Der Wang Professional Computer Programmentwicklungshandbuch

VORLÄUFIGE FASSUNG

WANG

Der Wang Professional Computer Programmentwicklungshandbuch

VORLÄUFIGE FASSUNG

1. Auflage — April 1984
Copyright © Wang Laboratories, Inc., 1983
717-0007

WANG

GEWÄHRLEISTUNGSAUSSCHLUSS UND HAFTUNGSBEGRENZUNG

Die Mitarbeiter von Wang Laboratories, Inc. haben dieses Handbuch mit der gebührenden Sorgfalt erarbeitet; nichts, was in ihm enthalten ist, bewirkt jedoch in irgendeiner Weise eine Abwandlung oder Änderung der normalen Bestimmungen und Bedingungen des Wang - Kauf - Leasing - oder Lizenzvertrags, aufgrund dessen dieses Softwarepaket erworben wurde, noch bewirkt es für Wang eine größere Haftung gegenüber dem Kunden. Auf keinen Fall haften die Wang Laboratories, Inc. bzw. ihre Tochtergesellschaften für Neben- oder Folgeschäden in Zusammenhang mit dem oder durch den Gebrauch des Softwarepaketes bzw. des begleitenden Materials.

VERMERK:

Die Benutzererlaubnis für alle Wang-Programmprodukte wird den Kunden in Übereinstimmung mit den Vorschriften und Bedingungen der Programmprodukt-Standard-Lizenz von Wang Laboratories, Inc. erteilt; Eigentum an Wang-Software wird nicht übertragen und jeglicher Gebrauch, der über die Bestimmungen der besagten Lizenz hinausgeht und für den keine schriftliche Einwilligung von Wang Laboratories, Inc. vorliegt, ist untersagt.

VORWORT

Das Wang Professional Computer Programmentwicklungshandbuch ist zur Unterstützung von Programmierern mit mittleren Kenntnissen bestimmt, die den Assembler oder kompilierte Sprachen auf dem WANG PC benutzen. Diese Anleitung sollte zusammen mit dem Wang Professional Computer Einführungshandbuch (700-8005/.01) und den Handbüchern für die verschiedenen Programmiersprachen benutzt werden. Titel und Bestellnummern der einzelnen Programmiersprachen-Handbücher befinden sich im Wang Professional Computer Documentation Guide (U.S.700-7589).

Das vorliegende Handbuch enthält:

- Beschreibungen der Wang PC Programmentwicklungs-Utilities und
- für den Programmierer nützliche technische Daten und Spezifikationen.

Die einzelnen Programmentwicklungs-Utilities werden in separaten Kapiteln behandelt. Das Thema jedes Kapitels wird nachstehend kurz zusammengefaßt:

Kapitel 1 Editor: Diese Utility kann Quelldateien für Programme in kompilierten Sprachen und Assembler erstellen und modifizieren. Textdateien, z. B. Mitteilungen, Berichte und Aufstellungen, können ebenfalls erstellt und modifiziert werden.

Kapitel 2 Linker: Diese Utility verknüpft mehrere einzeln kompilierte Objektmoduln zu einer verschiebbaren Programmdatei. Der Linker löst Bezugnahmen in modulexterne Symbole auf und kann Definitionen von nicht aufgelösten Bezugnahmen in Mehrfach-Library-Dateien suchen.

Kapitel 3 Library-Verwalter: Diese Utility erstellt, modifiziert und löscht Library-Dateien, die Programm-Moduln enthalten, welche die Linker-Utility mit anderen Moduln verbinden kann. Der Library-Verwalter ermöglicht die Erstellung allgemeiner Libraries für verschiedene Programme oder besonderer Libraries für spezifische Programme.

Kapitel 4 Debugger: Diese Utility bietet eine kontrollierte Testumgebung für Binär- und ausführbare Objektdateien. Während der Debugger-Benutzung kann der Inhalt einer Datei oder eines CPU-Registers geändert und ein Programm sofort wieder ausgeführt werden, um die Änderungen zu überprüfen.

Die Anhänge enthalten technische Informationen über Systemarchitektur, DOS, Dateibearbeitung und Disk E/A, DOS Unterbrechungen und Funktionsaufrufe, BIOS-Funktionen, Geräteantriebsroutinen, die logische Tastatur, Bildschirmsteuerung, Druckersteuerung, IBM zu WANG PC Code-Konvertierung und die WANG PC Systemsoftware-Diskette.

INHALT

ANHÄNGE

ABBILDUNGEN

TABELLEN

KAPITEL 1 EDITOR

KAPITEL 1
EDITOR

1.1 EINFÜHRUNG

Dieses Kapitel beschreibt den WANG PC Editor (PCEDIT.EXE), der benutzt wird, um Quelldateien für Programme in kompilierten Sprachen und Assembler zu erstellen und zu modifizieren. Der Editor kann jedoch auch zum Erstellen und Modifizieren von Textdateien, wie Mitteilungen, Berichten und Aufstellungen verwendet werden.

Bei Ablauf des Editors enthält der Bildschirm einen sog. "Fenster"-Bereich. Zur Benutzung des Editors einen Textabschnitt mit den Cursorsteuerungs-, der VORH- bzw. NÄCHS-Taste in das Fenster verschieben. Dann den Cursor an die gewünschte Position führen und den Text innerhalb des Fensters mit Hilfe der Editor-Befehle aufbereiten.

1.2 ÜBERBLICK

Dieser Abschnitt enthält eine Beschreibung des Editors, dessen Fenster, Zeilennumerierung, Speicherverbrauch und Hilfebildschirme.

1.2.1 Fenster

Das Editor-Fenster hat 21 Zeilen. Die ersten vier Zeilen sind für Kopf-, Status-, Dialoghinweis- und Tabzeile vorgesehen. Innerhalb des Fensters können zwei Aufbereitungsmodi benutzt werden: Zentriermodus und Gleitcursormodus. Im Zentriermodus ist die Cursorzeile immer in der Bildschirmmitte. Folglich kann nur der Text in der Mitte des Bildschirms aufbereitet werden. Um den Bildschirm nach oben oder unten abrollen zu lassen und eine neue Zeile in die Bildschirmmitte zu verschieben, die Nord- bzw. Südcursor-Steuerungstaste drücken. Bei Datei-Aufbereitungsbeginn gilt der Zentriermodus (Standardmodus).

Im Gleitcursormodus kann der Cursor an eine beliebige Zeile im Fenster verschoben und der Text dort aufbereitet werden. Um jeweils eine neue Zeile in das Fenster zu verschieben, den Cursor an die oberste Zeile setzen und die Nordcursor-Steuerungstaste oder den Cursor in die unterste Zeile setzen und die Südcursor-Steuerungstaste drücken. Bei beiden Modi können die vorausgehenden oder folgenden 21 Zeilen mit der VORH- bzw. NÄCHS-Taste in das Fenster verschoben werden.

1.2.2 Zeilennumerierung

Der Editor beginnt die fortlaufende Numerierung der Zeilen bei 1. Wird eine Zeile eingefügt oder gelöscht, werden die Zeilen umnumeriert. Zeilennummern sind nicht Teil des Textes, sondern dienen lediglich als Hinweis auf eine bestimmte Zeile oder Zeilengruppe. Cursorzeilennummern werden oben am Bildschirm und nicht neben den Zeilen angegeben.

1.2.3 Puffergröße

Bei Dateiaufbereitung speichert der Editor die gesamte Datei in einem Puffer. Die Größe dieses Puffers hängt von der im Betriebssystem verfügbaren Speicherkapazität und dem residenten Editorprogramm ab. Der verfügbare Speicherraum ist je nach Konfiguration des Systems unterschiedlich. Ist eine Datei für den verfügbaren Speicher zu groß, kann eine neue Datei erstellt und ein Teil der Datei mit dem Befehl TEILDATEI LADEN (siehe Abschnitt 1.5) in die neue Datei eingefügt werden.

1.2.4 Hilfe-Bildschirme

Editor-Befehle werden mit Einzel- oder Umschalt-Tastenanschlägen erteilt. Um den für einen Befehl erforderlichen Tastenanschlag zu finden, HILFE drücken. Ein Hilfe-Bildschirm, der den Tastenanschlag für jeden Befehl aufführt, erscheint auf den ersten drei Bildschirmzeilen. Ist der gesuchte Befehl nicht in der Anzeige enthalten, nochmals HILFE drücken, um einen weiteren Hilfe-Bildschirm anzuzeigen. Wiederholtes Drücken der HILFE-Taste zeigt nacheinander alle Hilfe-Bildschirme an und kehrt dann zum Ausgangsbildschirm zurück. Während der Hilfe-Anzeige bleibt das Fenster auf dem Bildschirm sichtbar, und Befehle können erteilt werden.

1.3 EDITOR AUFRUFEN

Der Editor kann vom Programmentwicklungsmenü oder vom DOS Befehlsprozessor aus aufgerufen werden. Bei der ersten Methode die Programmentwicklungsoption vom System-Hauptmenü wählen. Abbildung 1-1 zeigt das Programmentwicklungsmenü.

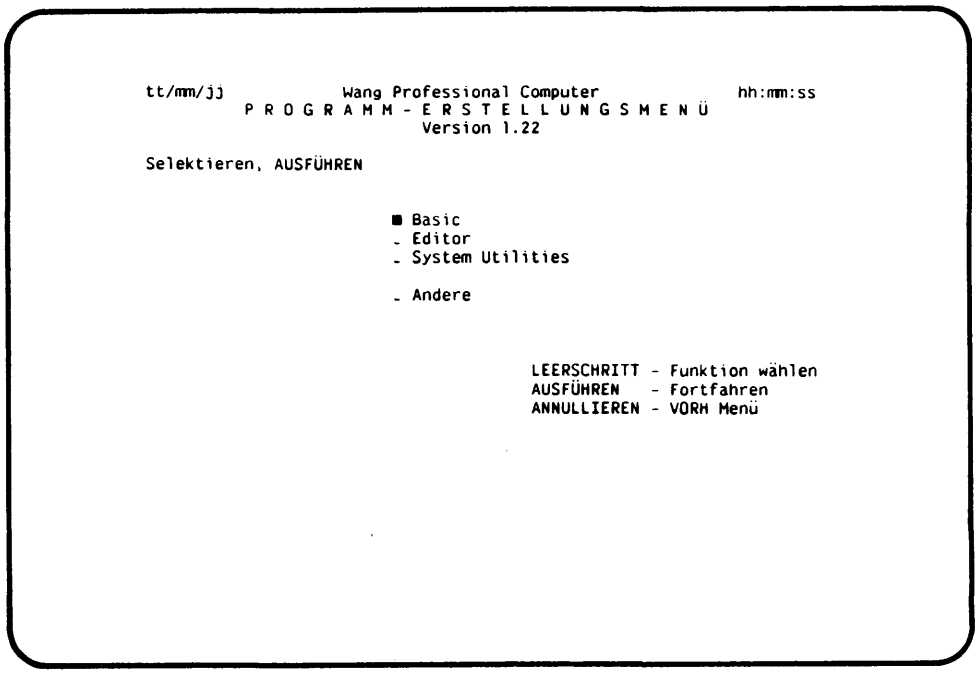


Abbildung 1-1. Programmentwicklungsmenü

Die Editor-Option vom Programmentwicklungsmenü wählen. Die Spezifikation der aufzubereitenden Datei nach dem entsprechenden Dialoghinweis eintragen.

Bei der zweiten Methode zum Aufruf des Editors die DOS Befehlsprozessor-Option im Hauptsystem-Menü wählen. Den DOS Dialoghinweis folgendermaßen beantworten:

PCEDIT [<Filespez.>]

Bei Angabe der wahlweisen Dateispezifikation lädt der Editor diese Datei automatisch. Wird die Dateispezifikation nicht angegeben, erscheint ein entsprechender Editor-Dialoghinweis.

Zur Erstellung einer Datei den Editor aufrufen und die Spezifikation für eine Datei eintragen, die nicht auf der adressierten Disk(ette) besteht. Der Editor erstellt daraufhin die Datei. (Eine Datei kann auch mit der FILE KOPIEREN System-Utility oder der DOS KOPIEREN Utility erstellt werden. Siehe Wang Professional Computer Einführungshandbuch.)

1.4 KONFIGURATIONSPARAMETER

Bei Eintrag der Dateispezifikation sucht der Editor die laufenden und alternativen Verzeichnisse nach der sog. CONFIG.EDT Datei ab. Diese Datei enthält Standard-Tab- und -Breiteeinstellungen für verschiedene Programmiersprachen. Die Dateinamenerweiterung identifiziert die Programmiersprache der Datei. Der Editor weist z. B. .ASM Dateien Standardtab-Einstellungen und -Zeilenlänge für Assembler zu. Falls CONFIG.EDT keine Werte für eine Erweiterung bestimmt, benutzt der Editor die Standardwerte 8, 8, 8, 8, 8, 8, 8, 8, 8 für Tabs und 80 für die Zeilenlänge.

ANMERKUNG:
Falls das aufzubereitende Dokument mit einer Breite von mehr als 80 erstellt wurde, muß die Standardbreite mit dem Befehl ZEILENLÄNGE FESTLEGEN (siehe Abschnitt 1.5) aufgehoben werden. Wird die Zeilenlänge nicht neu festgesetzt, werden die aufbereiteten Zeilen auf eine Länge von 80 Zeichen verkürzt. Die zulässige Zeilenlänge liegt zwischen 1 und 255.

CONFIG.EDT bestimmt ferner, ob der Editor die Datei in verdichtetem Format oder nicht verdichtetem ASCII-Format speichert und ob der Editor KTRL + Z als Endanzeige zur Datei hinzufügt. Ein verdichtetes Format benötigt weniger Diskspeicherraum, da jede Gruppe fortlaufender Leerstellen vor einer Acht-Spalten-Grenze durch einen Tab ersetzt wird. Als Standard bestimmmt CONFIG.EDT das nicht verdichtete Format.

Einige auf Wang PC laufende Programme benötigen KTRL + Z als Datei-Endanzeige. Da dies bei den meisten Programmen jedoch nicht erforderlich ist, bestimmt der von CONFIG.EDT vorgegebene Standard, daß KTRL + Z nicht an eine Datei angefügt wird.

Durch Aufbereitung können CONFIG.EDT Standardwerte geändert oder neue Standardwerte für andere Dateinamenerweiterungen hinzugefügt werden. Ein CONFIG.EDT Befehl ist eine Textfolge, die ein bis drei Zeichen enthält (Dateinamenerweiterung ohne Punkt), der eine Leerstelle, ein Schlüsselwort, ein Gleichheitszeichen und dann ein Wert oder mehrere durch Kommas getrennte Werte folgen. Schlüsselwörter sind TABS, WIDTH, COMPRESS und CTRLZ. Die Werte für TAB und WIDTH sind Dezimalzahlen; die Werte für COMPRESS und CTRLZ sind TRUE oder FALSE.

Eine CONFIG.EDT Datei könnte z. B. wie folgt aussehen:

```
ASM TABS=8,8,8,8
ASM WIDTH=120
ASM COMPRESS=FALSE
ASM CTRLZ=FALSE
```

Alle für verschiedene Erweiterungen festgelegten Standardvorgaben müssen in einer CONFIG.EDT Datei enthalten sein. Enthält CONFIG.EDT mehr als eine Definition eines bestimmten Wertes für eine Erweiterung, benutzt der Editor den letzten.

1.5 EDITOR-BEFEHLE

Der Editor reagiert auf Befehle, für die keine Parameter notwendig sind, sobald die vorgeschriebene Taste gedrückt wird. Bei Wahl eines Befehls, der die Eingabe von Parametern erforderlich macht, erscheint oben am Bildschirm eine Befehlszeile mit Feldern für diese Werte. Die Felder lassen sich wie Textzeilen aufbereiten. Um den Cursor von Feld zu Feld zu rücken, Tab, Rücktab und Zeilenschaltung benutzen. Wenn die gewünschten Werte eingetragen sind, AUSF drücken. Zum Abbruch eines Befehls ANNULLIER drücken.

Die Sonderfunktionstaste 15 ist eine Rücktaste für Parameterfelder und Textzeilen. SF 15 nach Aufruf eines Befehls drücken, um ein Parameterfeld mit dem Wert aufzubereiten, den das Feld beim letzten Aufruf des Befehls in dieser Phase hatte. Wird z. B. der SUCH-Befehl aufgerufen und SF 15 gedrückt, erscheint im Suchfeld die zuletzt gesuchte Zeichenfolge. Nach Überschreiben von bestehendem Text SF 15 drücken, um den vorhergehenden Text wiederherzustellen.

Es folgt eine Beschreibung der einzelnen Editor-Befehle:

EINFÜGMODUS

Im Einfügmodus fügt der Editor die eingetasteten Zeichen an der Cursorposition in den Text ein. Ist der Editor nicht im Einfügmodus, ersetzt der eingetastete Text vorhandene Zeichen. Durch Einfügen verschiebt sich vorhandener Text nach rechts. Bei Druck der Rückschritt-Taste im Einfügmodus wird das Zeichen links vom Cursor gelöscht.

Während der Einfügmodus benutzt wird, erscheint der Name der Funktion erhellt in der Statuszeile oben am Bildschirm, und die Leuchtdiode links außen ist erhellt.

