung hat das im Programmsystem GEM genutzte **GEM VDI** als Bestandteil des **GEM Development Toolkit** gefunden. Als Graphikerweiterung zum Betriebssystem CP/M und MS-DOS (und teilweise kompatibler Systeme wie concurrent-DOS, DOS-plus) wurde von der Firma Digital Research das Graphikpaket GSX angeboten, für das auch Spracheinbindungen für z.B. Turbo-Pascal und C existieren. Zur Realisierung graphischer Probleme im System GEM (vgl. Abschnitte 9.2. und 10.5.2.) wurde GSX zu GEM VDI weiterentwickelt. GEM VDI stellt eine geräteunabhängige Graphikschnittstelle dar, mit der man GEM-Applikationen realisieren kann. Diese Schnittstelle läßt sich auch unabhängig von GEM nutzen.

### 10.5.2. Graphikanwendungen

Die Graphikanwendungen sind durch die leichte Programmierbarkeit von Graphik auf dem PC sehr vielfältig. Da im Rahmen diese Buches keine umfassende Übersicht zu allen Programmen gegeben werden kann, folgt hier nur eine Aufzählung der Anwendungsklassen mit einigen typischen Vertretern.

**Präsentationsgraphik** zur professionellen Darstellung von Diagrammen:
MS-Chart (Microsoft), GEM-Graph (Digital Research), GEM-WordChart (Digital Research), Freelance (Lotus) als Ergänzung zu Lotus 1-2-3 und Symphony, Plotting System und File System (IBM, GSS)

**Bildgraphik** zur Erstellung von freien Zeichnungen:
GEM-Draw und GEM-Paint (Digital Research), MS-Paint (Microsoft)

**Konstruktion (CAD)** für komplizierte technische Zeichnungen:
AutoCad (Autodesk), VersaCAD (Versacad Corp.)

## 10.6. Multifunktionspakete (integrierte Software)

### 10.6.1. Übersicht

Die in den vorangehenden Abschnitten genannten Komponenten Textverarbeitung, Tabellenkalkulation, Datenbank, Kommunikation und Graphik sind in komplexen Anwendungen oft nebeneinander erforderlich und müssen für die gleichen Daten eingesetzt werden. Mit Multifunktionspaketen (integrierte Software) wird versucht,

die Notwendigkeit zum Umschalten zwischen verschiedenen Paketen und damit unterschiedlichen Bedienungsformen zu umgehen. Zu diesem Zweck bieten große Softwarehäuser alle genannten Funktionen unter einer einheitlichen Benutzerschnittstelle an. Die Leistungsfähigkeit dieser Pakete ist weitgehend ähnlich, die Bedienungsform z.T. sehr unterschiedlich. Alle Pakete bedürfen einer Eingewöhnungszeit, um alle Funktionen voll auszunutzen. In den folgenden Abschnitten wird eine kurze Charakteristik für die interierten Pakete Symphony, FrameWork II und Open Access II gegeben. Weitere Pakete sind im Abschn. 3.4. aufgeführt.

### 10.6.2. Symphony

Das integrierte Paket Symphony (Lotus) setzt die mit Lotus 1-2-3 angefangene Tradition fort und enthält weitere zusätzliche Funktionen (u.a. auch Fensterfunktionen). Es sind die Komponenten Tabellenkalkulation, Graphik, Datenbank, Textverarbeitung und Kommunikation unter einer relativ einfach zu handhabenden Menüsteuerung verfügbar. Die Aufnahme von Daten aus anderen Systemen wird ebenso unterstützt wie die Ansteuerung sehr vieler Gerätetypen durch entsprechende Treiber. Umfangreiche Datensammlungen sind durch die Nutzung der Erweiterungskonzepte für RAM-Speicher (z.B. EMS) möglich. Eine ausgebaute Hilfefunktion (Funktionstaste F1) informiert über mögliche Eingabevarianten in bestimmten Situationen. Durch die Eingabe von SYMPHONY meldet sich das Programm mit dem voreingestellten Funktionsbereich (z.B. BLATT d.h. Tabellenkalkulation). Das Menü zur Auswahl eines anderen Bereiches ist jederzeit durch Funktionstaste F10 (Menü) oder F9 (Service) möglich.

Im Funktionsbereich **Tabellenkalkulation (BLATT)** kann ein Arbeitsblatt 8192 Zeilen mit 256 Spalten enthalten. Integrierte Formeln (mehr als 70 verschiedene Funktionen für Arithmetik, Logik, Zeichenfolgen, Finanzen, Statistik u.a.), Formatierungshilfen und Druckoptionen gestatten eine gute Anpassung an die eigenen Problemstellungen. Die **Datenbank (MASKE)** unter Symphony enthält in jeder Zeile des Arbeitsblatts einen Datensatz mit mehreren Feldern. Die Datensammlung kann leicht nach gewünschten Feldern sortiert und selektiert werden. Die Bedienung erfolgt über Masken, die eine unkomplizierte Arbeitsweise gestatten. Der Funktionsbereich **Textverarbeitung (TEXT)** bietet die erforderlichen Funktionen zur Verwaltung eines Textfensters, mit denen sich einfache und komplexere Texte aufnehmen und mit den anderen Funktionsbereichen koppeln lassen. Damit sind z.B. auch Serienbriefe möglich. Mit der Komponente **Graphik** lassen sich Auswertungen des Arbeitsblatts und auch der Datenbank vornehmen. Die Verwendung der Fenstertechnik zur gleichzeitigen Darstellung macht gerade diese Funktion zur Datenanalyse sehr interessant. Über die **Kommu-**

**nikation** ist die Verbindung zu anderen Rechnern möglich.

Durch spezielle Zusatzprogramme ist z.B. auch die Aufnahme von Graphiken (z.B. Freelance von Lotus) möglich.

### 10.6.3. FrameWork II

Das Programmpaket FrameWork II stammt von der Firma Ashton Tate, dem Entwickler des erfolgreichen dBASE. Unter dem Systemkonzept **Frame** (allg. Bezugsrahmen) sind die Komponenten Textverarbeitung, Graphik, Tabellenkalkulation, Datenbank, Kommunikation und diverse Dienstprogramme zu einem integrierten Paket zusammengefaßt. Die Integration kann als sehr gelungen bezeichnet werden, da eine einheitliche Behandlung und Programmierung aller Funktionsbereiche möglich ist. Zu jedem Frame gehört ein Programmteil (Formel) in der Programmiersprache FRED, der mit dem Texteditor erfaßt und gewartet werden kann. Eine große Anzahl von Funktionen für alle Bereiche von FrameWork gestattet die Programmierung eigener Abläufe in einer PASCAL-ähnlichen Form. Die Interpretation des FRED-Programms kann kontinuierlich oder auf Anforderung erfolgen. Alle interaktiven Funktionen können auch aus dem Programm heraus aufgerufen werden. Die durch Pull-Down-Menüs erreichbaren Menüpunkte können mit den Cursortasten oder mit bestimmten Tastenkombinationen aufgerufen werden. Eine umfangreiche Hilfestellung erhält man über die Funktionstaste <F1>.

Mit der **Textverarbeitung** ist eine effektive Erfassung und Korrektur von Texten möglich. Textauszeichnungen werden direkt auf dem Bildschirm angezeigt. Verschiedene Schriftarten, Justierungen, Rand- und Absatzgestaltung gehören ebenso dazu wie Rechtschreibprüfung, Layoutgestaltung und Inhaltsverzeichnis (Konzept). Die Einbindung von Daten der anderen Funktionsbereiche ist ohne Probleme möglich. Durch Markierungen lassen sich bestimmte Funktionen auf ganze Bereiche ausdehnen. Mit der Konzeptfunktion lassen sich sehr leicht strukturierte Dokumente aufbauen, die durch geschachtelte Frames dargestellt werden.

Im Funktionsbereich **Tabellenkalkulation** sind die von anderen Kalkulationsprogrammen bekannten Funktionen realisiert. Auch hier erweist sich die Programmierung mit FRED als sehr leistungsfähig. Die Größe der Tabelle kann beim Anlegen des Tabellenframes festgelegt werden (maximal sind 32000 Zeilen und 32000 Spalten möglich). FrameWork gestattet im Funktionsbereich **Graphik** die 2-D-Darstellung der in Tabellen enthaltenen Informationen in verschiedenen Varianten.

In Tabellenform werden auch strukturierte Datensammlungen mit dem Programmteil **Datenbank** verwaltet. Je Zeile wird ein Datensatz mit Feldern einstellbarer Länge gespeichert. Interaktiv oder durch ein FRED-Programm können die Datensätze nach bestimmten Kriterien sortiert, durchsucht oder gefiltert werden. Umfang-

reiche Datenbanken werden nur teilweise im Speicher geführt. Beim Nachladen steigt naturgemäß die Bearbeitungszeit erheblich an. Bestehende Datenbanken im Format von dBASE können bearbeitet werden. Durch die Verknüpfung der Bereiche Datenbank und Textverarbeitung lassen sich sehr leicht Serienbriefe ausgeben.

Im Funktionsbereich **Kommunikation** werden Funktionen zur Verbindung mit anderen PC, einem Host oder auch dem Telefonnetz angeboten. Alle Datenübertragungsparameter lassen sich leicht den geforderten Bedingungen anpassen. Die Kommunikation kann parallel zu anderen Arbeiten ablaufen.

Einige nützliche Dienstprogramme können aus einer Bibliothek aufgerufen werden, die immer auf dem Arbeitsfeld verfügbar ist. Die Funktion **Drucken** kann beliebig konfiguriert werden und ebenfalls parallel zu anderen Arbeiten ablaufen. Durch die Bereitstellung einer großen Anzahl Druckertreiber sind alle gängigen Drucker für FrameWork verwendbar. Die Druckfunktion gilt für alle Framearten.