EINFÜGEN drücken, um den Einfügmodus zu wählen.
EINFÜGEN drücken, um den Einfügmodus wieder abzuwählen.

ERSETZEN

Dieser Befehl sucht nach einer Zeichenfolge und ersetzt diese mit einer anderen. Zuerst verlangt der Befehl nach einer Zeichensuch- und einer Zeichenersatzfolge. Dann sucht der Editor ab Spalte 1 der jeweiligen Cursorzeile nach der ersten Folge. Während des Suchvorgangs werden nur Zeichen erkannt, die wie in der eingetasteten Suchfolge entweder groß- oder kleingeschrieben sind. Wenn der Editor die Folge findet, erscheint der Dialoghinweis "Ersetz J oder N"; die Standardantwort ist "J". Um die Folge zu ersetzen, AUSF drücken. Soll diese Zeichenfolge jedoch nicht ersetzt werden, "N" eintragen und AUSF drücken. In beiden Fällen geht die Suche nach der nächsten Zeichenfolge weiter. ANNULLIER drücken, um den Befehl abzubrechen.

ANMERKUNG:
Falls die Zeichenersatzfolge länger als die Zeichensuchfolge ist, gehen u. U. Zeichen am Zeilenende verloren. Eine Mitteilung weist darauf hin, daß der Ersatz einen Zeichenverlust zur Folge haben wird.

Sollen alle übereinstimmenden Folgen ersetzt werden, ohne daß ein Dialoghinweis bei jeder einzelnen Folge erscheint, diesen Befehl mit UMSCH + ERSETZ aufrufen.

ERSETZ oder UMSCH + ERSETZ drücken.

GEHE NACH ERSTE ZEILE

Dieser Befehl verschiebt den Cursor zur ersten Zeile der Datei. Die Cursorspalte bleibt unverändert.

UMSCH + VORH drücken.

GEHE NACH LETZTE ZEILE

Dieser Befehl verschiebt den Cursor zur letzten Zeile der Datei. Die Cursorspalte bleibt unverändert.

UMSCH + NÄCHS drücken.

GEHE NACH ZEILE

Dieser Befehl erfordert Eintrag einer Zeilennummer und führt den Cursor an diese Zeile. Die Spalte bleibt unverändert.

GEHE NACH drücken.

MEHRERE ZEILEN LÖSCHEN

Dieser Befehl löscht einen Block von Textzeilen. Der Befehl fragt nach der Anfangs- und Endzeile. Der Standardwert für beide Felder ist die laufende Zeilennummer. Um die Zeilennummer in einem der Felder zu ändern, die Nord- bzw. Südcursor-Steuerungstaste (mit oder ohne UMSCH) oder die NÄCHS- bzw. VORH-Taste benutzen, um den Cursor an die gewünschte Zeile zu stellen. Die Zeilennummer im Feld wird automatisch zur Cursorzeilennummer. Mit der Tab-, Rücktab- und Zeilenschalttaste von Feld zu Feld rücken. Wenn die gewünschten Werte in den Hinweisfeldern erscheinen, AUSF drücken; die festgelegten Zeilen werden gelöscht.

STOP drücken.

NÄCHSTER BILDSCHIRM

Dieser Befehl verschiebt den ganzen nächsten Bildschirm ins Fenster. Die Cursorposition ändert sich nicht, es sei denn, der Cursor befindet sich außerhalb der letzten Textzeile am neuen Bildschirm. In diesem Fall ist der Cursor in der letzten Textzeile in derselben Spalte.

NÄCHS drücken.

NORDCURSOR und SÜDCURSOR

Der NORDCURSOR-Befehl versetzt den Cursor eine Zeile über die aktuelle Zeile; der SÜDCURSOR-Befehl verschiebt den Cursor in die Zeile darunter. Die Cursorspalte ändert sich nicht. Der Bildschirm rollt entsprechend der Zentrier- oder Gleitcursormodus-Einstellung ab (siehe ZENTRIERMODUS-Befehl).

Nord- bzw. Südcursor-Steuerungstaste drücken.

OSTCURSOR und WESTCURSOR

Der OSTCURSOR-Befehl versetzt den Cursor eine Position nach rechts; der WESTCURSOR-Befehl verschiebt den Cursor eine Position nach links. Die Cursorzeile ändert sich nicht. Ist die Zeile länger als 80 Zeichen und der Cursor steht am rechten oder linken Bildschirmrand, rollt der Bildschirm um eine Position nach rechts oder links ab. Der Abrollvorgang bricht ab, wenn der Cursor den Anfang oder das Ende der Zeile erreicht.

Ost- bzw. Westcursor-Steuerungstaste drücken.

PHASE BEENDEN UND ÄNDERUNGEN SPEICHERN

Dieser Befehl führt zum Programmentwicklungsmenü zurück und bietet die Möglichkeit, Änderungen aufzubewahren. Der Befehl zeigt den Hinweis PRESS EXECUTE AND THE TEXT WILL BE SAVED, TYPE N AND CHANGES WILL BE LOST an. Der Name der Datei, die aufbereitet wird, erscheint ebenfalls am Monitor, um anzuzeigen, welche Datei bei Druck von AUSF geschrieben wird. Wird N oder n eingetastet, gehen alle Änderungen verloren.

Wird der Dialoghinweis mit C oder c beantwortet, speichert der Editor den Text im verdichtetem Format. Wird AUSF gedrückt oder A bzw. a eingetastet, speichert der Editor den Text in unverdichtetem Format (Standardvorgabe). Das verdichtete und unverdichtete Format wird in Abschnitt 1.4 beschrieben.

Falls dieser Befehl versehentlich aufgerufen wird, kann er mit ANNULLIER aufgehoben werden.

UMSCH + ANNULLIER oder KTRL + C drücken.

REST LÖSCHEN

Dieser Befehl löscht die Zeichen von der derzeitigen Cursorposition bis zum Ende der Zeile.

LÖ REST drücken.

RÜCKSCHRITT

Der Cursor bewegt sich eine Spalte nach links, und eine Leerstelle ersetzt das Zeichen an der neuen Cursorposition.

Rückschritt-Taste drücken.

SUCHEN

Dieser Befehl fragt nach einer Zeichenfolge und beginnt die Suche nach dieser Folge auf der Zeile nach der aktuellen Cursorzeile. Die Zeile, in der eine Folge gefunden wird, wird zur neuen Cursorzeile und erscheint erhellt in der Mitte des Bildschirms. AUSF drücken, um die nächste Folge zu finden. Falls die Suche ergebnislos ist, erscheint eine diesbezügliche Mitteilung, und die Cursorposition bleibt unverändert. Soll die Zeichenfolge sowohl in Groß- als auch Kleinschreibung gesucht werden, den Befehl mit UMSCH + SUCH aufrufen.

SUCH oder UMSCH + SUCH drücken.

TAB

Dieser Befehl rückt den Cursor zum nächsten Tabstop. Falls sich rechts vom Cursor keine Tabstops befinden, bewegt sich der Cursor um eine Stelle nach rechts. Tabs treten im eigentlichen Text nicht auf und werden deshalb mit dem Suchbefehl nicht gefunden.

TAB drücken.

TABS SETZEN

Dieser Befehl ermöglicht die Festlegung und Änderung von Tabpositionen. Bei Aufruf dieses Befehls steht der Cursor in Spalte 0 in der Tabzeile. Den Cursor mit der Ost- bzw. Westcursor-Steuerungstaste an die gewünschte Tabposition verschieben. Einen Tab entweder mit der Tab-Taste oder T (groß- oder kleingeschrieben) setzen. Daraufhin erscheint ein Tabsymbol auf der Cursorposition in der Tabzeile. Um einen eingetragenen Tab zu löschen, das Tabsymbol mit der Leerschritt-Taste oder einem Bindestrich (-) überschreiben.

DEZI TAB drücken.

TEILDATEI LADEN

Dieser Befehl liest einen Zeilenbereich aus einer Datei in den Textpuffer und fügt sie hinter der jeweiligen Cursorposition in die bestehende Textdatei ein. Der Befehlsdialoghinweis fragt nach einer Dateispezifikation sowie nach der Anfangs- und Endzeile. Die Standardwerte für diese Felder sind jeweils 1 und 9999. Die den zu ladenden Zeilenbereich bestimmenden Werte eintasten, dann AUSF drücken.

UMSCH + EINRÜCK drücken.

TEILDATEI SPEICHERN

Dieser Befehl schreibt mehrere Zeilen vom Speicher auf Disk(ette). Der Befehlsdialoghinweis fragt nach einer Dateispezifikation sowie einer Anfangs- und Endzeile. Der Standardwert für beide Felder ist die laufende Zeilennummer. Um die Zeilennummer in beiden Feldern zu ändern, den Cursor mit der Nord- oder Südcursor-Steuerungstaste (mit oder ohne UMSCH) oder mit der VORH bzw. NÄCHS-Taste an die gewünschte Zeile stellen. Die Zeilennummer im Feld wird automatisch zur Cursorzeilennummer. Mit der Tab-, Rücktab- und Zeilenschalttaste von Feld zu Feld rücken. Bei Druck der AUSF-Taste werden die Zeilen in die angegebene Datei geschrieben.

ANMERKUNG:
Falls die bezeichnete Datei bereits besteht, löscht dieser Befehl die vorhergehende Fassung.

UMSCH + GEHE NACH drücken.

TEXT LADEN

Dieser Befehl lädt eine bestimmte Textdatei von einer Disk(ette) in den Speicher. Nach Erscheinen des Befehlsdialoghinweises die gewünschte Dateispezifikation eintragen und AUSF drücken, um die Datei laden. Falls in der laufenden Datei Änderungen vorgenommen wurden, fragt ein Dialoghinweis, ob dieser Textbereich gespeichert werden soll. Die Dateispezifikation des laufenden Textes erscheint ebenfalls. AUSF drücken, um die Änderungen zu speichern. Sollen die Änderungen nicht gespeichert werden, zuerst N und dann AUSF drücken.

Wird der Dialoghinweis mit C oder c beantwortet, speichert der Editor den Text in verdichtetem Format. AUSF drücken oder A bzw. a eintasten, um den Text in unverdichtetem Format (Standardwert) zu speichern. Verdichtete und unverdichtete Formate werden in Abschnitt 1.4 erläutert.

EINRÜCK drücken.

UMSCH + NÄCHS und UMSCH + VORH

Der Befehl UMSCH + NÄCHS versetzt den Cursor in die letzte Zeile der Datei, die aufbereitet wird. Der Befehl UMSCH + VORH versetzt den Cursor in die erste Zeile der Datei, die aufbereitet wird. Die Cursorspalte bleibt unverändert. Der Bildschirm rollt in dem gerade gültigen Modus ab, d. h. Zentrier- oder Gleitcursormodus (siehe ZENTRIERMODUS-Befehl).

UMSCH + NÄCHS oder UMSCH + VORH drücken.

UMSCH + OSTCURSOR und UMSCH + WESTCURSOR

Der Befehl UMSCH + OSTCURSOR versetzt den Cursor zum letzten Zeichen der Zeile. Der Befehl UMSCH + WESTCURSOR versetzt den Cursor in Spalte 1. Die Cursorzeile bleibt unverändert. Der Bildschirm rollt nach links oder rechts ab, falls die neue Cursorposition nicht im gerade angezeigten Bildschirm ist.

UMSCH + Ost- oder Westcursor-Steuerungstaste drücken.

VORHERGEHENDER BILDSCHIRM

Dieser Befehl verschiebt den ganzen vorhergehenden Bildschirm ins Fenster. Die Cursorposition ändert sich nicht, es sei denn, der Cursor befindet sich vor der ersten Textzeile am neuen Bildschirm. In diesem Fall ist der Cursor in der ersten Zeile in derselben Spalte.

VORH drücken.

ZEICHEN LÖSCHEN

Dieser Befehl löscht das Zeichen auf der Cursorposition. Die Zeichen auf der Zeile rechts vom Cursor werden um eine Position nach links verschoben. In die letzte Spalte wird eine Leerstelle gesetzt.

LÖSCH drücken.

ZEILE EINFÜGEN

Dieser Befehl fügt eine Leerzeile unmittelbar unter der Cursorzeile ein.

AUSF drücken.

ZEILE LEEREN

Dieser Befehl entfernt alle Zeichen auf der Cursorzeile, ersetzt sie mit Leerstellen und setzt den Cursor an den Anfang der Zeile.

UMSCH + LÖ REST.

ZEILE LÖSCHEN

Dieser Befehl löscht die Zeile an der aktuellen Cursorposition.

UMSCH + LÖSCH drücken.

ZEILEN DRUCKEN

Dieser Befehl schickt einen Textzeilenblock zum Drucker. Der Befehlsdialog-hinweis fragt nach der Anfangs- und Endzeile. Der Standardwert für beide Felder ist die laufende Zeilennummer. Um die Zeilennummer in einem der Felder zu ändern, den Cursor mit der Nord- oder Südcursor-Steuerungstaste (mit oder ohne UMSCH) bzw. der NÄCHS- oder VORH-Taste an die gewünschte Zeile setzen. Die Zeilennummer im Feld wird automatisch zur Cursorzeilennummer. Mit der Tab-, Rücktab- und Zeilenschalttaste von Feld zu Feld rücken. Wenn die gewünschten Werte in den Dialogfeldern erscheinen, AUSF drücken, um die Zeilen zu drucken.

DRUCK drücken.

ZEILEN KOPIEREN

Dieser Befehl kopiert einen Zeilenblock von einer bestimmten Textstelle an eine andere. Die Originalzeilen an der ersten Textstelle bleiben erhalten. Der Befehlsdialoghinweis fragt nach der Anfangs-, End- und Zielzeile. Die kopierten Zeilen werden nach der Zielzeile eingefügt. Der Standardwert für diese drei Felder ist die aktuelle Zeilennummer. Um die Zeilennummer in einem Feld zu ändern, die Nord- bzw. Süd-Cursor- Steuerungstaste (mit oder ohne UMSCH) oder die NÄCHS- bzw. VORH-Taste benutzen, um den Cursor an die gewünschte Zeile zu setzen. Die Zeilennummer im Feld wird automatisch zur Cursorzeilennummer. Mit der Tab-, Rücktab- und Zeilenschalttaste von Feld zu Feld rücken. Wenn die gewünschten Werte in den Hinweisfeldern erscheinen, AUSF drücken; der Kopiervorgang wird durchgeführt.

KOPIER drücken.

ZEILENLÄNGE FESTLEGEN

Dieser Befehl ermöglicht, die Zeilenlänge zu ändern. Bei Aufruf des Befehls muß die neue Zeilenlänge eingetragen werden. Eine Zahl zwischen 1 und 255 eingeben und AUSF drücken.

ANMERKUNG:
Mit diesem Befehl wird Text nicht umformatiert. Bei Verringerung der Zeilenlänge können bestehende Zeilen abgeschnitten werden. Der Text geht jedoch nicht verloren. Er kann durch Eingabe des vorhergehenden Zeilenlängenwertes wiederhergestellt werden.

FORMAT drücken.

ZEILEN VERSCHIEBEN

Dieser Befehl verschiebt einen Zeilenblock von einer bestimmten Textstelle an eine andere. Die Zeilen an der ursprünglichen Stelle werden gelöscht. Der Befehl fragt nach einer Anfangs-, End- und Zielzeile. Die verschobenen Zeilen werden nach der Zielzeile eingefügt. Der Standardwert für diese drei Felder ist die laufende Zeilennummer. Um die Zeilennummer in einem Feld zu ändern, den Cursor mit Hilfe der Nord- oder Südcursor-Steuerungstaste (mit oder ohne UMSCH) bzw. der NÄCHS- oder VORH-Taste an die gewünschte Zeile stellen. Die Zeilennummer im Feld wird automatisch zur Cursorzeilennummer. Mit der Tab-, Rücktab- und Zeilenschalttaste von Feld zu Feld rücken. Wenn die gewünschten Werte in den Dialogfeldern erscheinen, AUSF drücken, und die Zeilen werden verschoben.

VERSCHIE drücken.

ZENTRIERMODUS

Der Zentriermodus richtet die aufzubereitende Zeile in der vertikalen Bildschirmmitte aus. In diesem Modus rollt der Bildschirm weiter, wenn der Cursor an eine neue Zeile gesetzt wird. Im Zentriermodus ist die Zeile unmittelbar vor und nach der aufzubereitenden Zeile sichtbar.

Der Gleitcursormodus erlaubt Aufbereitung an einer beliebigen Stelle des Bildschirms. In diesem Modus rollt der Bildschirm bei dem Versuch weiter, den Cursor über die erste bzw. unter die letzte Anzeigezeile zu verschieben. Im Gleitcursormodus bleibt ein kleiner Teil des Textes zusammen mit den oberen Zeilen sichtbar, während z. B. die unteren Zeilen aufbereitetet werden. Der Gleitcursormodus ist auch bei Systemen mit einer Videosteckplatte mit niedrigem Auflösungswert von Vorteil, weil das Abrollen bei diesen Systemen langsamer verläuft.

Im Zentriermodus ist die Zeile in der Mitte des Bildschirms, im Gleitcursormodus ist der ganze Bildschirm erhellt. Es ist also am Bildschirm ersichtlich, welcher Modus benutzt wird.

ZENTRIER im Gleitcursormodus drücken, um den Zentriermodus aufzurufen.
ZENTRIER im Zentriermodus drücken, um den Gleitcursormodus aufzurufen.

KAPITEL 2
LINKER

2.1 EINFÜHRUNG

Das Linker-Utility-Programm (LINK.EXE) ist ein verschiebbarer Linker, der zur Verbindung von einzeln erstellten Objektcode-Moduln bestimmt ist. Bei den Objektcode-Moduln darf es sich nur um 8086 Dateien handeln.

Der Linker fordert Eingabe über Dialoghinweise an. Die auf Dialoghinweise gegebenen Antworten stellen Befehle an den Linker dar.

Der Linker arbeitet mit einer verzeichnis-indexierten Library-Suchmethode, welche die Verknüpfungszeit für Library-Suchen umfassende Phasen wesentlich verkürzen.

2.1.1 Überblick über den Linker-Vorgang

Der Linker verbindet mehrere einzeln kompilierte Objektmoduln in einen verschiebbaren Lademodul oder eine verschiebbare Programmdatei. Alle Adressen in der Programmdatei beziehen sich auf eins der vier 8086 Segment-Register (siehe Abschnitt A.1.2). Das Betriebssystem kann die Programmdatei an jeder verfügbaren 16-Byte-Grenze laden.

Bei Zuweisung der entsprechenden Adressen für die Objektmoduln löst der Linker die Bezugnahmen in modulexterne Symbole auf. Der Linker kann mehrfache Library-Dateien nach Definitionen aller externen, nicht aufgelösten Bezugnahmen absuchen. Ferner erstellt der Linker eine Listdatei mit gelösten externen Bezugnahmen und möglichen Fehlermeldungen.

Der Linker setzt die relativen Positionen der zu verknüpfenden Moduln fest, so daß das verknüpfte Programm so wenig Speicherraum wie möglich einnimmt. Ist der verfügbare Speicherraum erschöpft, erstellt der Linker eine vorläufige Diskdatei (VM.TMP) und operiert als virtueller Linker. Die Gesamtgröße der verknüpften Dateien darf jedoch 384KB nicht überschreiten.

Abbildung 2-1 illustriert die Ein- und Ausgabevorgänge des Linkers.

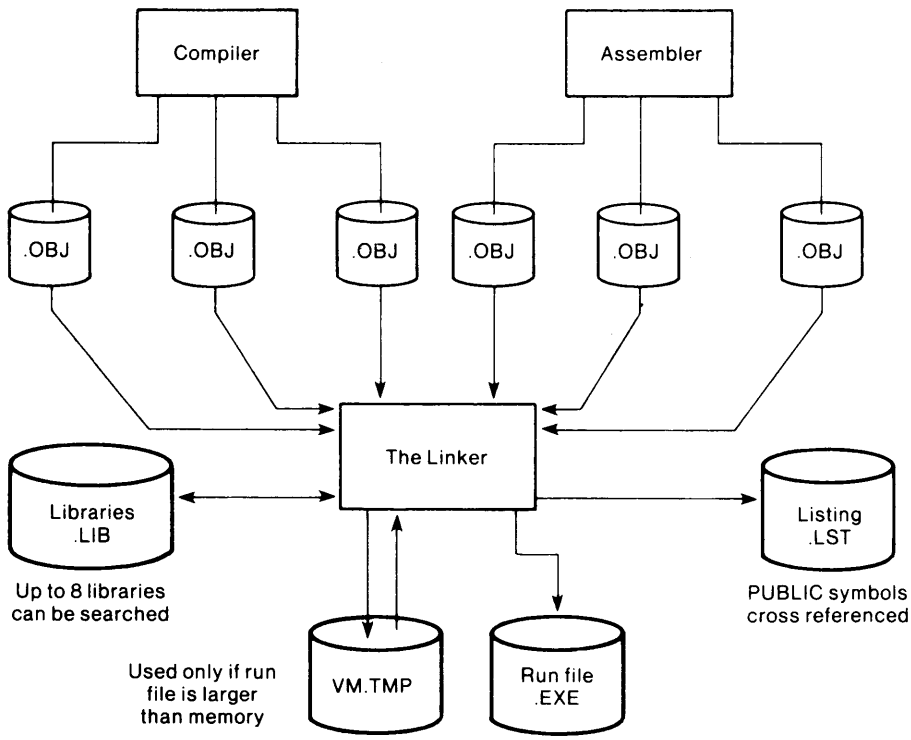


Abbildung 2-1. Linker-Eingabe und -Ausgabe

2.1.2 Linker-System-Erfordernisse

Die Linker-Utility benötigt einen Mindestspeicherraum von 50KB, und zwar 40KB für Code und Daten und 10KB für Programmlauf.

Einzellaufwerksysteme können den Linker nur dann benutzen, wenn die Ausgabe des Linkers zu derselben physischen Disk geschickt wird, von der auch die Eingabe kommt. Bei einer Einzellaufwerk-Konfiguration ist es während des Linker-Vorgangs nicht möglich, Disks zu wechseln. Ein Doppellaufwerksystem ist daher eine praktischere Konfiguration. Wenn sich die Eingabedatei jedoch nicht auf der aktuellen Disk befindet, fordert der Linker die Einlage einer neuen Diskette über einen Dialoghinweis an, bevor die Verarbeitung fortgesetzt wird.

2.2 SEGMENTE, GRUPPEN UND KLASSEN

Dieser Abschnitt beschreibt die Grundlagen für die Linker-Funktion. Segmente, Gruppen und Klassen, d. h. die Einheiten, die der Linker steuert, werden erläutert. Eine Erklärung, wie der Linker Segmente verbindet und anordnet, ist ebenfalls enthalten.

2.2.1 Definitionen

Ein Segment ist ein Programmm-Modul oder eine Reihe von Moduln mit dem gleichen Segmentnamen. Ein Segment kann ein ganzes oder ein Teilprogramm darstellen und ist die kleinste Einheit, die der Linker manipuliert.

In Assembler werden Moduln einem Segment zugewiesen, indem sie den gleichen Segmentnamen erhalten. In kompilierten Sprachen weist der Compiler den Segmentnamen zu.

Der Inhalt eines Segments belegt einen zusammenhängenden Speicherbereich bis zu 64KB. Der Segmentinhalt wird durch ein einzelnes Segmentregister plus Offset adressiert. Ein Segment kann an einer beliebigen 16-Byte-Grenze im 8086 Speicher beginnen.

Eine Gruppe ist eine Reihe von Segmenten, die verschiedene Segmentnamen tragen, jedoch vom gleichen Segmentregister adressiert werden. Die Segmente innerhalb einer Gruppe können nichtzusammenhängende Speicherbereiche belegen.

In Assembler werden Segmente einer Gruppe zugewiesen, indem sie den gleichen Gruppennamen erhalten. In kompilierten Sprachen weist der Compiler den Gruppennamen zu.

Da alle Segmente in der Gruppe durch ein einzelnes Segmentregister plus Offset adressiert werden, umfaßt die Gruppe maximal 64KB. Der Linker stellt sicher, daß die Objektmoduln einer Gruppe diese 64K-Byte-Grenze einhalten.

Eine Klasse besteht, ähnlich einer Gruppe, aus einer Reihe von Segmenten mit verschiedenen Segmentnamen. Im Gegensatz zu einer Gruppe belegen die Segmente einer Klasse einen zusammenhängenden Speicherbereich. Die Stellung der Segmente in der gleichen Klasse beeinflußt jedoch nicht ihre Adressierung, d. h. Segmente der gleichen Klasse werden nicht automatisch von einem Einzelsegmentregister adressiert.

Segmente werden zur Kompilier- oder Assemblierungszeit einer Klasse zugewiesen. In Assembler erhalten die Segmente den gleichen Klassenamen. In kompilierten Sprachen weist der Compiler den Klassenamen zu.

Durch Zuweisung der Segmente zu einer Klasse wird die Reihenfolge und die relative Stellung der Segmente im Speicher bestimmt. Eine Klasse geht einer anderen im Speicher voraus, wenn ein Segment der ersten Klasse allen Segmenten der zweiten Klasse in der Eingabe zum Linker vorausgeht. Sobald daher der Linker ein Segment von einer Klasse als Eingabe annimmt, werden alle anderen Segmente derselben Klasse vor Segmenten anderer Klassen zusammenhängend gespeichert. Der Linker lädt die Segmente einer Klasse in der Reihenfolge, in der sie in den Objektdateien auftreten. Klassen können über die 64K-Byte-Grenze hinaus geladen werden.

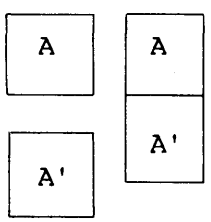
Angenommen, Klasse CODE enthält Segmente A und D und Klasse DATA Segmente B und C. Wenn das erste Segment, auf das der Linker stößt, A ist, stellt der Linker die der Klasse CODE angehörenden Segmente in einen zusammenhängenden Bereich im Speicher vor die Segmente der Klasse DATA. Gehören jedoch Segmente A und C zur Gruppe G gehören, haben ihre Adressen das gleiche Segmentregister, obwohl sie einen nicht zusammenhängenden Speicherbereich belegen.

2.2.2 Verbindung und Anordnung von Segmenten

Die Linker arbeitet mit vier "Verbindungsarten", die im Quellmodul für den Assembler oder Compiler angegeben werden: Privat, Stapel, Publik und Gemeinsam. (Die in MAKRO-86 von Microsoft verfügbare Speicherverbindungsart wird wie "Publik" behandelt. Der Linker stellt nicht automatisch Segmente der Speicherverbindungsart an die höchsten Adressen.)

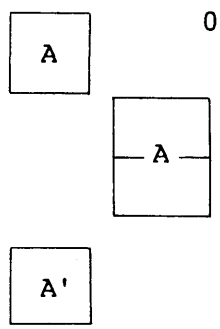
Der Linker verbindet Segmente folgendermaßen:

Privat



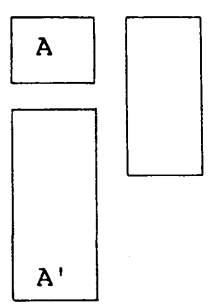
Privat-Segmente werden getrennt geladen und bleiben getrennt. Sie können physisch, aber nicht logisch zusammenhängen, auch wenn die Segmente den gleichen Namen tragen. Jedes Privat-Segment hat seine eigene Basisadresse.

Publik



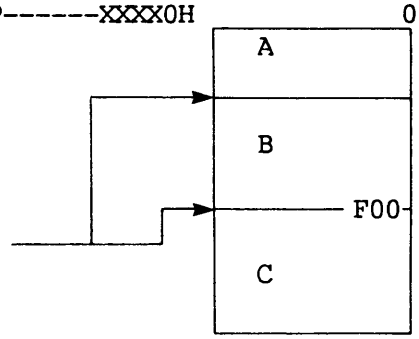
Publik-Segmente mit gleichem Namen und Klassennamen werden zusammenhängend geladen. Der Offset beginnt beim ersten und reicht bis zum letzten geladenen Segment. Es gibt nur eine Basisadresse für alle Publik-Segmente mit gleichem Namen und Klassennamen. Die Stapel- und Speicherverbindungsart wird wie Publik behandelt. Der Stapelanzeiger weist jedoch auf die erste Adresse des ersten Stapelspeichersegments. (Publik-Segmente mit gleichem Segment-, jedoch verschiedenen Klassennamen können zusammenhängend oder nicht zusammenhängend geladen werden.)

Gemeinsam



Gemeinsame Segmente mit gleichem Namen und Klassennamen werden mit Überlagerung geladen, d. h. eine Basisadresse gilt für alle gemeinsamen Segmente mit gleichem Namen. Nur ein Segment befindet sich jeweils im Speicher. Wenn das System ein anderes Segment an der gleichen Adresse lädt, wird das vorhergehende Segment überschrieben. Die Länge des gemeinsamen Bereichs ist die Länge des längsten Segments. (Die Anfangswerte der Variablen in einem gemeinsamen Segment können unbestimmt sein.)

Werden Segmente gruppenweise in den Assembler gestellt, wird Offset-Adressierung von Elementen von einer einzigen Basisadresse für alle Segmente in dieser Gruppe ermöglicht.



Eine beliebige Anzahl anderer Segmente kann zwischen die Segmente einer Gruppe treten. Daher kann der Offset von F00 größer als die Gesamtgröße der Segmente in der Gruppe, jedoch nicht größer als 64KB sein.

Operand DGROUP:F00 gibt den Offset von F00 vom Anfang des ersten Segments von DGROUP (hier Segment A) an.

Der Linker lädt Segmente nach deklarierten Klassennamen, und zwar alle Segmente, die zum zuerst auftretenden Klassennamen gehören, dann alle Segmente des danach auftretenden Klassennamens usw., bis alle Klassen geladen sind.

Enthält ein Assembler-Programm:	wird es wie folgt geladen:
A SEGMENT 'F00'	'F00
B SEGMENT 'BAZ'	A
C SEGMENT 'BAZ'	E
D SEGMENT 'Z00'	'BAZ'
E SEGMENT 'F00'	B
	C
	'Z00'
	D

Wird ein Assembler-Programm geschrieben, kann die Reihenfolge der Klassen im Speicher gesteuert werden, indem ein Pseudo-Modul geschrieben und dieses zuerst als Antwort auf den Objektmodul-Dialoghinweis des Linkers gelistet wird. Der Pseudo-Modul teilt die Segmente in Klassen in der Reihenfolge ein, in der sie geladen werden sollen. Zum Beispiel:

```
A SEGMENT   'CODE'
A ENDS
B SEGMENT   'CONST'
B ENDS
C SEGMENT   'DATA'
C ENDS
D SEGMENT   STACK   'STACK'
D ENDS
E SEGMENT   'MEMORY'
E ENDS
```

Darauf achten, daß alle im Programm benutzten Klassen in diesem Modul angegeben werden, da anderenfalls keine Kontrolle über die Reihenfolge der Klassen besteht.

Sollen Segmente mit der Speicherverbindungsart als letztem Programmsegment geladen werden, einfach MEMORY zwischen SEGMENT und 'MEMORY' in der E-Segment-Zeile oben einfügen. Der Linker lädt diese Segmente jedoch nur zuletzt, weil diese Kontrolle ausgeübt wurde und nicht aufgrund von eigenen Fähigkeiten im Linker- oder Assembler-Prozeß.

VORSICHT:
Diese Methode nicht bei BASIC-, COBOL-, FORTRAN- oder Pascal-Programmen anwenden. Compiler und Linker hier die normalen Vorgänge ausführen lassen.

2.3 VOM LINKER BENUTZTE DATEIEN

Der Linker arbeitet mit einer oder mehreren Eingabedateien, erstellt zwei Ausgabedateien, kann eine virtuelle Speicherdatei erstellen und angewiesen werden, eine bis acht Library-Dateien abzusuchen. Für jede Dateiart kann eine 3-teilige Dateispezifikation angegeben werden. Das Format für die Linker-Dateispezifikation lautet wie folgt:

[<d:>]<Filename>[.<Erweiterung>]

<d:>	Die Laufwerkbezeichnung. Gültige Laufwerkbezeichnungen für den Linker sind A: bis O:. Der Doppelpunkt ist immer Teil der Laufwerkbezeichnung. Wird keine Laufwerkbezeichnung eingetragen, gilt das Standardlaufwerk.
<Filename>	Jeder zulässige Dateiname von 1 bis 8 Zeichen.
.<Erweiterung>	Eine Dateinamenerweiterung von 1 bis 3 Zeichen. Der Punkt ist immer Teil der Erweiterung

2.3.1 Eingabedateien

Werden keine Erweiterungen in den Eingabe- (Objekt- und Library-) Dateispezifikationen angegeben, erkennt der Linker folgendes als Standardwert an: .OBJ für Objektdateien und .LIB für Librarydateien.

2.3.2 Ausgabedateien

Der Linker hängt folgende Standarderweiterungen an die Ausgabe- (Programm- und List-) Dateien an: .EXE bei Programmdateien und .MAP bei Listdateien. Der .EXE Standardwert kann nicht aufgehoben werden.

2.3.3 VM.TMP Datei

Der Linker benutzt den verfügbaren Speicher für die Linkphase. Falls die verknüpften Dateien eine Ausgabedatei erstellen, die den verfügbaren Speicherraum überschreitet, erstellt der Linker eine Arbeitsdatei und nennt sie VM.TMP. Ist die Erstellung von VM.TMP erforderlich, zeigt der Linker folgende Mitteilung an:

```
VM.TMP has been created.
Do not change diskette in drive, <d:>
```

Sobald diese Mitteilung erscheint, darf die Disk nicht vor Beendigung der Linkphase vom Standardlaufwerk entfernt werden. Wird die Disk vorher entfernt, ist der Linkvorgang unbestimmt, und der Linker zeigt u. U. folgende Fehlermeldung an:

```
Unexpected end of file on VM.TMP
```

Der Linker benutzt VM.TMP als virtuellen Speicher. Der Inhalt von VM.TMP wird anschließend in die Datei geschrieben, die auf den Programmdatei-Dialoghinweis hin bestimmt wird (siehe Abschnitt 2.5.2). VM.TMP ist nur eine vorläufige Datei und wird am Ende der Linkphase gelöscht.