### 10.6.4. Open Access II

Das Programmpaket Open Access II der Firma SPI (Software Products International) ist eine Sammlung von Programmodulen unter einer einheitlichen Benutzerschnittstelle. Die Module Datenbank, Kalkulation mit 3-D-Graphik, Textverarbeitung, Kommunikation und Deskmanager können einzeln oder auch in einer installierten Form gemeinsam genutzt werden. Die Installation erfolgt menügesteuert. Aus der Vielzahl von Druckertreibern, Graphiktreibern für alle Standardkarten, Zeitzoneneinstellungen und anderen Parameter wird eine angepaßte Version OA zusammengestellt, die auf Grund der Größe (mehr als 1 MByte) nur auf einer Festplatte untergebracht werden kann. Das Programm ist nur teilweise speicherresident; beim Wechsel des Funktionsbereichs wird der entsprechende Programmteil nachgeladen. Nach dem Start von OA wird die Bestätigung des aktuellen Datums gefordert und ein Menü zur Auswahl eines Funktionsbereichs angeboten. Zum Programmpaket gehört eine umfangreiche Dokumentation und ein Lehrprogramm mit sehr vielen Beispielen. Für genügend Übungsstoff ist damit gesorgt. Man kann sich auch gleich an die Arbeit wagen und mit der komfortablen Hilfefunktion (Funktionstaste F1) die nötigen Informationen erfragen. Wichtige Funktionen werden durch Funktionstasten aufgerufen.

Der Modul **Textverarbeitung** gestattet die Bearbeitung von formatiertem oder unformatiertem Text (z.B. Programme) bis zu einer Größe von 32 KByte. Der Text kann in der üblichen Weise formatiert werden; eine Trennhilfe existiert allerdings nicht, man muß mögliche Trennstellen beim Schreiben einfügen.

Der **Datenbankmodul** ist relativ leistungsfähig. Er besteht

aus zwei Komponenten, dem Verwaltungsteil und dem Programmierteil. Die mit dem Programmierteil angebotene Abfragesprache ist an SQL angelehnt, wobei die Abfrage durch OA gesteuert wird. Besonders gut macht sich die automatische Verknüpfung mehrerer Dateien über Relationen. Der Datenbankmodul wird auch in modifizierter Form als GBase unter GEM angeboten.

Im Modul **Kommunikation** sind eine Reihe von Prozeduren zur Datenübetragung zusammengefaßt. Modemansteuerung ist ebenso enthalten wie Terminalemulation, Datenübertragung mit verschiedenen Protokollen (XON/XOFF, XMODEM, Kermit), Verschlüsselung und Bulletin-Board-Funktionen.

Der Modul **Kalkulation** gestattet den Aufbau von Modellen bis zu einer (theoretischen) Größe von 3000 Zeilen und 216 Spalten. Neben den normalen Berechnungsfunktionen und der Darstellung in Tabellenform besteht die Möglichkeit zur graphischen Darstellung der Informationen in vielen Varianten in 2-D oder 3-D (z.B. Balken-, Linien- und Tortendiagrammen). Durch die Integration der Module unter einer einheitlichen Menüführung lassen sich die Graphiken z.B. auch in Texte einbinden.

Verschiedene **Dienstprogramme** sind in einem Deskmanager zusammengefaßt, der z.B. Notizblock, Terminkalender, Adressenkartei, Kalender, Stoppuhr, Taschenrechner, die Installation von speziellen Druckern, Konvertierungsroutinen für verschiedene Formate (WordStar, dBase, Lotus u.a.) und einen einfachen Makroeditor enthält.

# 11. Praktische Tips und Programme

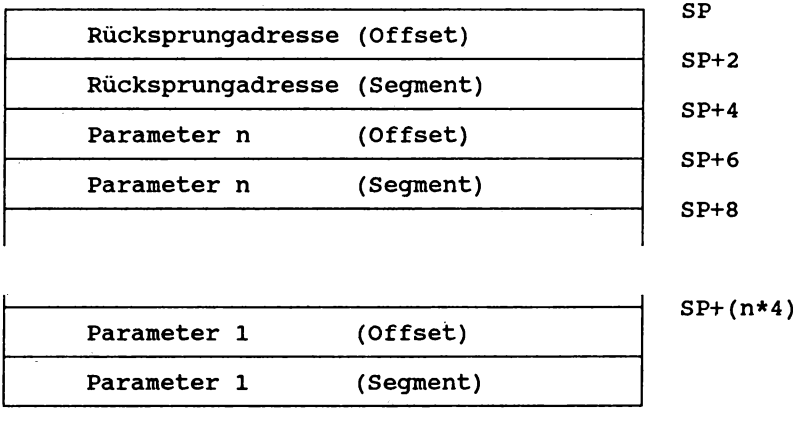
Im Text der vorangegangenen Abschnitte ist mehrfach auf Programmbeispiele in diesem Abschnitt hingewiesen worden. Hier soll versucht werden, die notwendigen Informationen zusammenzufassen, um von einer Programmiersprache aus die Möglichkeiten von MS-DOS voll auszunutzen. Entsprechend der Gliederung im Buch wird die Programmierung auf den einzelnen Niveaus an Beispielen erläutert. Die Beispiele sind vorwiegend in Turbo-Pascal formuliert. Für die Programmiersprachen Fortran, Pascal und C von Microsoft wird nur das Prinzip der Verbindung zu DOS gezeigt. Die Parameter sind weitgehend ähnlich wie bei Turbo-Pascal.

## 11.1. Programmierung auf Hardwareniveau

**Einbindung von Assemblermodulen**

Zur Realisierung von maschinennahen Aufgaben sind manchmal kleine Assemblermodule zu schreiben und mit einem Programm in einer höheren Programmiersprache zu verbinden. Die Compiler von Microsoft verwenden im Prinzip alle den gleichen Standard beim Aufruf von anderen Modulen. Dadurch ist es nicht sehr kompliziert, Module in verschiedenen Sprachen miteinander zu verbinden. Wenn man in einem Assemblermodul die gleiche Aufrufform wählt, lassen sich solche Module problemlos einbinden.

**Modulverbindung bei Compilern von Microsoft (LARGE-Model)**




Es ist zu beachten, daß der C-Compiler eine andere Parameterreihenfolge auf dem Stack verwendet als Pascal und Fortran.

Das hängt damit zusammen, daß bei C-Funktionen nicht alle Parameter angegeben werden müssen. Um im C-Programm die gleiche Parameterfolge wie bei Pascal zu erzeugen, ist die Direktive "pascal" zu verwenden. Bei Fortran werden Parameter auf dem Stack als Adressen übergeben. Bei Pascal gibt es sowohl die Übergabe als Adresse (Referenzparameter mit "var") wie auch als Wert (Wertparameter ohne "var"). Die Adressenangabe mit Segment und Offset erfolgt nur beim LARGE-Speichermodell. Bei den kleineren Modellen wird nur der Offsetteil übergeben. Die Funktionswerte beim Aufruf einer Funktion werden im Register AX zurückgegeben.

**Beispiel 11_1_1:**
**Aufruf einer Assemblerprozedur in MS-Pascal**

**Pascal-Programm CALLASM.PAS**

```
Program Callasm(Input,Output);
var i,summe:integer;
function inc(var i:integer; j:integer):integer;extern;
begin
  i:=20; summe:=inc(i,10); writeln(summe);
end.
```

**MASM-Modul CALLASM.ASM**

```
data      segment public 'DATA'
date      ends
dgroup    group data
          assume cs:incs,ds:dgroup,ss:dgroup
incs      segment 'CODE'
public    inc
inc       proc far
          push bp
          mov  bp,sp
          mov  ax,6[bp]
          mov  bx,8[bp]     ; nur Offset
          add  ax,[bx]
          pop  bp
          ret  4            ; Parameter überspringen
```

**Beispiel 11_1_2:**
**Aufruf einer Assemblerprozedur in MS-Fortran**

**Fortran-Programm**

```
      program Finkey
      integer*2 inkey
      do 101 i=1,250
      write(*,*) 'INKEY Call'
      j=inkey()
      if (j .ne. 0) goto 102
101   continue
102   write (*,*) 'Unterbrechung mit INKEY ',j,' ',char(j)
      end
```

**Assembler-Programm**

```
;       KEY - Aufruf Keyboard-Funktionen von MS-Pascal oder MS-Fortra
;
DATA    segment public 'DATA'
DATA    ends
dgroup  group   DATA
        assume  cs:KEY,ds:dgroup,ss:dgroup
KEY     segment 'CODE'
        public  Inkey
;
;       function Inkey:integer;external;
;       ================================
;
;       Return-Wert = 0   : keine Tastatureingabe
;       Return-Wert <> 0  : 00-xx (hex) --> ASCII-Taste
;                           01-xx (hex) --> Sonderzeichen
Inkey   proc    far
        push    bp
        mov     bp,sp
        push    ds
        mov     ah,0Bh
        int     21h
        mov     ah,0
        or      al,al
        jz      endkey
        mov     ah,08h
        int     21h
        mov     ah,0
        or      al,al
        jnz     endkey
        mov     ah,08h
        int     21h
        mov     ah,1
endkey: pop     ds
        pop     bp
        ret     0
Inkey   endp
KEY     ends
        end
```

**Beispiel 11_1_3:**
**Sichern und Laden von Bildschirmseiten in Datei**

```
program bsp11_1_3(input,output);
uses crt;
var datei:string; ch:char;
procedure pagesave(datei:string);
var   F:file; page:byte absolute $b800:0;
begin
  assign(f,datei); rewrite(f,1);
  blockwrite(f,page,$0fff);
  close(f);
end;

procedure pageload(datei:string);
var F:file; page:byte absolute $b800:0;
begin
  assign(f,datei); reset(f,1);
```

```
  blockread(f,page,$0fff);
  close(f);
end;

begin
  write('Bild-Datei:');readln(datei);
  write('Bild sichern oder laden (S / L):');readln(ch);
  if ch='S' then pagesave(datei);
  if ch='L' then pageload(datei);
end.
```

## 11.2. Programmierung auf BIOS-Niveau

Durch Turbo-Pascal wird eine eingebaute Prozedur zur Nutzung aller Softwareinterrupts der CPU angeboten, die zum Aufruf der BIOS-Funktionen genutzt werden kann. In den anderen Programmiersprachen sind ebenfalls Interruptfunktionen definiert, die entsprechend genutzt werden können. Der folgende Programmbaustein für BIOS-Funktionen wird in Turbo-Pascal als Unit (ab Version 4) genutzt. Für die Version 3 sind Include-Dateien für Programmbausteine üblich; der Text ist dafür entsprechend zu modifizieren.