VORSICHT:
VM.TMP nicht als Namen für andere Dateien benutzen. Falls sich eine Datei mit dem Namen VM.TMP im Standardlaufwerk befindet und vom Linker benötigt wird, wird sie von der Disk gelöscht und eine neue VM.TMP Datei erstellt. Der Inhalt der vorhergehenden VM.TMP Datei geht somit verloren.

2.4 LINKER-ABLAUF

Zum Linker-Ablauf sind zwei Befehlsarten erforderlich: ein Befehl zum Aufruf des Linkers und Antworten auf Befehlsdialoghinweise. Ferner steuern sechs Schalter die optionalen Linker-Funktionen. Im allgemeinen werden alle Linker-Befehle über die Tastatur eingegeben; es kann jedoch auch eine Datei erstellt werden, die Antworten auf Befehlsdialoghinweise und Schalter enthält. Der Linker unterstützt einige Sonderbefehlszeichen, die die Eingabe von Befehlen erleichtern.

Es gibt vier Arten zum Aufruf des Linkers. Bei Wahl der ersten zwei Methoden werden die übrigen Befehle als Antworten auf Dialoghinweise eingegeben. Bei der dritten Methode werden alle Befehle auf der Zeile eingetragen, die zum Aufruf des Linkers benutzt wird. Bei der vierten Methode wird eine alle notwendigen Befehle enthaltende Antwortdatei verwendet.

2.4.1 Methode 1

Zur Benutzung von Methode 1 die Programmentwicklungsoption vom Hauptsystem-Menü wählen. Die Linker-Option vom Programmentwicklungsmenü (Abbildung 1-1) wählen. Der Linker zeigt daraufhin nacheinander vier Dialoghinweise an. Die Antworten auf die Dialoghinweise bilden Befehle zur Durchführung bestimmter Linker-Aufgaben.

2.4.2 Methode 2

Zur Benutzung von Methode 2 den DOS Befehlsprozessor oder die Option "Andere" vom Hauptsystem-Menü wählen. Der DOS Dialoghinweis oder ein Dateispezifikations-Dialoghinweis erscheint. Den Dialoghinweis folgendermaßen beantworten:

```
LINK
```

Der Linker wird daraufhin in den Speicher geladen und zeigt nacheinander vier Dialoghinweise an.

2.4.3 Methode 3

Zur Benutzung von Methode 3 die DOS Befehlsprozessoroption vom Hauptsystem-Menü wählen. Der DOS Dialoghinweis oder ein Dateispezifikations-Dialoghinweis erscheint. Den Dialoghinweis wie folgt beantworten:

```
LINK
<Objekt-List>,<Programmdatei>,<Listdatei>,<Lib-List>[/Scha
lter...]
```

Die Eintragungen, die LINK folgen, sind Antworten auf die in Abschnitt 2.5 beschriebenen Befehlsdialoghinweise.

<Objekt-List> ist eine Aufstellung der Objektmoduln, die durch Pluszeichen getrennt sind.

<Programmdatei> ist der Name der ausführbaren Ausgabedatei.

<Listdatei> ist der Name der Datei, die die Auflistung enthalten soll.

<Lib-List> ist eine Aufstellung der Library-Moduln, die abgesucht werden sollen.

</Schalter...> sind wahlweise Schalter, die nach Eintragung einer beliebigen Antwort eingegeben werden können (vor Kommas oder, wie dargestellt, nach <Lib-List>).

Die Eingabefelder für Dialoghinweise müssen durch Kommas getrennt sein. Um den Standardwert für ein Feld zu wählen, einfach ein zweites Komma ohne Leerstellen eingeben. Zum Beispiel:

```
LINK FUN+TEXT+TABLE+CARE/P/M,,FUNLIST,COBLIB.LIB
```

In diesem Beispiel wird der Linker, dann die Objektmoduln FUN.OBJ, TEXT.OBJ, TABLE.OBJ und CARE.OBJ geladen. Der Linker hält dann (aufgrund des /P-Schalters) an. Wird eine beliebige Taste gedrückt, verknüpft der Linker die Objektmoduln, erstellt einen globalen Symbolbelegungsplan (aufgrund des /M-Schalters), die Standard-Programmdatei FUN.EXE, eine Listdatei mit dem Namen FUNLIST.MAP und sucht die Librarydatei COBLIB.LIB ab.

2.4.4 Methode 4

Zur Benutzung von Methode 4 den DOS Befehlsprozessor oder die Option "Andere" vom Hauptsystem-Menü wählen. Der DOS Dialoghinweis, der folgendermaßen beantwortet wird, erscheint:

```
LINK @<Filespez.>
```

<Filespez.> ist der Name einer Antwortdatei. Eine Antwortdatei enthält Antworten auf Linker-Dialoghinweise und u. U. auch Schalter. Methode 4 ermög- licht die Durchführung der Linkphase ohne interaktive (direkte) Beantwortung der Linker-Dialoghinweise.

Bevor Methode 4 zum Aufruf des Linkers benutzt werden kann, muß zunächst die Antwortdatei erstellt werden. Eine Antwortdatei besteht aus Textzeilen, und zwar eine für jeden Dialoghinweis. Antworten müssen in der gleichen Reihenfolge wie Befehlsdialoghinweise erscheinen. Schalter und Befehlszeichen müssen in der Antwortdatei auf die gleiche Weise benutzt werden wie bei Eintragung von Antworten auf der Tastatur.

Zu Beginn der Linkphase erscheint jeder Dialoghinweis mit den Antworten aus der Antwortdatei. Falls die Antwortdatei nicht auf alle Dialoghinweise Antworten (in Form von Dateinamen, Strichpunkt-Sonderzeichen oder Zeilenschaltungen) enthält, wartet der Linker in diesem Fall auf Eintragung einer gültigen Antwort. Nach Eintragung einer gültigen Antwort setzt der Linker die Linkphase fort.

Beispiel einer Linker-Antwortdatei:

```
FUN TEXT TABLE CARE <RETURN>
/PAUSE/MAP <RETURN>
FUNLIST <RETURN>
COBLIB.LIB <RETURN>
```

Diese Antwortdatei veranlaßt den Linker, die vier Objektmoduln zu laden. Vor Erstellung eines Publik-Symbol-Belegungsplans hält der Linker an, um den Diskwechsel zu ermöglichen (vor Benutzung dieses Schalters siehe Abschnitt 2.7.5, /PAUSE). Bei Druck einer beliebigen Taste erstellt der Linker Ausgabedateien mit dem Namen FUN.EXE und FUNLIST.MAP, sucht die Library-Datei nach COBLIB.LIB ab und benutzt Standardeinstellungen für die Kennzeichen.

2.5 BEFEHLSDIALOGHINWEISE

Wie bereits beschrieben, erhält der Linker Befehle durch Beantwortung von vier Textdialoghinweisen. Nach Beantwortung eines Dialoghinweises erscheint der nächste. Der Linker beginnt nach Beantwortung des letzten Dialoghinweises automatisch mit dem Linkvorgang. Nach Abschluß der Linkphase kehrt der Linker an die Stelle zurück, an der er aufgerufen wurde. Erscheint der Betriebssystemdialoghinweis, ist der Linkvorgang erfolgreich abgeschlossen; anderenfalls zeigt der Linker eine Fehlermeldung an.

Der Linker fragt über Dialoghinweise nach dem Namen von Objekt-, Programm- und Listdateien und Libraries. Die Dialoghinweise werden in der Reihenfolge gelistet, in der sie erscheinen. Bei Dialoghinweisen, die Standardantworten annehmen können, wird diese dem Dialoghinweis in eckigen Klammern ([ ]) nachgestellt.

2.5.1 Objektmoduln [.OBJ]: Dialoghinweis

Der Linker zeigt folgenden Dialoghinweis an:

```
Objekt Moduln [.OBJ]:
```

Die Liste der Objektmoduln eingeben, die verknüpft werden sollen. Die Moduln müssen durch eine Leerstelle oder ein Pluszeichen getrennt sein. Wird als letztes Zeichen ein Pluszeichen eingetragen, erscheint der Dialoghinweis erneut. Dieser Dialoghinweis hat keinen Standard-Dateinamen. Der Name einer Objektdatei ist einzutragen.

Der Standardwert für die Dateinamenerweiterung ist .OBJ. Trägt ein Objektmodul eine andere Dateinamenerweiterung, muß diese eingegeben werden. Anderenfalls kann die Erweiterung wegfallen. Diesem Dialoghinweis folgt nur deshalb eine Standarderweiterung, weil er keinen Standarddateinamen hat und ein Dateiname einzutragen ist.

Bei Bestimmung der Reihenfolge, in der die Objektmoduln eingetragen werden, ist daran zu denken, daß die Segmente in Klassen auch in dieser eingegebenen Reihenfolge geladen werden.

2.5.2 Programmdatei [Erster-Objekt-Dateiname.EXE]: Dialoghinweis

Der Linker zeigt folgenden Dialoghinweis an:

```
Programmdatei [<Objekt-Datei>.EXE]:
```

Den Dateinamen eintragen, damit die Datei den ausführbaren Objektcode erhält. Wird dieser Dialoghinweis nicht beantwortet, benutzt der Linker den ersten Dateinamen, der als Antwort auf den Objektmodul-Dialoghinweis eingegeben wurde, als Programmdateinamen mit der Erweiterung .EXE.

Der Linker benutzt den hier eingetragenen Dateinamen zum Speichern der Programm- (ausführbaren) Datei, die während der Linkphase entstanden ist. Alle Programmmdateien erhalten die Dateinamenerweiterung .EXE. Der Linker ignoriert andere Erweiterungen. Zum Beispiel:

```
Programmdatei [FUN.EXE]:  B:PAYROLL.COM/P
```

Diese Antwort weist den Linker an, die Programmdatei PAYROLL.EXE in Laufwerk B: zu erstellen. Der Linker hält ferner an (/P), damit eine neue Disk für die Programmdatei eingelegt werden kann.

2.5.3 Listdatei [Programmdateiname.MAP]: Dialoghinweis

Der Linker zeigt folgenden Dialoghinweis an:

```
Listdatei [<Programmdatei>.MAP]:
```

Den Dateinamen für die Listdatei eintragen. Wird dieser Dialoghinweis nicht beantwortet, benutzt der Linker den Programmdateinamen mit der Standarderweiterung .MAP.

Die Listdatei enthält einen Eintrag für jedes Segment in den Eingabe- (Objekt- und Library-) Moduln. Jeder Eintrag zeigt auch den Offset (Adressierung) in der Programmdatei.

2.5.4 Libraries [ ]: Dialoghinweis

Der Linker zeigt folgenden Dialoghinweis an:

```
Libraries [ ]:
```

Einen bis acht Namen abzusuchender Library-Dateien eintragen und mit Leerstellen oder Pluszeichen (+) voneinander trennen. Wird ein Pluszeichen als letztes Zeichen eingetragen, erscheint der Dialoghinweis erneut. Es kann auch mit Druck der Zeilenschaltung geantwortet werden. Zeilenschaltung allein bedeutet keine Library-Suche.

Library-Dateien werden vom Library-Verwalter erstellt (siehe Kapitel 3, Library-Verwalter). Die Standard-Dateinamenerweiterung für Library-Dateien ist .LIB.

Der Linker sucht die Library-Dateien in der aufgeführten Reihenfolge ab, um externe Bezugnahmen zu lösen. Wird der Modul gefunden, der das externe Symbol definiert, verarbeitet der Linker den Modul als weiteren Objektmodul. Bei mehrfach-definierten Symbolen bricht die Suche nach dem ersten auftretenden Fall ab.

Kann der Linker keine Library-Datei auf der Disk im Plattenlaufwerk finden, erscheint folgender Dialoghinweis:

```
Cannot find library <Library-Name>
Enter new drive letter:
```

Den Buchstaben der Laufwerkbezeichnung eintragen.

Der Linker sucht nicht sequentiell innerhalb jeder Library-Datei, sondern benutzt die sogenannte verzeichnis-indexierte Library-Suchmethode. Für jede Library erstellt der Library-Verwalter eine Tabelle, in der die definierten Publik-Symbole der Moduln innerhalb der Library indexiert sind. Folglich sucht der Linker Definitionen externer Bezugnahmen nicht von Dateianfang bis -ende, sondern durch Indexzugriff. Diese indexierte Suche schränkt die Linkzeit bei Phasen, die Library-Suchen beinhalten, wesentlich ein.

2.6 BEFEHLSZEICHEN

Linker-Befehle können mit drei Befehlszeichen eingegeben werden:

Das Pluszeichen (+) benutzen, um Eintragungen voneinander zu trennen und die laufende logische Zeile, die dem Objektmodul- und Library-Dialoghinweis folgt, zu erweitern. (Zum Trennen von Objektmoduln können auch Leerstellen verwendet werden.)

Um eine größere Anzahl u. U. längerer Antworten einzutragen, Pluszeichen/Zeilenschaltung am Ende der physischen Zeile eingeben (die logische Zeile wird auf diese Weise erweitert). Falls Pluszeichen/Zeilenschaltung die letzte Eintragung nach diesen Dialoghinweisen ist, fragt der Linker nach weiteren Modulnamen. Erscheint der Objektmodul- oder Library-Dialoghinweis erneut, weitere Antworten eintragen. Sind alle zu verknüpfenden Moduln gelistet, sicherstellen, daß die Antwortzeile mit einem Modulnamen und Zeilenschaltung und nicht mit Pluszeichen/Zeilenschaltung endet.

Beispiel:

```
Objektmoduln [.OBJ]:  FUN TEXT TABLE ARE+<RETURN>
Objektmoduln [.OBJ]:  F00+FLIPFLOP+JUNQUE+<RETURN>
Objektmoduln [.OBJ]:  CORSAIR<RETURN>
```

Einen einzelnen Strichpunkt (;) mit direkt nachfolgender Zeilenschaltung zu einem beliebigen Zeitpunkt nach dem ersten Dialoghinweis (ab "Programmdatei") benutzen, um Standardantworten auf die übrigen Dialoghinweise zu wählen. Diese Eigenschaft ist zeitsparend und macht die Eintragung mehrfacher Zeilenschaltungen überflüssig.

VORSICHT:
Sobald ein Strichpunkt eingetragen wird, kann in der jeweiligen Linkphase kein weiterer Dialoghinweis beantwortet werden. Den Strichpunkt daher nicht zum Übergehen von Dialoghinweisen benutzen. Um Dialoghinweise zu überspringen, Zeilenschaltung drücken.

Zum Beispiel:

```
Objektmoduln [.OBJ]:  FUN TEXT TABLE CARE<RETURN>
Programmdatei [FUN.EXE]:   ;<RETURN>
```

Die übrigen Dialoghinweise erscheinen nicht, und der Linker benutzt die Standardwerte (einschließlich FUN.MAP für die Listdatei).

UMSCH + ANNULLIER jederzeit zum Abbruch der Linkphase benutzen. Wird eine unrichtige Antwort, z. B. ein falscher oder nicht richtig buchstabierter Dateiname, eintragen, UMSCH + ANNULLIER drücken, um den Linker zu verlassen. Dann den Linker wieder aufrufen und die Prozedur wiederholen. Wird der Fehler vor tatsächlicher Eingabe in das System bemerkt, können die Zeichen auf dieser Zeile neugeschrieben werden.

2.7 SCHALTER

Am Ende jeder Dialoghinweiszeile können Schalter eingetragen werden, die wahlweise Linker-Funktionen steuern. Schalter können am Ende einer der Dialoghinweis-Antworten gruppiert oder am Ende verschiedener Antworten verstreut sein. Jedem Schalter muß ein Schrägstrich vorausgehen. Ist ein bestimmter Schalter nicht eingeschlossen, führt der Linker die für diesen Schalter bestimmte Funktion nicht aus.

Für alle Schalter kann eine aus Einzelbuchstaben bestehende Abkürzung oder der volle Schaltername verwendet werden. Die einzige Beschränkung ist, daß eine Abkürzung eine sequentielle Teilkette vom ersten bis zum letzten eingetragenen Buchstaben darstellen muß; Zwischenräume oder Transpositionen sind nicht erlaubt. Zum Beispiel:

Gültig	Ungültig
/D	/DSL
/DS	/DAL
/DSA	/DLC
/DSALLOCA	/DSALLOCT

2.7.1 /DSALLOCATE

Der Schalter /DSALLOCATE weist den Linker an, alle Daten im oberen Teil des Datensegments (DS) zu laden. Ohne diesen Schalter lädt der Linker alle Daten im unteren Teil des Datensegments. Zur Laufzeit enthält der DS-Anzeiger die unterste Adresse und läßt Gebrauch des gesamten DS-Segments zu. Die Compiler weisen die Datenteile der Programme einer sog. DGROUP zu. Der /DSALLOCATE-Schalter lokalisiert diese Gruppe im Speicher.

Das Vorhandensein des /DSALLOCATE-Schalters und das Fehlen des /HIGH-Schalters (siehe Abschnitt 2.7.2) erlauben dem Programm, verfügbaren Speicherraum unterhalb des speziell zugewiesenen Bereichs innerhalb DGroup dynamisch zuzuordnen, jedoch diesen Speicher mit demselben DS-Anzeiger zu adressieren. Das Anwendungsprogramm kann dynamisch bis zu 64KB (oder die tatsächlich verfügbare Menge) zuordnen, abzüglich der innerhalb DGROUP zugeordneten Menge. Pascal- und FORTRAN-Programme erfordern diese dynamische Zuweisung und müssen daher über diesen Schalter verfügen.

2.7.2 /HIGH

Der /HIGH-Schalter veranlaßt den Linker, die Programmdatei an die höchstmögliche Stelle im Speicher zu setzen. Ohne diesen Schalter setzt der Linker die Programmdatei so niedrig wie möglich.

VORSICHT:
Den /HIGH-Schalter nicht bei Pascal- oder FORTRAN-Programmen benutzen.

2.7.3 /LINENUMBERS

Der /LINENUMBERS-Schalter weist den Linker an, in der Listdatei Zeilennummern und Adressen der Quellanweisungen in den Eingabemoduln zu erfassen. Ohne diesen Schalter schließt der Linker die Zeilennummern nicht in der Listdatei ein.

Nicht alle Compiler erstellen Objektmoduln, die Zeilennummer-Information enthalten. In solchen Fällen kann der Linker keine Zeilennummern einschließen.

2.7.4 /MAP

/MAP weist den Linker an, alle in den Eingabemoduln definierten Publik- (Global-) Symbole einschließlich Definitionen zu listen. Wird /MAP nicht eingegeben, listet der Linker nur Fehler (einschließlich nicht definierter Global-Symbole).

Der Linker listet Symbole alphabetisch. Für jedes Symbol trägt der Linker den Wert und die Segment:Offset-Stelle in der Programmdatei ein. Die Symbole erscheinen am Ende der Listdatei.

2.7.5 /PAUSE

Der /PAUSE-Schalter veranlaßt den Linker, während der Linkphase anzuhalten und zu warten, bis die Zeilenschalttaste gedrückt wird. Im allgemeinen führt der Linker die Linkphase ohne Unterbrechung von Anfang bis Ende durch. Dieser Schalter ermöglicht Diskwechsel, bevor der Linker eine Ausgabedatei erstellt.

Trifft der Linker auf den /PAUSE-Schalter, erscheint folgende Mitteilung:

```
About to generate .EXE file
Change disks <hit any key>
```

Bei Druck einer beliebigen Taste nimmt der Linker die Verarbeitung wieder auf.

VORSICHT:
Weder die für Listdatei bestimmte noch die für VM.TMP benutzte Disk (falls erstellt) wechseln.

2.7.6 /STACK:<Zahl>

/STACK veranlaßt den Linker, der Programmdatei einen Stapelspeicher angegebener Größe zuzuweisen. <Zahl> stellt einen beliebigen positiven numerischen Wert (hexadezimaler Radix) bis zu 65536 Bytes dar. Wird ein Wert von 1 bis 511 eingetragen, benutzt der Linker 512.

Wird der /STACK-Schalter für eine Linkphase nicht benutzt, berechnet der Linker die erforderliche Stapelspeichergröße automatisch. Compiler und Assembler enthalten in den Objektmoduln Angaben, anhand welcher der Linker die benötigte Stapelspeichergröße berechnen kann.

Mindestens ein Objekt- (Eingabe-) Modul muß eine Stapelzuordnungs-Anweisung enthalten. Anderenfalls zeigt der Linker folgende Fehlermeldung an:

```
WARNING:  NO STACK STATEMENT
```

KAPITEL 3
LIBRARY-VERWALTER

3.1 EINFÜHRUNG

Der Library-Verwalter (LIB.EXE) ist ein Utility-Programm, das vom Linker benutzbare Library-Dateien erstellt, modifiziert und löscht. Dieses Kapitel beschreibt die Eigenschaften und die Arbeitsweise des Library-Verwalters.

3.1.1 Eigenschaften und Vorteile

Der Library-Verwalter erstellt Dateien, die Programm-Moduln enthalten, welche die Linker-Utility mit anderen Moduln verbinden kann (siehe Kapitel 2, Linker). Sobald der Library-Verwalter eine Library-Datei erstellt, kann er Moduln hinzufügen, löschen oder separate Objektdateien von Library-Dateimoduln erstellen.

Der Unterschied zwischen einer Objektdatei und einem Modul (oder Objektmodul) ist, daß ein Modul Teil einer Datei ist. Während eine Datei eine Laufwerkbezeichnung, zumindest als Standardvorgabe, erhalten muß und ihr eine Dateinamenerweiterung gegeben werden kann, sind diese beiden Angaben bei Objektmoduln nicht möglich. Laufwerkbezeichnung und Erweiterung werden jedoch für die Library-Datei benutzt, zu der ein Modul gehört.

Der Library-Verwalter bietet die Möglichkeit, allgemeine Libraries für verschiedene Programme oder besondere Libraries für spezifische Programme zu erstellen. Mit dem Library-Verwalter kann z. B. eine allgemeine Library für einen Sprach-Compiler oder eine besondere Library, die schnelles Linken und möglicherweise rationelleren Ablauf für ein einzelnes Programm ermöglicht, erstellt werden.

Individuelle Moduln innerhalb einer Library können modifiziert werden, indem die Moduln herausgelöst, die Änderungen vorgenommen und die Moduln dann wieder der Library hinzugefügt werden. Ein bereits bestehender Modul kann mit einem anderen oder mit einer neuen Fassung ersetzt werden.

3.1.2 Überblick über Library-Verwaltungsvorgänge

Der Library-Verwalter hat zwei grundsätzliche Aufgaben: Er wandelt Objektdateien in Moduln um und hängt sie an eine Library-Datei an. Er löscht Moduln von einer Library-Datei. Diese beiden Aufgaben sind Grundlage für die fünf Library-Verwaltungsfunktionen:

- Library-Datei erstellen
- Objektdatei als Modul einer Library anhängen
- Modul löschen
- Separate Objektdatei durch Kopieren eines Moduls erstellen
- Modul in der Library-Datei mit einem neuen Modul ersetzen

Während der Library-Phase löscht oder löst der Library-Verwalter zuerst Moduln heraus und hängt dann neue Moduln an. Durch diese Ablauffolge können keine Verwechslungen auftreten, wenn eine Fassung eines Moduls durch eine neue in der Library-Datei ersetzt wird. (Die Ersetzfunktion besteht aus aufeinanderfolgenden Lösch- und Anhängefunktionen.) Lösch-, Anhänge- und Herauslöse-Funktionen können in beliebiger Reihenfolge bestimmt werden.

In einem einzigen Vorgang lädt der Library-Verwalter jeden Modul in den Speicher, führt Übereinstimmungskontrolle durch und schreibt den Modul zurück in die Datei. Die Übereinstimmungskontrolle prüft, ob die syntaktische Form des Moduls korrekt und sein Dateiname einmalig ist. Wird ein Modul gelöscht, lädt der Library-Verwalter diesen Modul, schreibt ihn jedoch nicht in die Datei zurück.

Wenn der Library-Verwalter den nächsten zu speichernden Modul zurückschreibt, wird er an das Ende des zuletzt geschriebenen Moduls gestellt. Durch dieses Vorgehen wird der Dateiraum lückenlos ausgefüllt, um zu vermeiden, daß die Library-Datei größer wird als notwendig ist. Hat der Library-Verwalter die gesamte Library-Datei gelesen, hängt er neue Moduln am Dateiende an. Der Library-Verwalter erstellt ebenfalls einen Index, mit Hilfe dessen die Linker-Utility Moduln und Symbole in der Library-Datei auffinden kann. Schließlich erstellt der Library-Verwalter eine Querverweisliste der PUBLIC Symbole in der Library, falls eine solche Aufstellung angefordert wird. Zur Erstellung des Library-Verzeichnisses braucht der Library-Verwalter Zeit (in einigen Fällen bis zu 20 Sekunden).

Abbildung 3-1 illustriert die verschiedenen Library-Verwaltungsvorgänge.

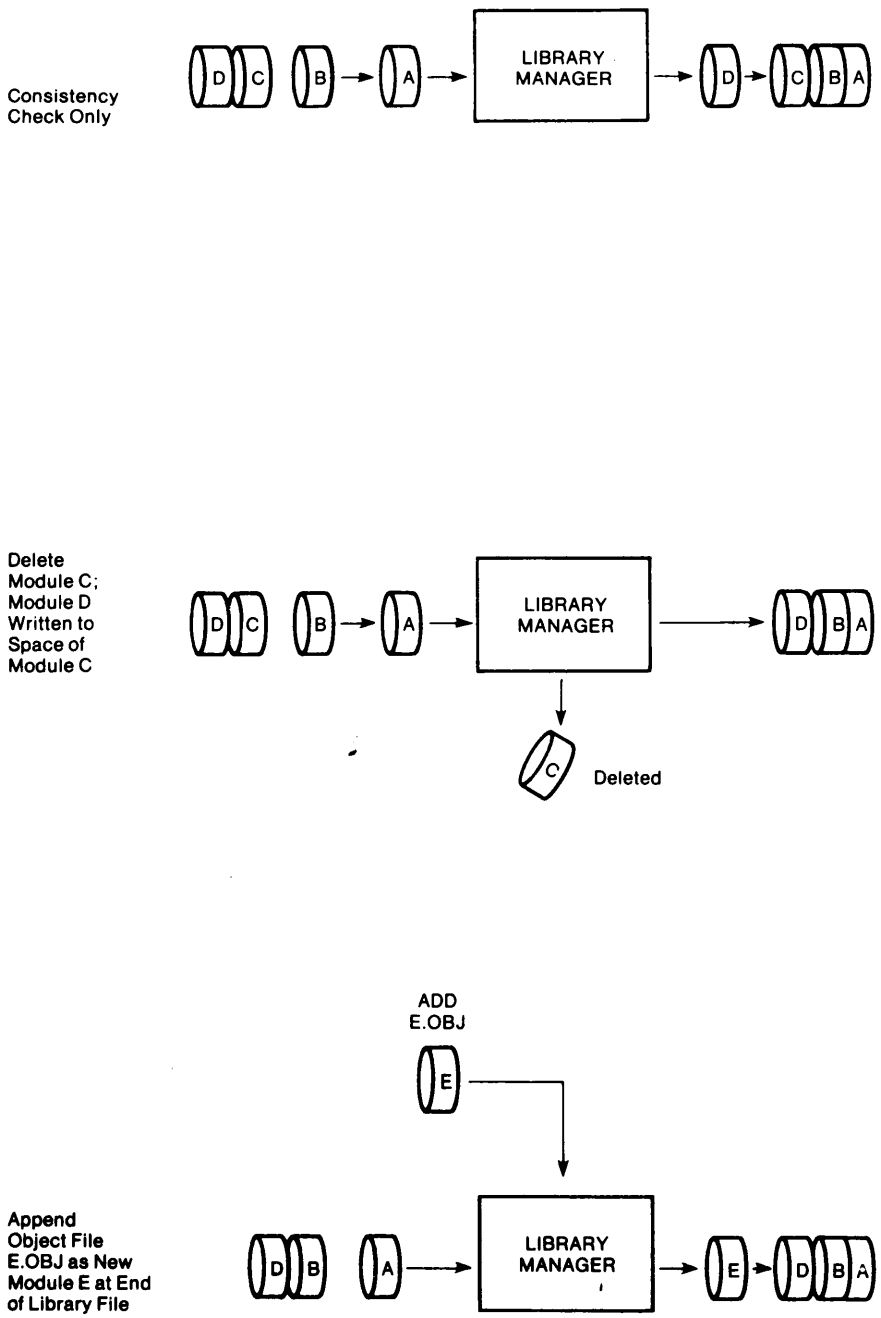


Abbildung 3-1. Library-Verwaltungsvorgänge

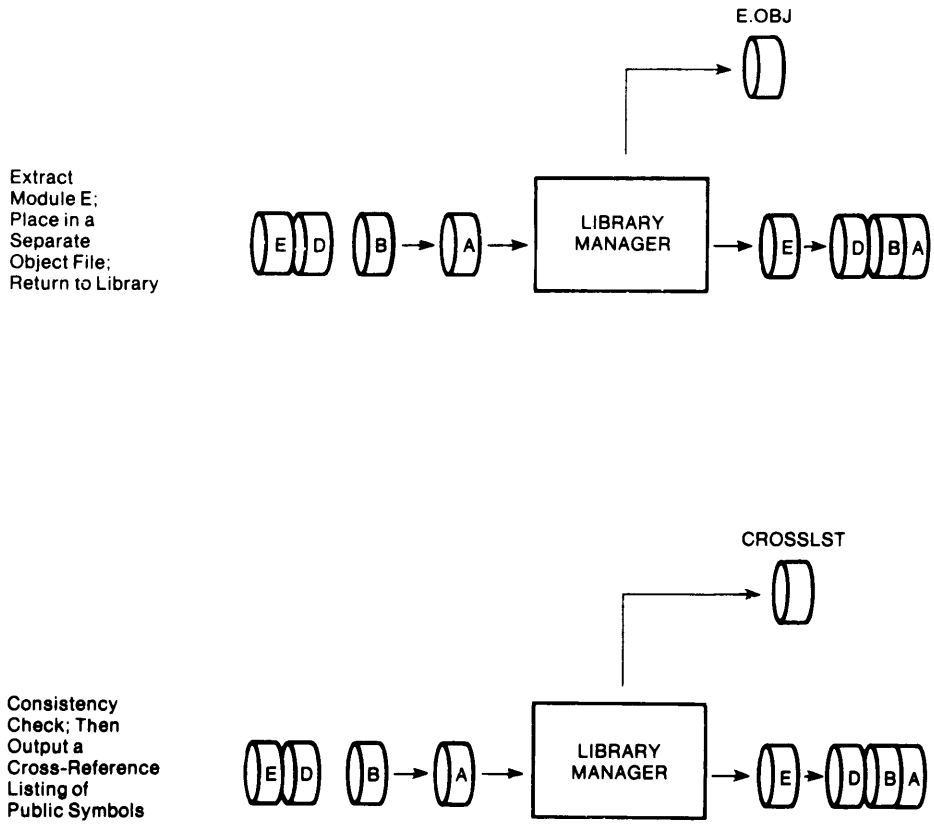


Abbildung 3-1. Library-Verwaltungsvorgänge (Forts.)

3.1.3 Erfordernisse für Library-Verwaltungssysteme

Der Library-Verwalter hat einen Mindestspeicherbedarf von 38KB, d. h. 28KB für Code und 10KB für Programmlauf.

Der Library-Verwalter kann nur dann bei einem Einzellaufwerksystem eingesetzt werden, wenn Ausgabe und Eingabe auf der gleichen physischen Disk erfolgen. Keines der Utility-Programme in diesem Paket läßt die zum Diskwechsel bei einem Einzellaufwerksystem erforderliche Zeit zu.

3.2 ABLAUF DES LIBRARY-VERWALTERS

Zum Ablauf des Library-Verwalters sind zwei Befehlsarten erforderlich: ein Befehl zum Aufruf des Library-Verwalters und Antworten auf Befehlsdialoghinweise. Im allgemeinen werden alle Befehle über die Tastatur eingegeben. Wahlweise können Antworten auf Befehlsdialoghinweise jedoch auch in einer Antwortdatei enhalten sein. Der Library-Verwalter unterstützt Sonderbefehlszeichen zum Eintrag von Befehlen.

Der Library-Verwalter läßt sich auf vier Arten aufrufen. Bei Benutzung der ersten zwei Methoden, können die übrigen Befehle als Antworten auf individuelle Dialoghinweise eingegeben werden. Die dritte Methode erlaubt, alle Befehle auf der Zeile einzutragen, die zum Aufruf des Library-Verwalters benutzt wird. Bei der vierten Methode wird eine alle notwendigen Befehle enthaltende Antwortdatei benutzt.

3.2.1 Methode 1

Zur Benutzung von Methode 1 die Programmentwicklungsoption vom Hauptsystem-Menü, dann die Library-Verwaltungsoption vom Programmentwicklungsmenü wählen (Abbildung 1-1). Der Library-Verwalter zeigt daraufhin der Reihe nach vier Dialoghinweise an. Die auf die Dialoghinweise gegebenen Antworten stellen Befehle an den Library-Verwalter zum Ausführen bestimmter Aufgaben dar.

3.2.2 Methode 2

Zur Benutzung von Methode 2 den DOS Befehlsprozessor oder die Option "Andere" vom Hauptsystem-Menü wählen. Der DOS Dialoghinweis oder ein Dateispezifikations-Dialoghinweis erscheint. Diesen Dialoghinweis folgendermaßen beantworten:

```
>LIB
```

Diese Antwort veranlaßt, daß der Library-Verwalter in den Speicher geladen wird. Der Library-Verwalter zeigt nacheinander vier Dialoghinweise an. Die auf diese Dialoghinweise gegebenen Antworten sind Befehle an den Library-Verwalter zum Ausführen bestimmter Aufgaben.

Beispiel-Antworten auf Dialoghinweise aus Methode 1 und 2:

```
Library file: pascal
Operation: -heap+heap;
List file:
```

Diese Antworten löschen den früheren Library-Modul HEAP von der Library-Datei und fügen dann eine neue Datei HEAP.OBJ als letzten Modul in der Library an.