**Beispiel 11_2_1: Scan-Codes der Tastatur anzeigen**

```
{Scan-Code und erweiterten Scan-Code lesen}
program taste(input, output);
uses crt;
var key:char;
begin
  writeln('Anzeige des Scan-Codes bzw. erweiterten Scan-
Codes');
  writeln('Taste betaetigen, Abbruch mit <Ctrl-Break>');
    repeat
    key:=readkey;
    if key=#0 then writeln('Erweiterter Scan-Code: ',
                           ord(readkey):3)
    else           writeln('            Scan-Code: ',key:3);
    until false;
end.
```

**Beispiel 11_2_2: Unit zur Nutzung der BIOS-Funktionen**

```
{Turbo-Pascal Unit mit BIOS-Funktionen}
{Alle Angaben zu Interrupts in hexadezimaler Schreibweise!}
unit BIOS;

interface
uses dos;

{Hardwarekonfiguration feststellen}

procedure GetROMDate(var Date:string);

procedure Printscreen; {BIOS 05 - Bildschirm ausdrucken}
```

```
{Bildschirmroutinen}

procedure SetVideoMode(Mode:byte);                    {BIOS 10-00}
procedure SetCursorSize(Start,Ende:byte);             {BIOS 10-01}
procedure SetCursorPage(Zeile,Spalte,Seite:byte);     {BIOS 10-02}
procedure GetCursorInfo(Seite:byte; var X,Y:byte;     {BIOS 10-03}
                        Start,Ende:byte);
procedure SetDisplayPage(Seite:byte);                 {BIOS 10-05}
procedure ScrollWindow(ZeileOben,ZeileUnten,     {BIOS 10-06,07}
                       SpalteLinks,SpalteRechts,
                       Anzahl,Attribut,Richtung:byte);
procedure GetCharPage(Seite:byte; var Zeichen:char;
                      var Attribut:byte);             {BIOS 10-08}
procedure WriteCharPage(Zeichen:char; Attribut,Seite:byte;
                        Anzahl:integer);              {BIOS 10-09}
procedure GetVideoMode(var Mode,Spalten,Seite:byte); {BIOS 10-0F
function GetDisplayPage:byte;                         {BIOS 10-0F}

{Tastaturfunktionen}

function ShiftState:byte;                             {BIOS 16-02}
function ShiftR:boolean;
function ShiftL:boolean;
function Ctrl:boolean;
function Alt:boolean;
function ScrollLock:boolean;
function NumLock:boolean;
function CapsLock:boolean;
function Ins:boolean;

{COM-Funktionen}

function InitCOM(Port,Baud,Par,Stop,
                 Data:integer):integer;               {BIOS 14-00}
function SendCOM(Port:byte;Ch:byte):integer;          {BIOS 14-01}
function ReceiveCOM (Port:byte;
                     var Ch:byte):integer;            {BIOS 14-02}
function StatusCOM(Port:byte):integer;                {BIOS 14-03}

implementation

{-------------------------------------------------------------}

var
  Regs:registers;
procedure Printscreen;
begin
  intr(5,regs);
end;

procedure GetROMDate;
var i:integer;
begin
  for I:=0 to 7 do Date[I+1] := chr(mem[$F000:$FFF5 + I]);
  Date[0] := chr(8);
end;

procedure SetVideoMode;
```

```
begin
  Regs.ax:=Mode; {Regs.ah:=$00; Regs.al:=Mode;}
  intr($10,Regs);
end;

procedure SetCursorSize;
begin
  Regs.ah:=$01; Regs.ch:=Start; Regs.cl:=Ende;
  intr($10,Regs);
end;

procedure SetCursorPage;
begin
  Regs.ah:=$02; Regs.dh:=Zeile; Regs.dl:=Spalte;
  intr($10,Regs);
end;

procedure GetCursorInfo;
begin
  Regs.ah:=$03; Regs.bh:=Seite;
  intr($10,Regs);
  Start:=Regs.ch; Ende:=Regs.cl; X:=Regs.dl; Y:=Regs.dh;
end;

procedure SetDisplayPage;
begin
  Regs.ah:=$05; Regs.bh:=Seite;
  intr($10,Regs);
end;

procedure ScrollWindow;
{Richtung=1 : Scroll nach oben}
{Richtung=0 : Scroll nach unten}
begin
  if Richtung=1 then Regs.ah:=$06 else Regs.ah:=$07;
  Regs.al:=Anzahl;     Regs.bh:=Attribut;
  Regs.ch:=ZeileOben;  Regs.cl:=SpalteLinks;
  Regs.dh:=ZeileUnten; Regs.dl:=SpalteRechts;
  intr($10,Regs);
end;

procedure GetCharPage;
begin
  Regs.ah:=$08; Regs.bh:=Seite;
  intr($10,Regs);
  Zeichen:=chr(Regs.al); Attribut:=Regs.ah;
end;

procedure WriteCharPage;
begin
  Regs.ah:=$09;    Regs.al:=ord(Zeichen);
  Regs.bh:=Seite;  Regs.bl:=Attribut;
  Regs.cx:=anzahl;
  intr($10,Regs);
end;

procedure GetVideoMode;
begin
  Regs.ah:=$0F;
```

```
  intr($10,Regs);
  Mode:=Regs.al; Spalten:=Regs.ah; Seite:=Regs.bh;
end;

function GetDisplayPage:byte;
begin
  Regs.ah:=$0F;
  intr($10,Regs);
  GetDisplayPage:=Regs.bh;
end;

function ShiftState:byte;
begin Regs.ah:=$02; intr($16,Regs); end;

function ShiftR:boolean;
begin ShiftR:=ShiftState and 1=1; end;

function ShiftL:boolean;
begin ShiftL:=ShiftState and 2=2; end;

function Ctrl:boolean;
begin Ctrl:=ShiftState and 4=4; end;

function Alt:boolean;
begin Alt:=ShiftState and 8=8; end;

function ScrollLock:boolean;
begin ScrollLock:=ShiftState and 16=16; end;

function NumLock:boolean;
begin NumLock:=ShiftState and 32=32; end;

function CapsLock:boolean;
begin CapsLock:=ShiftState and 64=64; end;

function Ins:boolean;
begin Ins:=ShiftState and 128=128; end;

function InitCOM;
begin
  with regs do begin
  ah:=0;
  case Baud of {nur Baudrate 9600 und 4800, sonst 300}
   4800:al:=192;
   9600:al:=224;
   else al:=64;
  end;
  al:=al+Par shl 3 + pred(stop) shl 2;
  if Data=7 then al:=al+2 else al:=al+3;
  dx:=port;
  Intr($14,regs); InitCOM:=ax;
  end;
end;

function SendCOM;
begin
  regs.ah:=$01; regs.al:=ord(ch); regs.dx:=port;
  Intr($14,regs); SendCOM:=regs.ax;
end;
```

```
function ReceiveCOM;
begin
  regs.ah:=$02; regs.dx:=port;
  Intr($14,regs); ch:=regs.al; ReceiveCOM:=regs.ax;
end;

function StatusCOM;
begin
  regs.ah:=$03; regs.dx:=port;
  Intr($14,regs); StatusCOM:=regs.ax;
end;

begin
end.
```

**Aufruf von Routinen aus der Unit BIOS**

```
{biosbsp.pas}
program biosbsp(input,output);
uses bios;
var
  datum:string;
begin
  writeln('BIOS-Funktionen');
  GetROMDate(datum);
  writeln('ROM-Datum :',datum);
  SetCursorSize(1,7);
  writeln('Cursor auf 1,7 gestellt, Weiter mit <cr>'); Readln;
  SetCursorSize(6,7);
  writeln('SetVideoMode(4);'); SetVideoMode(4);
  writeln('Printscreen');
  printscreen;
end.
```

Die Programmierung auf dem BIOS-Niveau ist sicher nur in Ausnahmefällen erforderlich. Durch Turbo-Pascal wird in der Version 4.0 eine komplete Unit Graph (Borland Graphic Interface BGI) und eine hardwarenahe Unit CRT zur Nutzung der Bildschirm- und Tastaturfunktionen angeboten. Die Unit System enthält weitere Standardfunktionen, um aus einem Pascalprogramm heraus spezielle Hardwareeffekte zu erzielen. Mit der in diesem Abschnitt definierten Unit BIOS werden Beispiele gezeigt, wie man sich seine eigenen Routinen ergänzen kann. Die notwendigen Informationen dazu findet man im Abschn 4.