3.2.3 Methode 3

Zur Benutzung von Methode 3 den DOS Befehlsprozessor oder die Option "Andere" vom Hauptsystem-Menü wählen. Der DOS Dialoghinweis oder ein Dateispezifikations-Dialoghinweis erscheint, der folgendermaßen beantwortet wird:

```
LIB <Library><Operationen>,<Liste>
```

Die LIB folgenden Eintragungen sind Antworten auf Befehlsdialoghinweise.

<Library>	Dateispezifikation einer Library-Datei.
<Operationen>	Reihe von Modul- oder Objektdateinamen, denen **jeweils eines der drei Befehlszeichen vorausgeht: Pluszeichen (+), Minuszeichen (-), Stern (*).**
<Liste>	Dateispezifikation der Datei, die die Querverweisliste der PUBLIC-Symbole in den Moduln der Library enthalten soll.

Ein Befehlszeichen muß allen Operationseingaben vorausgehen. Soll eine Querverweisliste erstellt werden, muß die Listdateispezifikation von der letzten Operationseingabe durch ein Komma getrennt sein. Sind eine Reihe von Operationen während einer Library-Phase durchzuführen, das Et-Befehlszeichen (&) benutzen, um den Befehlsdialoghinweis zu wiederholen, damit zusätzliche Objektdateispezifikationen und Modulnamen eingetragen werden können (siehe Abschnitt 3.4.5, Et-Zeichen).

Beispiele für Methode 3:

```
LIB PASCAL-HEAP+HEAP;
```

Dieser Befehl veranlaßt den Library-Verwalter, den Modul HEAP von der Library-Datei PASCAL.LIB zu löschen und dann die Objektdatei HEAP.OBJ als letzten Modul an PASCAL.LIB anzuhängen (der Modul trägt den Namen HEAP).

```
LIB PASCAL
```

Dieser Befehl veranlaßt den Library-Verwalter, eine Übereinstimmungskontrolle der Library-Datei PASCAL.LIB durchzuführen. Weitere Aufgaben werden nicht ausgeführt.

```
LIB PASCAL,PASCROSS.PUB
```

Dieser Befehl veranlaßt den Library-Verwalter, eine Übereinstimmungskontrolle der Library-Datei PASCAL.LIB durchzuführen und daraufhin eine Querverweis-Listdatei mit dem Namen PASCROSS.PUB zu erstellen.

3.2.4 Methode 4

Zur Benutzung von Methode 4 den DOS Befehlsprozessor oder die Option "Andere" vom Hauptsystem-Menü wählen. Der DOS Dialoghinweis oder ein Dateispezifikations-Dialoghinweis erscheint, der folgendermaßen beantwortet wird:

```
LIB @<Filespez.>
```

Methode 4 ermöglicht, die Library-Verwaltungsphase ohne interaktive Antworten auf die Library-Verwalter-Dialoghinweise durchzuführen. @<Filespez.> bezeichnet eine Antwortdatei, die Antworten auf die Library-Verwalter-Dialog- hinweise enthält. Die Antwortdatei muß vor Benutzung von Methode 4 zum Aufruf des Library-Verwalters erstellt werden.

Eine Antwortdatei enthält jeweils eine Textzeile für jeden Dialoghinweis. Die Antworten müssen in derselben Reihenfolge wie die Befehlsdialoghinweise angeordnet sein. Befehlszeichen in der Antwortdatei auf die gleiche Weise benutzen wie beim Eintasten der Antworten auf der Tastatur.

Zu Beginn der Library-Phase zeigt der Library-Verwalter abwechselnd jeden Dialoghinweis mit den Antworten aus der Antwortdatei an. Enthält die Antwortdatei nicht auf alle Dialoghinweise Antworten, werden Standardantworten benutzt.

Beispiel einer Antwortdatei für Methode 4:

```
PASCAL<RETURN>
-HEAP*FOIBLES+CURSOR+HEAP<RETURN>
CROSSLST<RETURN>
```

Diese Antwortdatei veranlaßt den Library-Verwalter, den Modul HEAP von der Datei der Library PASCAL.LIB zu löschen, den Modul FOIBLES herauszulösen und in eine Objektdatei mit dem Namen FOIBLES.OBJ zu stellen und dann die Objektdateien CURSOR.OBJ sowie HEAP.OBJ als die letzten zwei Moduln der Library anzuhängen. Daraufhin erstellt der Library-Verwalter eine Querverweis-Listdatei mit dem Namen CROSSLST.

3.3 BEFEHLSDIALOGHINWEISE

Wie bereits erwähnt, kann der Library-Verwalter Befehle erhalten, indem Antworten auf drei Textdialoghinweise eingetragen werden. Nach Beantwortung des laufenden Dialoghinweises erscheint der nächste. Wird der letzte Dialoghinweis beantwortet, setzt der Library-Verwalter seine Vorgänge fort. Ist die Library-Phase erfolgreich beendet, kehrt der Library-Verwalter an die Stelle zurück, an der er aufgerufen wurde. Verläuft die Library-Phase erfolglos, zeigt der Library-Verwalter eine Fehlermeldung an (siehe Abschnitt B.3, Library-Verwalter-Mitteilungen).

Der Library-Verwalter fragt nach dem Namen der Library-Datei, dem durchzuführenden Vorgang und der Spezifikation, die ggf. einer Querverweis-Listdatei gegeben werden soll. In den nachfolgenden Abschnitten werden diese Dialoghinweise erläutert.

3.3.1 Library File: Dialoghinweis

Der Library-Verwalter zeigt folgenden Dialoghinweis an:

```
Library file:
```

Die Dateispezifikation der zu benutzenden Library-Datei eintragen. Der Library-Verwalter geht davon aus, daß die Dateinamenerweiterung .LIB ist. Diese Erweiterung kann aufgehoben werden, wenn bei Eintrag des Library-Dateinamens eine Dateinamenerweiterung angegeben wird. Da der Library-Verwalter nur jeweils eine Library-Datei verwalten kann, kann nur ein Dateiname als Antwort auf diesen Dialoghinweis eingetragen werden. Zusätzliche Antworten, mit Ausnahme des Strichpunkt-Befehlszeichens, werden ignoriert.

Wird ein Library-Dateiname mit direkt folgendem Strichpunkt-Befehlszeichen eingetragen, führt der Library-Verwalter nur eine Übereinstimmungskontrolle durch und kehrt dann zum Betriebssystem zurück. Der Library-Verwalter meldet in der Datei aufgetretene Fehler.

Ist der eingetragene Dateiname nicht vorhanden, zeigt der Library-Verwalter folgenden Dialoghinweis an:

```
Library file does not exist.  Create?
```

Der Library-Verwalter sucht die Antwort nach dem Buchstaben J ab, der das erste Zeichen sein muß. Wird ein anderes Zeichen zuerst eingegeben, bricht der Library-Verwalter den Vorgang ab und kehrt zum Betriebssystem zurück.

3.3.2 Operation: Dialoghinweis

Der Library-Verwalter zeigt folgenden Dialoghinweis an:

```
Operation:
```

Ein oder mehrere Befehlszeichen zum Manipulieren von Moduln eintragen (+, - und *), denen direkt (kein Zwischenraum) ein Modulname oder eine Objektdatei-Spezifikation folgt. Das Pluszeichen hängt eine Objektdatei als letzten Modul an die Library-Datei an. Das Minuszeichen löscht einen Modul von der Library-Datei. Der Stern löst einen Modul aus der Library heraus und stellt ihn in eine separate Objektdatei, wobei der Modulname zum Dateinamen mit einer Dateinamenerweiterung .OBJ wird. Es gibt keine Standardvorgabe.

Der Library-Verwalter ermöglicht, Vorgänge in Moduln und Objektdateien in beliebiger Reihenfolge einzutragen. Darauf achten, daß vor jedem Modul- oder Objektdateinamen eines der Befehlszeichen für Operationen eingetragen ist.

Näheres über die Ablauffolge und Bearbeitung der Moduln durch den Library-Verwalter befindet sich in Abschnitt 3.4, Befehlszeichen.

ANMERKUNG:
Der Library-Verwalter erlaubt Eingabe der Operationsantworten in beliebiger Reihenfolge. Der Ablauf weicht nie von der in Abschnitt 3.1.2, Überblick über Library-Verwaltungsvorgänge, beschriebenen Folge ab.

3.3.3 List File: Dialoghinweis

Der Library-Verwalter zeigt folgenden Dialoghinweis an:

List file:

Die Dateispezifikation für eine Querverweis-Listdatei eintragen, falls eine Querverweisliste der PUBLIC-Symbole in den Moduln der Library-Datei gewünscht wird. Wird keine Dateispezifikation angegeben, erstellt der Library-Verwalter keine Querverweisliste.

Da die Antwort auf den Listdatei-Dialoghinweis eine Dateispezifikation ist, kann zusammen mit dem Dateinamen eine Laufwerk- (oder Geräte-) Bezeichnung und eine Dateinamenerweiterung angegeben werden. Der Library-Verwalter gibt der Listdatei keine Standard-Dateinamenerweiterung.

Die Querverweis-Listdatei enthält zwei Aufstellungen. Die erste ist eine alphabetische Aufstellung aller PUBLIC-Symbole. Jedem Symbolnamen folgt der entsprechende Modulname. Die zweite Aufstellung enthält die Moduln in der Library in alphabetischer Reihenfolge. Unter jedem Modulnamen befindet sich eine alphabetische Aufstellung der PUBLIC-Symbole in diesem Modul. Der Library-Verwalter erstellt die Liste nach Abschluß aller anderen Moduloperationen.

Bei Eintrag eines Library-Dateinamens mit direkt folgendem Komma und Listdateispezifikation führt der Library-Verwalter die Übereinstimmungskontrolle der Library-Datei durch und erstellt dann die Querverweis-Listdatei.

3.4 BEFEHLSZEICHEN

Der Library-Verwalter stellt sechs Befehlszeichen zur Verfügung, drei für erforderliche und drei für wahlweise Antworten auf Dialoghinweise. Die ersten drei sind: Pluszeichen (+), Minuszeichen (-) und Stern (*). Die Antworten auf den Operationsdialoghinweis erfordern den Gebrauch eines oder mehrerer dieser drei Zeichen. Die Zeichen für wahlweise Befehle sind: Strichpunkt (;), Et-Zeichen (&) und UMSCH + ANNULLIER.

3.4.1 Pluszeichen

Als Antwort auf den Operationsbefehl hängt das Pluszeichen (+), dem eine Objektdateispezifikation folgt, die Objektdatei als letzten Modul an die Library an. Trifft der Library-Verwalter auf das Pluszeichen, geht er davon aus, daß die Dateinamenerweiterung .OBJ ist. Diese Erweiterung kann durch Angabe einer anderen Erweiterung aufgehoben werden.

Der Library-Verwalter entfernt die Laufwerkbezeichnung und die Erweiterung von der Objektdateispezifikation und behält nur den Dateinamen bei. Falls die an eine Library als Modul anzuhängende Objektdatei z. B. B: CURSOR.OBJ ist, folgende Antwort auf den Operationsdialoghinweis eintragen:

```
+B: CURSOR.OBJ
```

Diese Antwort veranlaßt den Library-Verwalter, B: und .OBJ zu entfernen und nur CURSOR zu belassen, aus dem ein Modul mit dem Namen CURSOR in der Library wird.

3.4.2 Minuszeichen

Das Minuszeichen (-), dem ein Modulname folgt, löscht diesen Modul von der Library-Datei. Der Library-Verwalter schließt dann den durch Löschen freiwerdenden Dateiraum. Dadurch wird verhindert, daß sich die Datei unnötig durch Leerraum vergrößert. (Neue Moduln, auch Ersatzmoduln, werden am Dateiende hinzugefügt.)

3.4.3 Stern

Der Stern (*), dem ein Modulname folgt, kopiert den Modul von der Library-Datei in eine separate Objektdatei. Der Modul in der Library bleibt erhalten. Der Modulname wird als Dateiname benutzt. Der Library-Verwalter fügt die Standard-Laufwerkbezeichnung und Dateinamenerweiterung .OBJ hinzu.

Angenommen, der herauszulösende Modul ist CURSOR und das Standardlaufwerk A. Ist die Antwort auf den Operationsdialoghinweis *CURSOR, kopiert der Library-Verwalter den Modul mit dem Namen CURSOR von der Library-Datei in eine Objektdatei. Die Dateispezifikation der Objektdatei ist A:CURSOR.OBJ.

Die Laufwerkbezeichnung und Dateinamenerweiterung kann nicht aufgehoben werden. Die Datei kann jedoch umbenannt werden und eine neue Dateinamenerweiterung erhalten und/oder auf ein neues Plattenlaufwerk kopiert werden und einen neuen Dateinamen und/oder eine neue Dateinamenerweiterung erhalten.

3.4.4 Strichpunkt

Einen Strichpunkt (;) mit direkt folgender Zeilenschaltung nach Beantwortung des ersten Dialoghinweises (Library file:) benutzen, um Standardantworten auf die übrigen Dialoghinweise zu wählen. Diese Eigenschaft ist zeitsparend und erübrigt die Beantwortung weiterer Dialoghinweise. Zum Beispiel:

```
Library file:  FUN
Operation:  +CURSOR;
```

Der übrige Dialoghinweis erscheint nicht; der Library-Verwalter benutzt den Standardwert (keine Querverweisdatei).

Wird ein Library-Dateiname mit direkt folgendem Strichpunkt eingetragen, liest der Library-Verwalter die Library-Datei und führt eine Übereinstimmungskontrolle durch. Die Moduln in der Library-Datei werden nicht verändert.

ANMERKUNG:
Sobald ein Strichpunkt eingetragen ist, können Dialoghinweise in dieser Library-Phase nicht mehr beantwortet werden. Der Strichpunkt sollte daher nicht zum Überspringen von Dialoghinweisen verwendet werden. Sollen Dialoghinweise ignoriert werden, Zeilenschaltung benutzen.

3.4.5 Et-Zeichen

Dieses Befehlszeichen nur mit dem Operationsdialoghinweis benutzen. Die Länge der Dialoghinweis-Antwortzeile ist 255 Zeichen. Bei einer größeren Anzahl von Antworten auf den Operationsdialoghinweis ein Et-Zeichen an das Zeilenende setzen. Der Library-Verwalter zeigt den Operationsdialoghinweis erneut auf der nächsten Zeile an. Weitere Antworten können dann eingetragen werden.

Das Et-Zeichen so oft wie nötig benutzen. Der Library-Verwalter kann viele Funktionen während einer Library-Phase durchführen. Die Anzahl der anhängbaren und der ersetz- oder kopierbaren Moduln ist lediglich durch den verfügbaren Diskspeicherraum begrenzt. Die Anzahl der löschbaren Moduln ist nur durch die Anzahl der Moduln in der Library-Datei begrenzt. Zum Beispiel:

```
Library file:  FUN
Operation:  -HEAP*FOIBLES*INIT+CURSOR&
Operation:  +HEAP+ASSUME+RIDE;
```

Der Library-Verwalter löscht den Modul HEAP, kopiert die Moduln FOIBLES und INIT (wobei zwei Dateien erstellt werden, FOIBLES.OBJ und INIT.OBJ) und hängt dann die Objektdateien CURSOR, HEAP, ASSUME und RIDE an.

3.4.6 UMSCH + ANNULLIER

UMSCH + ANNULLIER jederzeit zum Abbruch der Library-Phase benutzen. Wurde eine fehlerhafte Antwort, z. B. ein falscher oder falsch buchstabierter Datei- oder Modulname, eingetragen, UMSCH + ANNULLIER drücken, um den Library-Verwalter zu verlassen; dann den Library-Verwalter wieder aufrufen. Wird der Fehler vor tatsächlicher Eingabe in das System bemerkt, können die Zeichen auf dieser Zeile gelöscht werden.

KAPITEL 4
DEBUGGER

4.1 EINFÜHRUNG

Der Debugger (DEBUG.COM) bietet eine kontrollierte Austestumgebung für Binärdateien und ausführbare Objektdateien. Assembler und Compiler erstellen Binärdateien. Der Linker erstellt ausführbare (.EXE) Dateien.

Obwohl der Debugger für Programme in jeder kompilierten Sprache benutzt werden kann, erfordern viele Debugger-Funktionen Kenntnis der Assemblerbegriffe. Für eine Reihe von Debugger-Funktionen ist ebenfalls Kenntnis der Systemarchitektur, des Betriebssystems und der in Anhang A enthaltenen Definitionen erforderlich.

Der Editor (siehe Kapitel 1) wird zur Änderung von Quelldateien benutzt. Der Debugger ist das Gegenstück des Editors für Binärdateien. Bei Benutzung des Debuggers ist es nicht notwendig, ein Programm zu reassemblieren, um festzustellen, ob ein Problem durch eine geringfügige Änderung behoben wurde. Während des Gebrauchs des Debuggers kann der Inhalt einer Datei oder eines CPU Registers geändert und ein Programm sofort wieder ausgeführt werden, um die Änderungen zu prüfen.

Die Debugger-Befehle können jederzeit mit der ANNULLIER-Taste abgebrochen werden. Mit KTRL + S hält die Anzeige an und kann gelesen werden, bevor sie vom Bildschirm abrollt. Zum Wiederanlauf der Anzeige KTRL + Q drücken. Bei Benutzung des Debuggers gelten zusätzlich das Steuerzeichen und die Sonderaufbereitungsfunktionen, die auf DOS Befehlsstufe verfügbar sind.

4.2 DEBUGGER AUFRUFEN

Drei Methoden stehen zum Aufruf des Debuggers zur Verfügung. Bei Methode 1 werden WANG PC Menüs benutzt. Vom Hauptmenü "Programmentwicklung" wählen. Der darauffolgende Bildschirm zeigt das Programmentwicklungsmenü (Abbildung 1-1) an.

Vom Programmentwicklungsmenü die Debugger-Option wählen. Der Debugger-Dialoghinweis, eine rechte spitze Klammer (>), erscheint am Bildschirm. Dieser Dialoghinweis wird durch Eintrag der in Abschnitt 4.6 beschriebenen Debugger- Befehle beantwortet.

Bei der zweiten Methode wird der Debugger über die DOS Befehlsprozessoroption (COMMAND.COM) im Hauptmenü aufgerufen. Am Bildschirm erscheint der DOS Dialoghinweis A:, wobei A das Standardlaufwerk darstellt. Den Dialoghinweis mit folgendem Format beantworten:

DEBUG [<Filespez.> [<Argumentliste>]]

Wird eine Datei bezeichnet, lädt der Debugger diese Datei in den Speicher. Ist <Filespez.> vorhanden, kann eine <Argumentliste> bestimmt werden. Die <Argumentliste> enthält Dateinamenparameter und Schalter, die an das auszutestende Programm weitergegeben werden sollen. Wie bei Aufruf vom DOS Befehlsprozessor lädt der Debugger die bezeichnete Datei in den Speicher. Der Debugger-Dialoghinweis erscheint, und das Programm kann durch Beantwortung des Dialoghinweises mit den Debugger-Befehlen bearbeitet werden.

Bei der dritten Methode zum Aufruf des Debuggers wird die Option "Andere" im Hauptmenü gewählt. Ein Dialoghinweis für eine Dateispezifikation erscheint am Bildschirm. Diesen Dialoghinweis wie den DOS Befehlsprozessor-Dialoghinweis beantworten.

Mit diesen drei Methoden kann der Debugger aufgerufen werden, ohne eine auszutestende Datei anzugeben. Debug-Programme vom Programmentwicklungsmenü wählen oder den DOS Dialoghinweis bzw. den Dateispezifikations-Dialoghinweis der Option "Andere" mit DEBUG beantworten. Der Debugger-Dialoghinweis erscheint. Nun können die in Abschnitt 4.6 beschriebenen N- und L-Befehle benutzt werden, um eine auszutestende Datei zu bestimmen und in den Speicher zu laden.

4.3 INITIALISIERUNG

Wenn das auszutestende Programm keine .EXE Datei ist, haben 8086 Kennzeichen und Register zu Beginn des Debugger-Programms folgende Werte:

- Die Segmentregister (CS, DS, ES und SS) enthalten das erste Segment nach dem Debugger-Programm.
- Der Instruktionsanzeiger (IP) enthält 0100H.
- Der Stapelanzeiger (SP) enthält das Ende des Stapelsegments oder die untere Adresse des Übergangsteils von COMMAND.COM (siehe Abschnitt A.3.3), je nachdem welche niedriger ist. Die Segmentgröße bei Offset 6 im Programmsegmentvorsatz (PSH) des auszutestenden Programms wird um X'100' reduziert, um einen Stapel dieser Größe zu ermöglichen.
- Die anderen Register (AX, BX, CX, DX, BP, SI und DI) enthalten Null. Wird der Debugger jedoch mit einem <Filespez.> aufgerufen, enthalten die Register BX und CX die Länge der Datei in Bytes.

- Die Kennzeichen werden gelöscht. Die gelöschten Werte sind in Abschnitt 4.6.14, R-Befehl (Register), enthalten.

- Die Standarddisk-Übertragungsadresse ist Offset 80H im Codesegment.

Wenn der Debugger eine .EXE Datei lädt, führt er die notwendige Verschiebung durch und setzt die Segmentregister, den Instruktionsanzeiger und den Stapelanzeiger auf die in der Datei definierten Werte fest. Die DS und ES Register enthalten jedoch die Adresse des PSH am untersten freien Segment. Das CX Register enthält Null.

VORSICHT:
Der Debugger setzt bei Aufruf einen PSH bei Offset 0 im Programmarbeitsbereich. Wird bei Aufruf des Debuggers keine Dateispezifikation angegeben, wird ein Standard-PSH aufgestellt. Der Standard-PSH kann überschrieben werden. Sobald der Debugger eine Dateispezifikation erhalten hat, darf der PSH jedoch nicht auf einen niedrigeren Offset als 5CH abgeändert werden. Die Modifikationen in diesem Bereich könnten einen Systemausfall bei Austritt aus dem Debugger verursachen. Es wird empfohlen, keine Modifikationen vor 100H vorzunehmen.

Wird der entsprechende Schalter angegeben, wenn der Linker die Datei erstellt (siehe Abschnitt 2.7), lädt der Debugger das Programm am oberen Speicherende.

Der Debugger setzt voraus, daß eine .HEX Datei eine ASCII-Darstellung der Hexadezimalzeichen enthält. Während des Ladevorgangs konvertiert der Debugger diese Dateien zu Binärdateien.

4.4 DEBUGGER BENUTZEN

Jeder Debugger-Befehl besteht aus einem Einzelbuchstaben, dem ein oder mehrere Parameter folgen.

Bei Auftritt eines Syntaxfehlers druckt der Debugger die Befehlszeile erneut und zeigt die Fehlerstelle mit einem Aufwärtspfeil (↑) und dem Wort "error" an. Zum Beispiel:

```
dcs:100 cs:110
      ↑ error
```

Alle Befehle können in beliebiger Kombination von Groß- und Kleinbuchstaben eingetragen werden.

Die Debugger-Befehle werden in Tabelle 4-1 zusammengefaßt und eingehend mit Beispielen in Abschnitt 4.6 beschrieben. Die Debugger-Befehlsparameter werden in Abschnitt 4.5 erläutert.

Tabelle 4-1. Debugger-Befehle

Funktion	DEBUG-Befehl
Assemblieren	A<Adresse>
Vergleichen	C<Bereich> <Adresse>
Speicherauszug	D[<Adresse>] D[<Bereich>]
Eintragen	E<Adresse> [<Liste>]
Füllen	F<Bereich> <Liste>
Go	G[=<Adresse>] [<Adresse>...]
Hex	H<Adresse> <Adresse>
Eingabe	I<Wert>
Laden	L[<Adresse> [<Laufwerk> <Satz> <Wert>]]
Verschieben	M<Bereich> <Adresse>
Name	N<Filespez.>[/S...] [<Filespez.>[/S...]...]
Ausgabe	O<Wert> <Byte>
Abbrechen	Q
Register	R[<Registername>]
Suchen	S<Bereich> <Liste>
Ablaufverfolgung	T[=<Adresse>] [<Wert>]
Disassemblieren	U[<Adresse>] U[<Bereich>]
Schreiben	W[<Adresse> [<Laufwerk> <Satz> <Wert>]]
(E/O-Gerät ändern)	X<Gerät>

4.5 DEBUGGER-BEFEHLSPARAMETER

Wie aus Tabelle 4-1 ersichtlich ist, nehmen alle Debugger-Befehle Parameter an, mit Ausnahme des Q-Befehls. Parameter können durch Begrenzungssymbole (Leerstellen oder Kommas) getrennt werden; ein Begrenzungssymbol ist jedoch nur zwischen zwei aufeinanderfolgenden Hexadezimalwerten erforderlich. Folgende Befehle sind demnach gleichwertig:

Debugger

```
00dcs:100 110
        d cs:100 110
        d,cs:100,110
```

Die nachstehende Aufstellung zeigt die Debugger-Parameter und ihre Definitionen:

Parameter	Definition
<Adresse>	Entweder eine alphabetische Segmentregister-Bezeichnung oder eine 1- bis 4-stellige Segmentadresse, der ein Doppelpunkt und ein Offset-Wert folgen. Zum Beispiel: CS:0100 04BA:0100 Der Doppelpunkt ist zwischen einer Segmentbezeichnung (numerisch oder alphabetisch) und einem Offset erforderlich. Die Segmentbezeichnung oder Segmentadresse kann ausgelassen werden; in diesem Fall gilt das Standardsegment. DS ist das Standardsegment für alle Befehle, mit Ausnahme von A, G, L, T, U und W, deren Standardsegment CS ist. Alle numerischen Werte sind hexadezimal. Die <Adresse>n müssen sich auf existierende Speicherstellen beziehen. Anderenfalls sind die Ergebnisse unbestimmt.
<Byte>	Ein 1- oder 2-stelliger Hexadezimalwert, der in eine Adresse oder in ein Register gesetzt oder daraus gelesen wird.
<Gerät>	Ein reservierter Name für ein Gerät.
<Laufwerk>	Ein 1- oder 2-stelliger Hexadezimalwert, der bestimmt, von welchem Laufwerk eine Datei zu laden oder zu welchem Laufwerk eine Datei zu schreiben ist. Die zulässigen Werte sind 0 bis 3F. Die gegenwärtig den Laufwerken zugewiesenenen Werte sind wie folgt: <Wert> <Laufwerk> 0 A 1 B 2 C
<Filespez.>	Ein 1- bis 8-Zeichen langer Dateiname, dem wahlweise eine Laufwerkbezeichnung vorausgehen oder ein Punkt und eine 1- bis 3-Zeichen-Erweiterung folgen kann.

Parameter	Definition
<Liste>	Eine Reihe von <Byte> Werten oder Zeichenfolgen (strings). <Liste> muß der letzte Parameter auf der Befehlszeile sein.

Folgendes Beispiel enthält drei Elemente in <Liste>, zwei Byte-Eintragungen und eine Zeichenfolgen-Eintragung:

```
fcs:100 L 10 42 'abc'
```

<Bereich>

Entweder zwei <Adresse>n (<Adresse> <Adresse>) oder eine <Adresse>, ein L und ein <Wert> (<Adresse> L <Wert>), wobei <Wert> die Anzahl der Zeilen oder Bytes ist, mit denen der Befehl arbeitet. Werden zwei <Adresse>n angegeben, muß die zweite höher als die erste sein und darf nur durch einen Offset angegeben werden. Zum Beispiel:

```
CS:100 110
CS:100 L 10
```

Folgender <Bereich> ist ungültig, da die zweite Adresse ein Register umfaßt:

```
CS:100 CS:110
       ↑ error
```

Die Grenze für <Bereich> ist 10000H. Um einen <Wert> von 10000H in vier Ziffern anzugeben, 0000 (oder 0) eintragen.

<Sektor>

Ein 1- bis 4-stelliger Hexadezimalwert zur Bestimmung des physischen Sektors auf der Disk, an dem der Ladevorgang beginnt.

<Registername>

Entweder ein 2-Zeichen-Registername oder das Zeichen F für Kennzeichen (Flag).

<Zeichenfolge>

Eine beliebige Anzahl von Zeichen, die in Anführungszeichen eingeschlossen sind. Die ASCII-Werte der Zeichen in der Zeichenfolge werden zu einer <Liste> von Bytewerten. Einzel- (') oder Doppelanführungsstriche (") können verwendet werden. Innerhalb der <Zeichenfolge>n kann der zweite Satz der Anführungszeichen frei als Literal erscheinen. Falls die Begrenzungssymbol-Anführungsstriche innerhalb einer <Zeichenfolge> erscheinen müssen, muß jeder Anführungs- strich, ob einzeln oder doppelt, wiederholt werden. Nachstehende Zeichenfolge ist z. B. gültig:

```
'Diese "Zeichenfolge" ist o. k.'
'Diese ''Zeichenfolge'' ist o. k.'
```

Diese Zeichenfolge ist jedoch ungültig:

```
'Diese 'Zeichenfolge' ist nicht o. k.'
```

Diese Zeichenfolgen sind gültig:

```
"Diese 'Zeichenfolge' ist o. k."
"Diese ""Zeichenfolge"" ist o. k."
```

Diese Zeichenfolge ist jedoch ungültig:

```
"Diese "Zeichenfolge" ist nicht o. k."
```

Die wiederholten Anführungsstriche in der folgenden Zeichenfolge sind unnötig und sollten nur dann stehen, wenn sie Teil der Zeichenfolge sind:

```
"Dies ist eine ''Zeichenfolge''."
'Dies ist eine ""Zeichenfolge"".'
```

<Wert> Ein 1- bis 4-stelliger Hexadezimalwert, der eine Anschluß- stellennummer oder die Anzahl der Bytes, die bearbeitet werden sollen, angibt bzw. bestimmt, wieviele Male ein Befehl seine Funktionen wiederholen soll.

4.6 DEBUGGER-BEFEHLE

Dieser Abschnitt erläutert die Funktionen und Syntax der Debugger-Befehle. Beispiele für jeden Befehl sind ebenfalls enthalten.

4.6.1 A-Befehl (Assemblieren)

Funktion

Dieser Befehl konvertiert MACRO-86 Instruktionen in Binärform und speichert sie an der bezeichneten Adresse.

Format

A <Adresse>

<Adresse>	entweder eine alphabetische Segmentregister-Bezeichnung oder eine 1- bis 4-stellige Segmentadresse, der ein Doppelpunkt und ein Offset-Wert folgen. Das Standardsegment ist CS.

Bemerkungen

1. Bei Eingabe dieses Befehls erscheint der Debugger-Dialoghinweis auf der nächsten Zeile; die angegebene Adresse folgt. Den Dialoghinweis beantworten, indem eine zu assemblierende Anweisung eingegeben und dann Zeilenschaltung gedrückt wird. Der Debugger konvertiert die Anweisung zu Maschinencode, setzt die Werte in den Speicher und zeigt die Adresse der nächsten verfügbaren Speicherstelle an.

2. Um den Befehl zu beenden, Zeilenschaltung drücken, ohne eine Antwort auf den Dialoghinweis einzutragen.

3. Der Debugger anwortet auf unzulässige Anweisungen mit der Mitteilung:

   ↑ Error

und nochmaliger Anzeige der jeweiligen Assemblier-Adresse.

4. Im allgemeinen ist die Syntax der Assemblier-Sprachanweisungen, die im A-Befehl benutzt werden, standardmäßige 8086/8088 Assemblersprache. Geringfügige Ausnahmen zu dieser Regel sind:

   a. Der Debugger setzt voraus, daß alle eingetragenen numerischen Werte hexadezimal sind.

   b. Mehrere Präfixe (Sperren, Wiederholen und Segment-Override) können in einer Anweisung erscheinen, aber alle Präfixe müssen dem Op-Code der Anweisung vorausgehen. Ein Präfix kann ebenfalls auf einer separaten Zeile eingetragen werden.

   c. Der Debugger unterscheidet folgendermaßen zwischen Byte- und Wortfolgeinstruktionen:

Byte	Wort
LODSB	LODSW
STOSB	STOSW
SCASB	SCASW
MOVSB	MOVSW
CMPSB	CMPSW

d. Der mnemonische Code für Far Return ist RETF. Der Assembler assembliert automatisch kurze, nahe und weite Sprünge und Aufrufe, je nach Byteverschiebung an die Bestimmungsortadresse. Die normale Interpretation der Byteverschiebung kann mit dem NEAR- oder FAR-Präfix aufgehoben werden. Zum Beispiel:

```
0100:500 JMP 502        ; kurzer Sprung von 2 Bytes
0100:0502 JMP NEAR 505  ; naher Sprung von 3 Bytes
0100:0505 JMP FAR 50A   ; weiter Sprung von 5 Bytes
```

Das Präfix NEAR kann auf NE abgekürzt werden, das FAR-Präfix läßt jedoch keine Abkürzung zu.

e. Die mnemonischen Codes für Segment-Override sind CS:, DS:, ES: und SS:.

f. Operanden, die sich entweder auf ein Byte oder ein Wort beziehen können, sind unklar. In diesen Fällen muß das Präfix WORD PTR bzw. BYTE PTR vorausgehen. Diese Präfixe können auf BY und WO abgekürzt werden. Zum Beispiel:

```
INC     BYTE PTR [BP]
NOT     WO [1234]
```

g. Operanden, die den Speicher direkt adressieren, müssen in eckige Klammern eingeschlossen sein, um sie von umgebenden Werten abzuheben. Zum Beispiel:

```
ADD     AX,5    ;5 zum Register AX hinzufügen
ADD     AX,[5]  ;Inhalt der Speicherstelle 5 zu AX hinzufügen
```

h. Zwei neue Pseudo-Instruktionen werden eingeschlossen. Der DB-Op-Code assembliert Bytewerte direkt in den Speicher. Der DW-Op-Code assembliert Wortwerte direkt in den Speicher. Zum Beispiel:

```
DB      1,2,3,4,"THIS IS AN EXAMPLE"
DB      'THIS IS A QUOTE: "'
DB      "THIS IS A QUOTE: '"
DW      1000,2000,3000,"I CAN DO IT TOO!"
```

i. Alle Formen indirekter Befehle des Registers werden unterstützt. Zum Beispiel:

```
ADD     BX,34[BP+2].[SI-1]
POP     [BP+DI]
PUSH    [SI]
```

j. Alle Op-Code-Synonyme werden unterstützt. Zum Beispiel:

```
LOOPZ 100
LOOPE 100
JA    200
JNBE  200
```

Beispiel

```
>a 3fa:100
>03fa:0100  MOV  BX,AX          ; BB CB
>03fa:0102  CALL [BX].25        ; FF 57 25
```

4.6.2 C-Befehl (Vergleichen)

Funktion

Dieser Befehl vergleicht den Speicherbereich, dessen Anfangspunkt und Länge in <Bereich> bestimmt sind, mit einem Bereich der gleichen Größe, der bei <Adresse> beginnt. Er zeigt ebenfalls alle Unterschiede in den Bereichen an.