## 11.3. Programmierung auf DOS-Niveau

Auch für das DOS-Niveau bietet Turbo-Pascal eine spezielle Unit DOS an, die Interruptprozeduren, Prozeduren für Datum und Uhrzeit, Statusverwaltung von Disketten und Festplatten, Verzeichnisverwaltung und Prozeßmanagement enthält. Zum Aufruf von DOS-Funktionen wird eine spezielle Prozedur MsDos bereitgestellt,

die wie im Beispiel 11_3_1 genutzt werden kann. In diese Unit lassen sich alle weiteren DOS-Funktionen aufnehmen, die nicht durch Turbo-Pascal unterstützt werden.

**Beispiel 11_3_1: Unit DOS1 zur Ergänzung der Unit DOS**

```
{ Unit zur Ergänzung der Unit DOS }
Unit DOS1;

interface

uses dos;

var
  Regs:registers;

function GetDate1:string; {DOS Funktion 2A}
function GetTime1:string; {DOS Funktion 2C}

implementation

function GetDate1:string;
const
  Woche : array[0..6] of string[10] =
          ('Sonntag','Montag','Dienstag','Mittwoch',
           'Donnerstag','Freitag','Sonnabend');

var
  Jahr:string[4];
  Monat,Tag:string[2];
  Wochentag:byte;

begin
  Regs.ah:=$2A; msdos(Regs);
  str(Regs.ch*256+Regs.cl,Jahr);
  str(Regs.dh,Monat); str(Regs.dl,Tag);
  Wochentag:=Regs.al;
  GetDate1:=Tag+'.'+Monat+'.'+Jahr+' '+Woche[Wochentag];
end;

function GetTime1:string;
var
  Stunde,Minute,Sekunde,Hundertstel:string[2];

begin
  Regs.ah:=$2c; msdos(Regs);
  str(Regs.ch:2,Stunde); str(Regs.cl:2,Minute);
  str(Regs.dh:2,Sekunde); str(Regs.dl:2,Hundertstel);
  GetTime1:=Stunde+'.'+Minute+'.'+Sekunde+'.'+Hundertstel;
end;

end.


Program bsp11_3_1(input,output);

uses DOS1;
```

```
begin
  writeln('Datum :',GetDate1,' Uhrzeit: ',GetTime1);
end.
```

**Beispiel 11_3_2: Inhaltsverzeichnis erstellen**

Für die Verzeichnisverwaltung können in Turbo-Pascal Standardfunktionen verwendet werden. Folgende Funktionen/Prozeduren sind definiert (genaue Informationen sind in der Dokumentation zu Turbo-Pascal enthalten):

- **FindFirst(pfad:string; attr:byte; var s:searchrec);**
- **FindNext(var s:searchrec);**
- **GetFAttr(var f; var attr:word);**
- **SetFAttr(var f; attr:byte);**

```
{ Inhaltsverzeichnis erstellen }
program bsp_11_3_2(input,output);

uses DOS;

var
  Eintrag:SearchRec;

begin
  FindFirst('*.*', AnyFile, Eintrag);
  while DosError = 0 do begin
    Writeln(Eintrag.Name:20,' ',Eintrag.Size:6);
    FindNext(Eintrag);
    end;
end.
```

**Beispiel 11_3_3: Erzeugen von Kindprozessen**

Turbo-Pascal enthält ab Version 4.0 eine Prozedur EXEC, die zum Aufruf weiterer Programme genutzt werden kann. Die Anwendung der etwas komplizierten DOS-Funktion 4Bh wird dadurch sehr vereinfacht. Der Rückkehrcode wird durch die DOS-Funktion 4Dh bereitgestellt. Auch dafür gibt es in Turbo-Pascal eine Funktion DosExitCode, die diesen Wert liefert. Das folgende Beispiel zeigt die Anwendung der Funktionen. Zu beachten ist, daß mit der Direktive $M die Größe des HEAP begrenzt werden muß, um Platz für das gerufene Programm frei zu lassen.

```
{Aufruf eines Kind-Prozesses mit Turbo-Pascal}
program bsp11_3_3(input,output);

{$M $2000,0,0}
uses DOS;
```

```
var
  progname, progparm:string;

begin
  write('Programm: '); readln(progname);
  write('Progparm: '); readln(progparm);
  EXEC(progname,progparm);
  if doserror<>0 then writeln('Fehlercode: ', Doserror)
                 else writeln('Returncode: ', Dosexitcode);
end.
```

**Beispiel 11_3_4: TSR-Routinen (terminate and stay resident)**

Eine sehr interessante Variante der Prozeßverwaltung ist mit der Abkürzung TSR (termiante and stay resident) verbunden. TSR-Programme sind als Unterstützung der Arbeit mit MS-DOS sehr verbreitet (z.B. SideKick und SuperKey von Borland, Uhrzeitanzeige u.a.). Grundlage dieser Arbeitsweise ist die Möglichkeit, Programme speicherresident zu beenden und an bestimmte Bedingungen zur Reaktivierung zu binden (z.B. Timerinterrupt, spezielle Tastenkombinationen). Wichtigste DOS-Funktion für diese Zwecke ist die Funktion 3200h (keep process). Das Programm sollte nur den unbedingt benötigten Speicher reservieren und den Rest freigeben. Es sind eine Reihe von Bedingungen einzuhalten, damit eine störungsfreie Arbeit gesichert ist, da MS-DOS nicht reentrant programmiert ist und von zwei Programmen nicht unabhängig voneinander genutzt werden kann. Von Turbo-Pascal wird für die TSR-Programmierung eine Prozedur KEEP bereitgestellt. Bei der Anbindung an spezielle Interrupts (Tastatur, Timer) ist darauf zu achten, das die aktiven Interruptroutinen zusätzlich aufgerufen werden (Kettung der Interruptroutinen).

# 12. Alternativen zu MS-DOS

MS-DOS ist durch die Implementierung als PC-DOS auf dem IBM-PC zum wichtigsten Betriebssystem für PC geworden. Es gibt außerdem eine Reihe anderer Betriebssysteme, die auf PC genutzt werden können und unter bestimmten Bedingungen eine gute Alternative darstellen. Das für 8-Bit-Rechner so bekannte CP/M ist auf PC praktisch bedeutungslos geworden. Durch gestiegene Leistungsfähigkeit der PC (höhere Taktfrequenz, 80286, 80386) besteht auch der Wunsch nach einer Nutzung eines PC durch mehrere Anwender gleichzeitig (multi-user, time-sharing) und dem Ablauf mehrerer Programme (multi-tasking). MS-DOS kann diesen Anforderungen nicht gerecht werden, da die Systemkonzeption darauf nicht eingestellt ist. Mehrere Versuche, MS-DOS zu einem entsprechenden System zu erweitern, haben letztlich zu einem völlig neuen Systemkonzept geführt, das im **OS/2** (Microsoft, IBM) für die neue PS/2-Serie von IBM bzw. andere Rechner auf der Basis des 80286 und 80386 realisiert wurde. Für diese Rechner wird sicher in Zukunft das OS/2 zum neuen Standardbetriebssystem werden. Für alle Rechner mit 8086/8088 wird MS-DOS weiterhin voll genutzt werden. Die derzeit (1988) neueste Version von MS-DOS ist Version 3.3, die einige neue Kommandos zur Verbesserung des Dateizugriffs (FASTOPEN, Unterstützung von Disketten mit 3 1/2 Zoll), zur besseren Anpassung an nationale Besonderheiten (NLS, DISPLAY.SYS, KEYBOARD.SYS) und Erweiterungen im COMMAND.COM (neue CALL-Funktion zum Aufruf weiterer BATCH-Dateien) u.a. enthält.

Als Standardbetriebssystem für Mehrnutzerbetrieb (multi-user) ist **UNIX** natürlich auch auf PC implementiert. Von Microsoft gibt es das System **XENIX**, das eine spezielle Implementierung von UNIX ist. Auf PC/XT ist eine effektive Arbeit im Mehrnutzerbetrieb nicht realistisch, auf PC/AT mit 80286 bzw. 80386 dagegen durchaus möglich. Die guten Graphikeigenschaften des PC kommen aber auf normalen Bedienterminals nicht zur Geltung, da dann keine BIOS-Funktionen oder direkte Bildspeicheroperationen nutzbar sind. Durch neue Graphikkarten (z.B. mit dem Graphikcontroller 82786 von Intel) lassen sich aber die gleichen Leistungen wie auf anderen Graphikworkstations erreichen. Für die Arbeit von PC im Verbund mit anderen UNIX-Rechnern ist dadurch auch die Nutzung von UNIX-Systemen auf PC sehr interessant. Die DOS-Funktionen zur Dateiverwaltung sind im MS-DOS stark an die Möglichkeiten von UNIX/XENIX angelehnt. Auch die Bibliotheken des C-Systems von Microsoft enthalten sehr viel UNIX-Philosophie. Trotzdem hat sich um MS-DOS eine eigene, sehr große Programmbibliothek gebildet, die eine eigenständige Linie darstellt.

Neben MS-DOS gibt es eine ganze Familie von Betriebssystemen der Firma Digital Research, die den gleichen Nutzerkreis wie bei

MS-DOS ansprechen wollen. Diese Systeme versuchen, die Programmwelt von CP/M und MS-DOS unter einem gemeinsamen Betriebssystem zu vereinen. Mit **concurrent DOS** (CDOS) existieren leistungsfähige Versionen für Systeme auf der Basis des 80386 (concurrent DOS 386) und 8086/80286 (concurrent DOS XM). CDOS erlaubt den parallelen Betrieb mehrerer DOS-Programme und spezieller CDOS-Applikationen, wie z.B. Datenbankverwaltung, Textverarbeitung und Projektmanagement. Es sind bis zu 255 Tasks gestattet, die durch Intertakskommunikation Informationen austauschen, gemeinsamen Speicher benutzen und Dateien mit einem Paßwort schützen können. Die DOS-Applikationen sind zu MS-DOS Version 3.3 kompatibel.

Echte Real-Time-Systeme, wie z.B. RMX-86 von Intel, haben auf PC bisher keine Verbreitung gefunden. In OEM-Produkten wird jedoch zunehmend auch PC-Technik eingesetzt (Industrie-PC für Automatisierungsaufgaben, Zell- und Leitrechner u.a.). Diesen Anforderungen versucht z.B. das System **FlexOS** von Digital Research nachzukommen. Es handelt sich dabei um eine Betriebssystemfamilie, die für Echtzeit, Multi-user und Multi-tasking ausgelegt ist. Als Bestandteile sind ein Echtzeitkern, eventgesteuerter Dispatcher, Multi-user-Dateistruktur, DOS-Kompatibilität, Graphikfunktionen und Netzwerkfähigkeit zu nennen. Zusammen mit einem Developer-Kit ist es besonders für OEM-Anwendungen gedacht.

Ein Betriebssystem, das selbst nicht zu MS-DOS kompatibel ist, DOS-Programme aber als Task emulieren kann, ist das System **QNX** (Quantum, Repas). Durch Ausnutzung spezieller Eigenschaften des 80286 sind schnelle Umschaltzeiten zwischen einzelnen Tasks möglich, und daher ist QNX auch als Echtzeitsystem verwendbar.

Mehrnutzerbetrieb ist auch mit einer speziellen Version **WINDOWS 386** (Microsoft) möglich. Dieses Programmsystem nutzt die Eigenschaft des 80386 aus, mehrere 8086 auf einem Prozessor nachzubilden. Ebenfalls für den parallelen Ablauf mehrerer DOS-Programme gedacht sind die Systeme **MultiLink** und **PC-MOS 386** der Firma Softwarelink. Durch Aufteilung des verfügbaren Speichers in mehrere Bereiche (partitions) und zeitgesteuerte Umschaltung sind bis zu 9 Programme gleichzeitig ablauffähig. Die Programme werden entweder über das Bedienterminal am PC gesteuert (Umschaltung der Bildschirmanzeigen ist durch Tastendruck möglich) oder über seriell angeschlossene Zusatzterminals. Durch entsprechende Software (LANLINK) ist auch die Verbindung zu anderen PC im Servermodus möglich.

Die kurze Aufzählung von Alternativen zu MS-DOS, deren zusätzlichen Eigenschaften und auch Begrenzungen macht deutlich, welche wichtige Stellung MS-DOS einnimmt und wo Verbesserungen möglich sind. Als interessanteste Alternative zu MS-DOS ist wohl das OS/2 anzusehen. Alle neuen Möglichkeiten des Prozessors 80386 werden aber auch dabei schon nicht ausgenutzt, so daß Spielraum für Nachfolgeversionen und weitere Alternativen vorhanden ist.

# Anhang

## A1. ASCII-Code

(DEZ = dezimal, HX = hexadez., ASC = ASCII, IBM = erweiterter ASCII)

| DEZ | HX | ASC |
|---|---|---|
| 0 | 00 | NUL |
| 1 | 01 | SOH |
| 2 | 02 | STX |
| 3 | 03 | ETX |
| 4 | 04 | EOT |
| 5 | 05 | ENQ |
| 6 | 06 | ACK |
| 7 | 07 | BEL |
| 8 | 08 | BS |
| 9 | 09 | HT |
| 10 | 0A | LF |
| 11 | 0B | VT |
| 12 | 0C | FF |
| 13 | 0D | CR |
| 14 | 0E | SO |
| 15 | 0F | SI |
| 16 | 10 | DLE |
| 17 | 11 | DC1 |
| 18 | 12 | DC2 |
| 19 | 13 | DC3 |
| 20 | 14 | DC4 |
| 21 | 15 | NAK |
| 22 | 16 | SYN |
| 23 | 17 | ETB |
| 24 | 18 | CAN |
| 25 | 19 | EM |
| 26 | 1A | SUB |
| 27 | 1B | ESC |
| 28 | 1C | FS |
| 29 | 1D | GS |
| 30 | 1E | RS |
| 31 | 1F | US |
| 32 | 20 | SP |
| 33 | 21 | ! |
| 34 | 22 | " |
| 35 | 23 | # |
| 36 | 24 | $ |
| 37 | 25 | % |
| 38 | 26 | & |
| 39 | 27 | ' |
| 40 | 28 | ( |
| 41 | 29 | ) |
| 42 | 2A | * |
| 43 | 2B | + |
| 44 | 2C | , |
| 45 | 2D | - |
| 46 | 2E | . |
| 47 | 2F | / |
| 48 | 30 | 0 |
| 49 | 31 | 1 |
| 50 | 32 | 2 |
| 51 | 33 | 3 |
| 52 | 34 | 4 |
| 53 | 35 | 5 |
| 54 | 36 | 6 |
| 55 | 37 | 7 |
| 56 | 38 | 8 |
| 57 | 39 | 9 |
| 58 | 3A | : |
| 59 | 3B | ; |
| 60 | 3C | < |
| 61 | 3D | = |
| 62 | 3E | > |
| 63 | 3F | ? |
| 64 | 40 | @ |
| 65 | 41 | A |
| 66 | 42 | B |
| 67 | 43 | C |
| 68 | 44 | D |
| 69 | 45 | E |
| 70 | 46 | F |
| 71 | 47 | G |
| 72 | 48 | H |
| 73 | 49 | I |
| 74 | 4A | J |
| 75 | 4B | K |
| 76 | 4C | L |
| 77 | 4D | M |
| 78 | 4E | N |
| 79 | 4F | O |
| 80 | 50 | P |
| 81 | 51 | Q |
| 82 | 52 | R |
| 83 | 53 | S |
| 84 | 54 | T |
| 85 | 55 | U |
| 86 | 56 | V |
| 87 | 57 | W |
| 88 | 58 | X |
| 89 | 59 | Y |
| 90 | 5A | Z |
| 91 | 5B | [ |
| 92 | 5C | \ |
| 93 | 5D | ] |
| 94 | 5E | ^ |
| 95 | 5F | _ |
| 96 | 60 | ` |
| 97 | 61 | a |
| 98 | 62 | b |
| 99 | 63 | c |
| 100 | 64 | d |
| 101 | 65 | e |
| 102 | 66 | f |
| 103 | 67 | g |
| 104 | 68 | h |
| 105 | 69 | i |
| 106 | 6A | j |
| 107 | 6B | k |
| 108 | 6C | l |
| 109 | 6D | m |
| 110 | 6E | n |
| 111 | 6F | o |
| 112 | 70 | p |
| 113 | 71 | q |
| 114 | 72 | r |
| 115 | 73 | s |
| 116 | 74 | t |
| 117 | 75 | u |
| 118 | 76 | v |
| 119 | 77 | w |
| 120 | 78 | x |
| 121 | 79 | y |
| 122 | 7A | z |
| 123 | 7B | { |
| 124 | 7C | \| |
| 125 | 7D | } |
| 126 | 7E | ~ |
| 127 | 7F | DEL |

| DEZ | HX | IBM |
|---|---|---|
| 128 | 80 | Ç |
| 129 | 81 | ü |
| 130 | 82 | é |
| 131 | 83 | â |
| 132 | 84 | ä |
| 133 | 85 | à |
| 134 | 86 | å |
| 135 | 87 | ç |
| 136 | 88 | ê |
| 137 | 89 | ë |
| 138 | 8A | è |
| 139 | 8B | ï |
| 140 | 8C | î |
| 141 | 8D | ì |
| 142 | 8E | Ä |
| 143 | 8F | Å |
| 144 | 90 | É |
| 145 | 91 | æ |
| 146 | 92 | Æ |
| 147 | 93 | ô |
| 148 | 94 | ö |
| 149 | 95 | ò |
| 150 | 96 | û |
| 151 | 97 | ù |
| 152 | 98 | ÿ |
| 153 | 99 | Ö |
| 154 | 9A | Ü |
| 155 | 9B | ¢ |
| 156 | 9C | £ |
| 157 | 9D | ¥ |
| 158 | 9E | ₧ |
| 159 | 9F | ƒ |
| 160 | A0 | á |
| 161 | A1 | ü |
| 162 | A2 | ó |
| 163 | A3 | ú |
| 164 | A4 | ä |
| 165 | A5 | Ñ |
| 166 | A6 | ª |
| 167 | A7 | º |
| 168 | A8 | ¿ |
| 169 | A9 | ⌐ |
| 170 | AA | ¬ |
| 171 | AB | ½ |
| 172 | AC | ¼ |
| 173 | AD | ¡ |
| 174 | AE | Ä |
| 175 | AF | » |
| 176 | B0 | ▒ |
| 177 | B1 | ▒ |
| 178 | B2 | ▓ |
| 179 | B3 | ┤ |
| 180 | B4 | ┤ |
| 181 | B5 | ╡ |
| 182 | B6 | ╢ |
| 183 | B7 | ╖ |
| 184 | B8 | ╕ |
| 185 | B9 | ╣ |
| 186 | BA | ║ |
| 187 | BB | ╗ |
| 188 | BC | ╝ |
| 189 | BD | ╜ |
| 190 | BE | ╛ |
| 191 | BF | ┐ |
| 192 | C0 | └ |
| 193 | C1 | ┴ |
| 194 | C2 | ┬ |
| 195 | C3 | ├ |
| 196 | C4 | ─ |
| 197 | C5 | ┼ |
| 198 | C6 | ╞ |
| 199 | C7 | ╟ |
| 200 | C8 | ╚ |
| 201 | C9 | ╔ |
| 202 | CA | ╩ |
| 203 | CB | ╦ |
| 204 | CC | ╠ |
| 205 | CD | ═ |
| 206 | CE | ╬ |
| 207 | CF | ╧ |
| 208 | D0 | ╨ |
| 209 | D1 | ╤ |
| 210 | D2 | ╥ |
| 211 | D3 | ╙ |
| 212 | D4 | ╘ |
| 213 | D5 | ╒ |
| 214 | D6 | ╓ |
| 215 | D7 | ╫ |
| 216 | D8 | ╪ |
| 217 | D9 | ┘ |
| 218 | DA | ┌ |
| 219 | DB | ▉ |
| 220 | DC | ▄ |
| 221 | DD | ▌ |
| 222 | DE | ▐ |
| 223 | DF | ▀ |
| 224 | E0 | α |
| 225 | E1 | ß |
| 226 | E2 | Γ |
| 227 | E3 | π |
| 228 | E4 | Σ |
| 229 | E5 | σ |
| 230 | E6 | µ |
| 231 | E7 | τ |
| 232 | E8 | Φ |
| 233 | E9 | Θ |
| 234 | EA | Ω |
| 235 | EB | δ |
| 236 | EC | ∞ |
| 237 | ED | φ |
| 238 | EE | ε |
| 239 | EF | ∩ |
| 240 | F0 | ≡ |
| 241 | F1 | ± |
| 242 | F2 | ≥ |
| 243 | F3 | ≤ |
| 244 | F4 | ⌠ |
| 245 | F5 | ⌡ |
| 246 | F6 | ÷ |
| 247 | F7 | ≈ |
| 248 | F8 | ° |
| 249 | F9 | ∙ |
| 250 | FA | · |
| 251 | FB | √ |
| 252 | FC | $^{n}$ |
| 253 | FD | $^{2}$ |
| 254 | FE | ■ |
| 255 | FF | |

## A2. SCAN-Code der Tastatur

Im Abschn. 4.3.4. sind die folgenden erweiterten Tastencodes erläutert. Eine Routine zur Anzeige eines Tastencodes ist im Abschnitt 11. enthalten.

```
03 Ctrl-2 (Null)
15 Shift-Tab

Alternative Buchstaben- und Zifferntasten
-------------------------------------------------------------------------
16 Alt-Q          30 Alt-A          44 Alt-Z          120 Alt-1
17 Alt-W          31 Alt-S          45 Alt-X          121 Alt-2
18 Alt-E          32 Alt-D          46 Alt-C          122 Alt-3
19 Alt-R          33 Alt-F          47 Alt-V          123 Alt-4
20 Alt-T          34 Alt-G          48 Alt-B          124 Alt-5
21 Alt-Y          35 Alt-H          49 Alt-N          125 Alt-6
22 Alt-U          36 Alt-J          50 Alt-M          126 Alt-7
23 Alt-I          37 Alt-K                            127 Alt-8
24 Alt-O          38 Alt-L                            128 Alt-9
25 Alt-P                                              129 Alt-0
                                                      130 Alt-
                                                      131 Alt-
Modifikationen der Funktionstasten
-------------------------------------------------------------------------
59 F1             84 Shift-F1       094 Ctrl-F1       104 Alt-F1
60 F2             85 Shift-F2       095 Ctrl-F2       105 Alt-F2
61 F3             86 Shift-F3       096 Ctrl-F3       106 Alt-F3
62 F4             87 Shift-F4       097 Ctrl-F4       107 Alt-F4
63 F5             88 Shift-F5       098 Ctrl-F5       108 Alt-F5
64 F6             89 Shift-F6       099 Ctrl-F6       109 Alt-F6
65 F7             90 Shift-F7       100 Ctrl-F7       110 Alt-F7
66 F8             91 Shift-F8       101 Ctrl-F8       111 Alt-F8
67 F9             92 Shift-F9       102 Ctrl-F9       112 Alt-F9
68 F10            93 Shift-F10      103 Ctrl-F10      113 Alt-F10

Cursortasten und Modifikationen
-------------------------------------------------------------------------
                                    114 Ctrl-PrtSc
71 Home                             119 Ctrl-Home
72 Up Arrow
73 PgUp                             132 Ctrl-PgUp
75 Left Arrow                       115 Ctrl-Left Arrow
77 Right Arrow                      116 Ctrl-Right Arrow
79 End                              117 Ctrl-End
80 Down Arrow
81 PgDn                             118 Ctrl-PgDn
82 Ins
83 Del
```

## A3. Steuersequenzen des ANSI-Treibers (ANSI.SYS)

Der ANSI-Treiber enthält Funktionen zur Bildschirm- und Tastaturansteuerung, die durch spezielle ESC-Sequenzen aufgerufen werden.

**Funktionen zur Bildschirmsteuerung**

**Cursorsteuerung**

```
ESC[nA  (CUU cursor up)
        Cursor n Zeilen nach oben
ESC[nB  (CUD cursor down)
        Cursor n Zeilen nach unten
ESC[nC  (CUF cursor forward)
        Cursor n Spalten nach rechts
ESC[nD  (CUB cursor backward)
        Curosr n Spalten nach links
ESC[x;yH oder ESC[x;yF (CUP cursor position)
        Cursor auf Position x;y stellen
ESC[s   (SCP save cursor position)
        Cursorposition merken
ESC[u   (RCP restore cursor position)
        Cursorposition zurückspeichern
ESC[6n  (DSR device status report)
        CPR-Sequenz ausgeben
ESC[x;yR (CPR cursor position report)
        CPR-Sequenz auf stdin
```

**Löschfunktionen**

```
ESC[2J  (ED erase display)
        Bildschirm löschen
ESC[K   (EL erase line)
        Zeile ab Cursor löschen
```

**Bildschirmmodus und Graphikparameter**

```
ESC[#;#m (SGR set graphik rendition)
        Zeichenattribute <#> einstellen
ESC[=#h oder ESC[=#l (SM set mode)
        Bildschirmmodus einstellen (vgl. Tafel 4.3)
```

**Definition von Tastaturmakros**

```
ESC[#;"string"p (assign key)
        Tastenzuordnung zwischen Scancode <#> und
        <"string">
```

## A4. BIOS-Funktionen

**BIOS-Funktionen**

| Nr. (hex.) | Funktion |
|---|---|
| **10-nn** | **Bildschirmfunktionen** |
| 10-00 | Bildschirmmodus einstellen |
| 10-01 | Cursorgröße festlegen |
| 10-02 | Cursorposition setzen |
| 10-03 | Cursorposition abfragen |
| 10-04 | Lichtstiftposition abfragen |
| 10-05 | Bildschirmseite festlegen |
| 10-06 | Fenster nach oben rollen |
| 10-07 | Fenster nach unten rollen |
| 10-08 | Zeichen/Attribut an Cursorposition lesen |
| 10-09 | Zeichen/Attribut an Cursorposition schreiben |
| 10-0A | Zeichen an Cursorposition schreiben |
| 10-0B | Farbpalette festlegen |
| 10-0C | Graphikpunkt setzen |
| 10-0D | Graphikpunkt abfragen |
| 10-0E | Zeichen ausgeben und Cursor weiterbewegen |
| 10-0F | aktuellen Bildschirmmodus ermitteln |
| 10-10 | EGA-Farbpalettenregister setzen |
| 10-11 | EGA-Zeichengeneratorroutine |
| 10-12 | EGA-Konfigurierung |
| 10-13 | Zeichenkette (string) schreiben |
| **13-nn** | **Disketten- und Festplattenfunktionen** |
| 13-00 | Diskettenlaufwerke zurücksetzen |
| 13-01 | Diskettenstatus ermitteln |
| 13-02 | Sektor lesen (read) |
| 13-03 | Sektor schreiben (write) |
| 13-04 | Sektor verifizieren (verify) |
| 13-05 | Spur formatieren |
| **14-nn** | **Funktionen für die serielle Schnittstelle** |
| 14-00 | Initialisierung der COM-Schnittstelle |
| 14-01 | Zeichen senden |
| 14-02 | Zeichen empfangen |
| 14-03 | COM-Status ermitteln |
| **16-nn** | **Tastaturfunktionen** |
| 16-00 | Tastencode aus Tastaturpuffer lesen |
| 16-01 | Status Tastaturpuffer feststellen |
| 16-02 | Umschaltstatus feststellen |
| **17-nn** | **Druckerfunktionen** |
| 17-00 | ein Byte an Drucker senden |
| 17-01 | Drucker initialisieren |
| 17-02 | Druckerstatus ermitteln |
| **19-nn** | **Bootstrap** |

## A5. DOS-Funktionen

| DOS-Interrupts | |
|---|---|
| **Nr.** (hex.) | **Funktion** |
| 20 | Programm beenden |
| 21-nn | DOS-Funktion aufrufen |
| 22 | Adresse Programmendebehandlung |
| 23 | Adresse Unterbrechungsbehandlung (Ctrl-C) |
| 24 | Adresse Fehlerbehandlung |
| 25 | Diskette/Festplatte lesen (absolut) |
| 26 | Diskette/Festplatte schreiben (absolut) |
| 27 | Programm speicherresident beenden |
| 2F | Druckerspooler |
| 33 | Mausroutinen |

| DOS-Funktionen (Int 21-nn) | |
|---|---|
| **Nr.** (hex.) | **Funktion** |
| 00 | Programm beenden |
| 01 | Zeichen von Standardeingabe lesen |
| 02 | Zeichen auf Standardausgabe schreiben |
| 03 | Zeichen von Hilfsport lesen |
| 04 | Zeichen nach Hilfsport schreiben |
| 05 | Zeichen auf Drucker schreiben |
| 06 | direkte Ein-/Ausgabe auf stdin/stdout |
| 07 | Zeichen direkt von Standardeingabe lesen (warten) |
| 08 | Zeichen von Standardeingabe lesen |
| 09 | String auf Standardausgabe schreiben |
| 0A | String von Standardeingabe lesen |
| 0B | Status Eingabepuffer testen |
| 0C | Puffer leeren und Eingabefunktion ausführen |
| 0D | Laufwerk zurücksetzen, Puffer leeren |
| 0E | aktuelles Laufwerk einstellen |
| 0F | traditionelle Dateiverwaltung mit FCB |
| ... | ... |
| 17 | traditionelle Dateiverwaltung mit FCB |
| 19 | aktuelles Laufwerk feststellen |
| 1A | DTA-Adresse setzen |
| 1B | Parameter von aktuellem Laufwerk lesen |
| 1C | Parameter von beliebigem Laufwerk lesen |
| 21 | traditionelle Dateiverwaltung mit FCB |
| ... | ... |
| 24 | traditionelle Dateiverwaltung mit FCB |
| 25 | Interruptvektor setzen |
| 26 | Erzeugen eines Programmsegmentpräfix |
| 27 | wahlfreies Lesen eines Blockes mit FCB |
| 28 | wahlfreies Schreiben eines Blockes mit FCB |

| DOS-Funktionen (Fortsetzung) | |
|---|---|
| 29 | Dateinamen in String suchen |
| 2A | Datum lesen |
| 2B | Datum setzen |
| 2C | Zeit lesen |
| 2D | Zeit setzen |
| 2E | Status für Verifizieren ändern |
| 2F | DTA-Adresse lesen |
| 30 | Version lesen |
| 31 | Prozeß speicherresident beenden |
| 33 | Status Ctrl-C lesen/setzen |
| 35 | Interruptvektor lesen |
| 36 | Platz auf Laufwerk lesen |
| 38 | Länderspezifik |
| 39 | Verzeichnis anlegen |
| 3A | Verzeichnis löschen |
| 3B | Verzeichnis wechseln |
| 3C | Kanal anlegen |
| 3D | Kanal eröffnen |
| 3E | Kanal schließen |
| 3F | Kanal lesen |
| 40 | Kanal schreiben |
| 41 | Eintrag löschen |
| 42 | Kanalposition verändern |
| 43 | Attribute setzen/lesen |
| 44 | IOCTL |
| 45 | Kanal duplizieren |
| 46 | Kanal kopieren |
| 47 | aktuelles Verzeichnis melden |
| 48 | Speicher anfordern |
| 49 | Speicher freigeben |
| 4A | Speicheranforderung ändern |
| 4B | Programm laden |
| 4C | Prozeß beenden |
| 4D | Returncode von Kindprozeß lesen |
| 4E | ersten Eintrag suchen |
| 4F | nächsten Eintrag suchen |
| 54 | Status für Verifizieren lesen |
| 56 | Dateinamen ändern |
| 57 | Datum/Uhrzeit im Verzeichnis setzen/lesen |
| 58 | Zuteilungsstrategie setzen/lesen |
| 59 | erweiterte Fehlerbehandlung |
| 5A | Kanal temporär anlegen |
| 5B | Kanal neu anlegen |
| 5C | Dateibereich sperren/freigeben |
| 5E | spezielle Netzwerkfunktionen |
| 5F | Verwaltung der Netzzuweisungsliste |
| 62 | Programmsegmentpräfix lesen |

# A6. DOS-Fehlercodes

### Standardfehler (standard error codes)

| Nr. | Bedeutung |
|---|---|
| 01 | Invalid function code |
| 02 | File not found |
| 03 | Path not found |
| 04 | Too many open files |
| 05 | Access denied |
| 06 | Invalid handle |
| 07 | Memory control blocks destroyed |
| 08 | Insufficient memory |
| 09 | Invalid memory-block address |
| 10 | Invalid environment |
| 11 | Invalid format |
| 12 | Invalid access code |
| 13 | Invalid data |
| 15 | Invalid drive |
| 16 | Attempt to remove current dir |
| 17 | Not same drive |
| 18 | No more files |
| 19 | Disk is write-protected |
| 20 | Bad disk unit |

| Nr. | Bedeutung |
|---|---|
| 21 | Drive not ready |
| 22 | Invalid disk command |
| 23 | CRC error |
| 24 | Invalid length (disk operation) |
| 25 | Seek error |
| 26 | Not an MS-DOS disk |
| 27 | Sector not found |
| 28 | Out of paper |
| 29 | Write fault |
| 30 | Read fault |
| 31 | General failure |
| 32 | Sharing violation |
| 33 | Lock violation |
| 34 | Wrong disk |
| 35 | FCB unavailable |
| 50 - 88 | Network error |

### Erweiterte Fehlercodes (extended error codes)

Fehler werden nach DOS-Funktion 59h in Registern abgelegt:

BH Fehlerklasse (error class)
BL vorgeschlagene Aktion (suggested action)
CH Fehlerort (locus).

| BH | Fehlerklasse |
|---|---|
| 01 | Out of a resource |
| 02 | Not an error (temporary situation) |
| 03 | Authorization problem |
| 04 | An internal error in system SW |
| 05 | Hardware failure |
| 06 | A system software failure |
| 07 | Application program error |
| 08 | File or item not found |
| 09 | File or item of invalid format |
| 10 | File or item interlocked |
| 11 | Wrong disk (storage problems) |
| 12 | Other error |

| BL | Vorgeschlagene Aktion |
|---|---|
| 01 | Retry, then prompt user |
| 02 | Retry after a pause |
| 03 | Prompt for again |
| 04 | Terminate with clean up |
| 05 | Terminate immediately |
| 06 | Error is informational |
| 07 | Prompt the user for action, retry |

| CH | Fehlerort |
|---|---|
| 01 | Unknown |
| 02 | Related to random-access block dev |
| 03 | Related to Network |
| 04 | Related to serial-access char. dev |
| 05 | Related to random-access memory |

## A7. DOS-Kommandos, Editier- und Funktionstasten

| | Seite |
|---|---|
| **APPEND** [d:][Pfad][; ...] | 151 |
| **ASSIGN** [d1=d2 ...] | 138 |
| **ATTRIB** [+R \| -R] [+A \| -A] Dateiname | 147 |
| **BACKUP** [Quelle:][Dateiname] Ziel: /Optionen<br>/A Anfügen /L:datei Log-Datei anlegen<br>/D:datum ab <datum> /M Veränderungen<br>/S Unterverzeichnis<br>/F Formatieren /T:zeit ab <zeit> | 144 |
| **BREAK** on \| off | 125 |
| **CD** [d:][Pfad]<br>**CHDIR** | 148 |
| **CHKDSK** [d:][Dateiname] /Optionen<br>/F Fehlerkorr. /V Fehleranzeige | 142 |
| **CLS** | 132 |
| **COMMAND** [d:][Pfad][ctty] /Optionen<br>/D kein Datum/Zeit /P unterbindet EXIT<br>/E:n Größe Environment /CKommando Ausführung Komm.<br>/F Rückkehr bei Fehler | 128 |
| **COMP** [d:]Dateiname1 [d:]Dateiname2 | 158 |
| **COPY** [d:]Dateiname[+...] [d:][Dateiname]<br>/A Textdateien (ASCII) /V Verifizieren<br>/B Binärdateien | 157 |
| **CTTY** AUX \| COM1 \| COM2 \| CON | 126 |
| **DATE** [Datum] | 124 |
| **DEL** [d:]Dateiname<br>**ERASE** | 159 |
| **DIR** [d:][Pfad \| Dateiname] /Optionen<br>/P Pause nach Seite /W breite Ausgabe | 149 |

| | |
|---|---|
| **DISKCOMP** [d1:] [d2:] /Optionen<br><br>/1 Vergleich 1.Seite<br>/8 8 Sektoren/Spur | 143 |
| **DISKCOPY** [d1:] [d2:] /Optionen<br><br>/1 Vergleich 1.Seite | 143 |
| **ECHO** on \| off \| [Text] | 162 |
| **EXE2BIN** [d:]Dateiname1 [d:]Dateiname2 | 184 |
| **EXIT** | 129 |
| **FDISK** (menügesteuert) | 138 |
| **FIND** "text" /Optionen [d:][Dateiname]<br><br>/C Anzahl mit "Text" /N Zeilennr. ausgeben<br>/V Anzahl ohne "Text" | 154 |
| **FORMAT** [d:] /Optionen<br><br>/1 einseitiges Format /B Systemplatz reservieren<br>/4 Format 360 KByte /T:n Anzahl Spuren<br>/8 8 Sektoren/Spur /N:n Sektoren/Spur<br>/S System übertragen /V Datenträgerkennung | 140 |
| **GRAFTABL** optionen<br><br>/? Optionenliste /STA Anzeige Zeichensatz<br>nnn Tabellen-Nr. | 132 |
| **GRAPHICS** <drucker> /Optionen<br><br><drucker> /B Hintergrund farbig<br>COLOR1 /R Inversdruck<br>COLOR4<br>COLOR8<br>COMPACT<br>GRAPHICS (Standard) | 136 |
| **JOIN** [d:][Pfad] /Option<br><br>/D Verknüpfung löschen | 150 |
| **KEYB xx**<br>xx Länderkennung z.B. US - USA, GR - deutsch | 130 |
| **LABEL** [d:][Datenträgerkennung] | 140 |
| **MD** [d:]Pfad<br>**MKDIR** | 147 |

**MODE** COMn:Baud[,[Pari][,[Daten][,[Stop][,P]]]] 134
LPTn[:][80|132][,6|8][,P] 135
LPTn[:]=COMm[:]
40|80|BW40|BW80|CO40|CO80|MONO 132

**MORE** (Filter) 155

**PATH** [d:][Pfad][;...] 152

**PAUSE** [Text] 165

**PRINT** [d:][Dateiname] [...] /Optionen 136, 156

/B:n Puffergröße /Q:n Anzahl Dateien (1...32)
/C Löschmodus ein /S:n Zeitscheibe
/D:<gerät> Ausgabegerät /T alle Dateien löschen
/M:n Druckzeiteinheit /U:n Wartezeit bis bereit
/P Druckmodus ein

**PROMPT** <Text> 127

<Text> $$ Zeichen '$' $H Rückschritt (Backspace)
$_ CR/LF $L Zeichen '<'
$B Zeichen '|' $N aktuelles Laufwerk
$D Datum $P aktuelles Verzeichnis
$E ESC (X'1B') $Q Zeichen '='
$G Zeichen '>' $T Uhrzeit
$V DOS-Version

**RECOVER** [d:][Dateiname] 142

**RD** [d:]Pfad 148
**RMDIR**

**REN** [d:]Dateialt Dateineu 158
**RENAME**

**REPLACE** [d:]Dateiname [d:][Pfad] /Optionen

/A Hinzufügen /R Überschreiben read-only
/P Bestätigung /S Suchen in Unterverzeichn.
/W Diskettenwechsel

**RESTORE** d1: d2:[Dateiname][+...] /Optionen 145

/A:datum nach <datum> /M nur Modifikationen
/B:datum vor <datum> /N nur neue Dateien
/E:zeit vor <zeit> /P Bestätigung
/L:zeit nach <zeit> /S alle Unterverzeichnisse

**SET** [<variable>=<wert>] 127

| | |
|---|---|
| **SHARE** /Optionen<br>/F:n Bytes für Dateien /L:n Anzahl Sperrungen | |
| **SORT** /Optionen (Filter)<br>/R rückläufig sortieren /+n Start Sortiermerkmal | 155 |
| **SUBST** [d:] [Pfad]<br>/D Substitution aufheben | 150 |
| **SYS** d: | 141 |
| **TIME** [hh:mm[:ss[.cc]]] | 125 |
| **TREE** [d:] /Option<br>/F Dateinamen ausgeben | 149 |
| **TYPE** [d:]Dateiname | 125 |
| **VER** | 124 |
| **VERIFY** [on \| off] | 126 |
| **VOL** [d:] | 141 |
| **XCOPY** [d1:]Dateiname1 [d2:]Dateiname2<br>/A Kopieren bei +A-bit /P Bestätigung<br>/D:<datum> Änderungen /S mit Unterverzeichnis<br>/E inkl. leere Verz. /V Verifizieren<br>/M wie /A Lösche A-bit /W Warten Diskettenwechsel | 159 |

**Editier- und Funktionstasten**

| | |
|---|---|
| F1 | ein Zeichen aus Puffer lesen |
| F2 | bis zum angegebenen Zeichen aus Puffer lesen |
| F3 | restliche Zeichen aus Puffer lesen |
| DEL | ein Zeichen im Puffer überspringen |
| F4 | bis zum angegebenen Zeichen überspringen |
| ESC | Eingabe löschen, Puffer wird nicht verändert |
| INS | Einfügemodus ein- bzw. ausschalten |
| F5 | aktuelle Zeile in Puffer übertragen |
| F6 | EOF (<Ctrl-Z> in Puffer schreiben |
| <Ctrl-C> | Kommando abbrechen |
| <Ctrl-P> | Druckerprotokoll ein- bzw. ausschalten |
| <Ctrl-S> | Bildschirmausgabe anhalten, weiter mit beliebiger Taste |

## A8. Punktkommandos für WordStar

| | Parameter | Funktion |
|---|---|---|
| .AV | Meldung,Variable,Format | Variablennamen lesen |
| .AW | 1/0 | Zeilenumbruch an/aus |
| .BN | 1-4 | Auswahl Papierschacht |
| .BP | 1/0 | bidirektionaler Druck an/aus |
| .CP | Zahl | bedingter Seitenumbruch |
| .CS | Meldung | Bildschirm löschen, Meldung |
| .CW | 5-24 | Schriftdichte (Zeichen/Zoll) |
| .DF | | Datendatei lesen |
| .DM | Text | Meldung beim Druck |
| .EI | | END IF |
| .EL | | ELSE |
| .FO | Text (.F1, .F2, .F3) | Fußzeilentext |
| .FI | Dateiname | Datei beim Druck einlesen |
| .FM | Zahl | Fußzeilenabstand |
| .GO | TOP / BOT | Sprung zum Dateianfang/-ende |
| .HE | Text (.H1, .H2, .H3) | Kopfzeilentext |
| .HM | Zahl | Kopfzeilenabstand |
| .IF | Bedingung | bedingte Anweisung (Mischdruck) |
| .IG | Kommentar (auch ..) | Kommentarzeile |
| .IX | Text | Zeile für Index markieren |
| .LH | 5-24 | Zeilenhöhe (9,6 2 Zeilen/Zoll) |
| .LM | Zahl | linker Rand |
| .LQ | 1/0 | Korrespondenzqualität an/aus |
| .LS | Zahl | Zeilenabstand |
| .MA | Variablenname=Formel | Gleichung für Mischdruck |
| .MB | Zahl | unterer Rand |
| .MT | Zahl | oberer Rand |
| .OJ | 1/0 | Blocksatz an/aus |
| .OP | | keine Seitennumerierung |
| .PA | | neue Seite |
| .PC | Zahl | Spalte für Seitennummer |
| .PF | 1/0/dis | Formatieren beim Druck an/aus/wahlweise |
| .PG | | Seitennumerierung einschalten |
| .PL | Zahl | Seitenlänge |
| .PM | Zahl | Festeinzug (Paragraph) |
| .PN | Zahl | Seitennummer festlegen |
| .PO | Zahl | linker Druckrand |
| .PS | 1/0 | Proportionaldruck an/aus |
| .RM | Zahl | rechter Rand |
| .RR | Zeichen | Unterlineal einfügen |
| .RV | V1,V2,... | Variablennamen einlesen |
| .SR | Zahl | Abstand Exponent/Index (in 1/48") |
| .SV | V=Text | Wertzuweisung für Variable |
| .TC | Text | Zeile für Inhaltsverzeichnis |
| .UJ | 1/0/dis | echter Blocksatz an/aus/wahlweise |
| .UL | 1/0 | Leerzeichen unterstreichen an/aus |
| .XE | Zeichenfolge | freies Steuerzeichen für ^PE |
| .XL | Zeichenfolge | Steuersequenz für Seitenvorschub |
| .XQ | Zeichenfolge | freies Steuerzeichen für ^PQ |
| .XR | Zeichenfolge | freies Steuerzeichen für ^PR |
| .XW | Zeichenfolge | freies Steuerzeichen für ^PW |

# Literatur zu MS-DOS/PC-DOS

Rector, R.; Alexy, G.: Das 8086/8088-Buch.
München: tewi-Verlag 1982

Henk, M.: Die IBM-Personal Computer.
Haar bei München: Verlag Markt & Technik 1985

Microsoft: MS-DOS 3.1 - Programmierhandbuch.
(Programmer's Reference Manual) in englischer Sprache.
Haar bei München:Verlag Markt & Technik 1986

Microsoft: MS-DOS (Versions 1.0 - 3.2).
Technical Reference Encyclopedia. Microsoft Press 1986

IBM: Personal Computer XT. Technical Reference. 1983

IBM: Disk Operating System version 3.00. Reference. 1984

Duncan, R.: MS-DOS für Fortgeschrittene.
Braunschweig: Vieweg-Verlag 1987

Norton, P.: Die verborgenen Möglichkeiten des IBM-PC.
München: Hanser-Verlag 1985

Norton, P.: Programmierhandbuch für den IBM-PC.
Braunschweig: Vieweg-Verlag 1986

Biethan, G.: MS-DOS / PC-DOS kurz und bündig.
Würzburg: Vogel-Verlag 1985

Höfs, W.: SYBEX-Ratgeber MS-DOS.
Düsseldorf: Sybex-Verlag 1986

Smode, D.: Das große MS-DOS-Profi-Arbeitsbuch.
München: Franzis Verlag 1987

Heimsoeth & Borland: Turbo Pascal.
Referenz- und Benutzerhandbuch. München 1987

Jamsa, K.; Nameroff, S.: Turbo Pascal Programmer's Library.
Berkeley: Osborne/McGraw-Hill 1987

Hartwig, O.: Turbo-Pascal für Insider.
Haar bei München: Verlag Markt&Technik 1987

Ludwigs, D.: Professionell arbeiten mit dem IBM-PC.
Würzburg: Vogel-Verlag 1985

Paulin, G.: Turbo-Pascal. Berlin: VEB Verlag Technik 1988

MicroPro: WordStar Version 4.0. Handbuch. SanRafael 1987.

Softwareführer '88 für Personal-Computer.
München: Rossipaul Verlag 1987

## Sachwörterverzeichnis

MS-DOS ist das international am meisten genutzte Betriebssystem für 16-Bit-PC. Das Buch enthält alle wichtigen Informationen für die System- und Anwendungsprogrammierung. Dazu gehören: Hardwareübersicht, BIOS- und DOS-Funktionen, Kommandos, Stapelverarbeitung, Programmierwerkzeuge und wichtige Anwendungsprogramme.

ISBN 3-341-00707-5