Format

C<Bereich> <Adresse>

<Bereich>	entweder zwei <Adressen> oder eine <Adresse>, ein L und ein <Wert> (<Adresse> L <Wert>). Eine zweite <Adresse> muß höher als die erste und darf nur ein Offset sein.
<Adresse>	entweder eine alphabetische Segmentregister-Bezeichnung oder eine 1- bis 4-stellige Segmentadresse, der ein Doppelpunkt und ein Offset-Wert folgen. Das Standardsegment ist DS.

Bemerkungen

1. Sind die beiden Speicherbereiche identisch, erscheint keine Anzeige. Der C-Befehl bricht ab, und der Debugger-Dialoghinweis erscheint wieder am Bildschirm.

2. Weisen die beiden Bereiche Unterschiede auf, erscheinen sie in folgendem Format:

<Adresse1> <Byte1> <Byte2> <Adresse2> ...

Beispiele

Folgende Befehle haben das gleiche Ergebnis:

```
C100,200 300

C100L100 300
```

Jeder Befehl vergleicht den Speicherblock von 100H bis 200H mit dem Speicherblock von 300H bis 400H.

4.6.3 D-Befehl (Speicherauszug)

Funktion

Dieser Befehl zeigt den Inhalt einer Reihe von Bytes im Speicher an.

Formate

1. D[<Adresse>]

<Adresse>	entweder eine alphabetische Segmentregister-Bezeichnung oder eine 1- bis 4-stellige Segment- adresse, der ein Doppelpunkt und ein Offset-Wert folgen. Das Standardsegment ist DS.

2. D<Bereich>

<Bereich>	entweder zwei <Adressen> oder eine <Adresse>, ein L und ein <Wert> (<Adresse> L <Wert>). Eine zweite <Adresse> muß höher als die erste und darf nur ein Offset sein.

Bemerkungen

1. Wird nur eine einzelne Adresse angegeben, zeigt der Debugger den Inhalt der 128 Bytes, die an dieser Adresse beginnen. Wird ein Bereich von Adressen angegeben, zeigt der Debugger den Inhalt des Bereichs an. Wird der D-Befehl ohne Parameter eingetragen, zeigt der Debugger den Inhalt der 128 Bytes an, die an der aktuellen Stelle im DS-Segment beginnen. Für den ersten D-Befehl ist die aktuelle Stelle bei Offset X'100'. Für nachfolgende Befehle liegt die aktuelle Stelle nach der zuletzt angezeigten Adresse.

2. Die Anzeige besteht aus zwei Teilen: einem Teil, der jedes Byte als Hexadezimalwert und einem Teil, der die Bytes als ASCII-Zeichen anzeigt. Bytes, die nicht druckbare Zeichen enthalten, werden durch einen Punkt (.) im ASCII-Teil der Anzeige bezeichnet.

3. Jede Anzeigenzeile zeigt 16 Bytes mit einem Bindestrich zwischen dem achten und neunten Byte.

4. Jede angezeigte Zeile beginnt an einer 16-Byte-Grenze, es sei denn, die im Befehl angegebene Anfangsadresse ist nicht auf einer Grenze. In diesem Fall kann die erste Zeile weniger als 16 Bytes enthalten, und die zweite Zeile beginnt an einer Grenze.

Beispiele

Angenommen, folgender Befehl wird eingetragen:

```
dcs:100 110
```

Die nachstehende Anzeige (hier aus Platzgründen auf zwei Zeilen) erscheint:

```
04BA:0100 42 45 52 54 41 20 54 00-20 42 4F 52 4C 41 4E
    44 BERTA T. BORLAND
04BA:0100 2E
```

Angenommen, folgender Befehl wird eingetragen:

```
D
```

Der Debugger zeigt 128 Bytes an, wobei jede Zeile der Anzeige, wie oben unter Bemerkungen beschrieben, formatiert wird. Jede Zeile der Anzeige beginnt mit einer Adresse, die um 16 größer ist als die Adresse auf der vorhergehenden Zeile. Durch nachfolgende D-Befehle, die ohne Parameter eingetragen werden, erscheinen die Bytes direkt nach den zuletzt angezeigten.

Angenommen, folgender Befehl wird eingetragen:

```
DCS:100 L 20
```

Die Anzeige wird wie oben beschrieben formatiert, alle Bytes für 20H Bytes erscheinen jedoch.

Angenommen, folgender Befehl wird eingetragen:

```
DCS:100 115
```

Die Anzeige wird wie oben beschrieben formatiert, aber alle Bytes im Bereich der Zeilen 100H bis 115H im Segment CS erscheinen.

4.6.4 E-Befehl (Eintragen)

Funktion

Dieser Befehl zeigt ein Byte nach dem anderen an und ermöglicht Modifikation des Inhalts.

Format

E<Adresse> [<Liste>]

<Adresse>	entweder eine alphabetische Segmentregister-Bezeichnung oder eine 1- bis 4-stellige Segmentadresse, der ein Doppelpunkt und ein Offset-Wert folgen. Das Standardsegment ist DS.
<Liste>	eine Reihe von <Byte>-Werten oder Zeichen, die in Anführungszeichen eingeschlossen ist.

Bemerkungen

1. Wird die wahlweise <Liste> hexadezimaler Werte eingetragen, erfolgt die Ersetzung von Bytewerten automatisch. Bei Auftritt eines Fehlers bleiben die Bytewerte unverändert.

2. Wird die <Adresse> ohne wahlweise <Liste> eingetragen, zeigt der Debugger die Adresse und deren Inhalt an, wiederholt dann die Adresse auf der nächsten Zeile und wartet auf eine Eingabe. Der E-Befehl wartet zu diesem Zeitpunkt auf einen der folgenden Vorgänge:

   a. Einen Bytewert durch einen anderen einzutastenden Bytewert ersetzen. Der Wert wird einfach nach dem aktuellen Wert auf der Cursorposition eingetastet. Falls der eingegebene Wert kein gültiger Hexadezimalwert ist oder falls mehr als zwei Ziffern eingetastet werden, ertönt ein akustisches Signal und das ungültige oder überflüssige Zeichen wird nicht in den Speicher gesetzt.

   b. Die Leerschritt-Taste drücken, um das nächste Byte anzuzeigen. Um den Wert zu ändern, einfach den neuen Wert gemäß Abschnitt (2a) eintragen. Wird eine 8-Byte-Grenze überschritten, beginnt der Debugger eine neue Anzeigenzeile und zeigt die Adresse am Anfang der Zeile an.

   c. Zur Änderung eines vorhergehenden Bytes, einen Bindestrich (-) eintasten, um zur vorhergehenden Position zurückzukehren. Eine neue Zeile beginnt mit der Adresse des Bytes, und sein Wert wird angezeigt.

d. Zeilenschaltung an einer beliebigen Byteposition drücken, um den E-Befehl zu beenden.

VORSICHT:
Der E-Befehl sollte nicht benutzt werden, um den Videospeicher zu ändern, da dieser wortorientiert ist.

Beispiel

Angenommen, folgender Befehl wird eingetragen:

```
ECS:100
```

Der Debugger zeigt folgendes an:

```
04BA:0100  EB._
```

Um diesen Wert auf 41 abzuändern, 41 wie folgt eintragen:

```
04BA:0100  EB.41_
```

Um die nachfolgenden Bytes zu durchgehen, die Leerschritt-Taste drücken. Folgende Zeile erscheint:

```
04BA:0100  EB.41    10.    00.    BC._
```

Um BC auf 42 abzuändern, 42 wie folgt eintragen:

```
04BA:0100  EB.41    10.    00.    BC.42_
```

Um die 10 auf 6F abzuändern, den Bindestrich so oft eintragen, wie es zur Rückkehr zu Byte 0101 (Wert 10) erforderlich ist, dann 10 mit 6F wie folgt ersetzen:

```
04BA:0100  EB.41    10.   00.   BC.42-
04BA:0102  00.-
04BA:0101  10.6F_
```

Zeilenschaltung drücken, um den E-Befehl zu beenden und zur Debugger-Befehlsstufe zurückzukehren.

4.6.5 F-Befehl (Füllen)

Funktion

Dieser Befehl füllt die Adressen im <Bereich> mit den Werten in der <Liste>.

Format

F<Bereich> <Liste>

<Bereich>	entweder zwei <Adressen> oder eine <Adresse>, ein L und ein <Wert> (<Adresse> L <Wert>). Eine zweite <Adresse> muß höher als die erste und darf nur ein Offset sein.
<Liste>	eine Reihe von <Byte>-Werten oder Zeichen, die in Anführungszeichen eingeschlossen sind.

Bemerkungen

1. Der F-Befehl gleicht insofern mehrfachen E-Befehlen, da jeweils mehr als eine Adresse geändert werden kann. Der F-Befehl bricht bei einem Fehler nicht wie der E-Befehl ab. Falls eine Speicherstelle im <Bereich> nicht zulässig (d. h. beschädigt oder nicht vorhanden) ist, können die übrigen Teile des <Bereichs> unrichtige Daten erhalten.

2. Falls der <Bereich> mehr Bytes enthält als Werte in der <Liste> ent- halten sind, benutzt der Befehl die <Liste> wiederholt, bis alle Bytes im <Bereich> gefüllt sind. Enthält die <Liste> mehr Werte als Bytes im <Bereich> vorhanden sind, ignoriert der Befehl die überflüssigen Werte in der <Liste>.

VORSICHT:
Der F-Befehl sollte nicht zum Ändern des Videospeichers benutzt werden, da dieser wortorientiert ist.

Beispiel

Angenommen, folgender Befehl wird eingetragen:

```
F04BA:100 L 100 42 45 52 54 41
```

Der Debugger füllt die Speicherstellen 04BA:100 bis 04BA:200 mit den angegebenen Bytes. Die fünf Werte werden wiederholt, bis alle 100H Bytes gefüllt sind.

4.6.6 G-Befehl (Go)

Funktion

Dieser Befehl führt das Programm aus, das sich gegenwärtig im Speicher befindet. Nach Wahl kann der Ablauf an einem Programmstop unterbrochen und Register, Kennzeichen und decodierte Instruktion angezeigt werden.

Format

G[=<Adresse>] [<Adresse>...]

<Adresse>	entweder eine alphabetische Segmentregister-Bezeichnung oder eine 1- bis 4-stellige Segmentadresse, der ein Doppelpunkt und ein Offset-Wert folgen. Das Standardsegment ist CS.

Bemerkungen

1. Wird der G-Befehl ohne Parameter eingetragen, führt das Programm so wie bei Ablauf außerhalb des Debuggers aus.

2. Wird die =<Adresse> bestimmt, beginnt der Ablauf an dieser Adresse. Wird die Segmentbezeichnung bei der =<Adresse> ausgelassen, setzt der Befehl nur den Instruktionsanzeiger. Ist die Segmentbezeichnung in der =<Adresse> ein- geschlossen, setzt der Befehl sowohl das CS-Segment als auch den Instruk- tionsanzeiger. Das Gleichheitszeichen (=) ist erforderlich, damit der Debugger die Anfangs-<Adresse> von den Programmstop-<Adresse>n unterscheiden kann.

3. Ist die Programmstop-<Adresse>n-Liste festgelegt, wird der Ablauf an der Programmstop-<Adresse> unterbrochen, die zuerst auftritt, ungeachtet der Programmverzweigung. Erreicht der Programmablauf einen Programmstop, erscheinen Register, Kennzeichen und decodierte Instruktion für die nächste Instruktion. (Das Ergebnis ist dasselbe wie bei Eintrag des R-Befehls für die Programmstop-Adresse.)

4. Es können bis zu zehn Programmstops gesetzt werden. Da Programmstops instruktionsorientiert sind, können sie nur an die Adresse des ersten Bytes einer Instruktion gesetzt werden. Werden mehr als 10 Programmstops festgelegt, zeigt der Debugger die BP- (Zu viele Programmstops) Fehlermeldung an.

5. Wenn ein ausführendes Programm auf einen Programmstop stößt, führt das Programm weiter aus, wenn der G-Befehl nochmals eingetragen wird, und zwar so als wäre der Dateiname auf DOS Befehlsstufe eingegeben worden. Der einzige Unterschied liegt darin, daß der Programmablauf an der Instruktion nach dem Programmstop und nicht an der gewöhnlichen Anfangsadresse wieder aufgenommen wird.

6. Der Anwender-Stapelanzeiger muß zulässig sein und der Stapelspeicher für diesen Befehl über sechs Bytes verfügen. Der G-Befehl benutzt eine IRET-Instruktion (Rückkehr nach Unterbrechung), um einen Sprung zum auszutestenden Programm zu veranlassen. Der Befehl setzt den Anwender-Stapelanzeiger und schiebt die Anwender-Kennzeichen, das Codesegmentregister und den Instruktionsanzeiger auf den Anwender-Stapel.

VORSICHT:
Ein nicht zulässiger oder zu kleiner Anwender-Stapelspeicher kann Ausfall des Betriebssystems verursachen.

7. Der G-Befehl setzt einen 1-Byte Unterbrechungs-Op-Code (03H) an der/den angegebenen Programmstop-Adresse(n). Der Debugger reserviert einen Bereich, um die ursprünglichen Instruktionen dieser Adressen zu speichern. Wenn das ausführende Programm auf eine Instruktion mit Programmstopcode stößt, verwandelt der Debugger alle Programmstop-Adressen in die ursprünglichen Instruktionen zurück. Wird die Ausführung nicht an einem der Programmstops unterbrochen, werden die Unterbrechungscodes nicht mit den ursprünglichen Instruktionen ersetzt.

Beispiel

Angenommen, folgender Befehl wird eingetragen:

```
GCS:7550
```

Das sich gegenwärtig im Speicher befindende Programm führt auf Adresse 7550H im CS-Segment aus. Nach Anzeige der Register und Kennzeichen erscheint der Debugger-Dialoghinweis.

4.6.7 H-Befehl (Hex Arithmetik)

Funktion

Dieser Befehl führt hexadezimale arithmetische Operationen an zwei Parametern durch.

Format

H<Adresse> <Adresse>

<Adresse>	entweder eine alphabetische Segmentregister-bezeichnung oder eine 1 bis 4-stellige Segment-adresse, der ein Doppelpunkt und ein Offset-Wert folgen. Das Standardsegment ist DS.

Bemerkungen

Der Debugger addiert die zwei Parameterwerte, subtrahiert dann den zweiten Parameter vom ersten. Die Ergebnisse, d. h. zuerst die Summe und dann die Differenz, erscheinen auf einer Zeile.

Beispiel

Angenommen, folgender Befehl wird eingetragen:

```
H10A 19F
```

Der Debugger führt die Berechnungen durch und zeigt dann die Ergebnisse an:

```
02A9 0095
```

4.6.8 I-Befehl (Eingabe)

Funktion

Dieser Befehl gibt ein Byte von der durch <Wert> bestimmten Anschlußstelle ein und zeigt es an.

Format

I<Wert>

<Wert>	eine 1- bis 4-stellige Hexadezimalziffer.

Bemerkungen

Eine 16-Bit-Anschlußstellenadresse ist erlaubt.

Beispiel

Angenommen, folgender Befehl wird eingetragen:

```
I2F8
```

Angenommen, das Byte an der Anschlußstelle enthält 42H. Der Debugger gibt das Byte ein und zeigt folgenden Wert an:

```
42
```

4.6.9 L-Befehl (Laden)

Funktion

Dieser Befehl lädt eine Datei in den Speicher.

Format

L[<Adresse> [<Laufwerk> <Sektor> <Wert>]]

<Adresse>	entweder eine alphabetische Segmentregister-Bezeichnung oder eine 1- bis 4-stellige Segmentadresse, der ein Doppelpunkt und ein Offset-Wert folgen. Das Standardsegment ist CS.
<Laufwerk>	ein 1-stelliger Hexadezimalwert.
<Sektor>	ein 1- bis 4-stelliger Hexadezimalwert, der den physischen Sektor auf der Disk bezeichnet, an dem der Ladevorgang beginnt.
<Wert>	eine 1- bis 4-stellige Hexadezimalzahl.

Bemerkungen

1. Die Datei muß entweder im Debugger-Aufruf oder mit dem zuletzt erteilten N-Befehl benannt worden sein. Sowohl der Aufruf als auch die N-Befehle formatieren einen Dateinamen ordnungsgemäß in normalem Format eines Dateisteuerblocks bei CS:5C (siehe Abschnitt A.3.5, Programm-Initialisierung, und Abschnitt A.4.1, Dateisteuerblock-Definition).

2. Wird eine Datei oder eine Teildatei angegeben, die für den verfügbaren Speicher zu groß ist, bricht der L-Befehl ab, und der Debugger-Dialoghinweis kehrt zum Bildschirm zurück.

3. Wird der L-Befehl ohne Parameter eingetragen, beginnt der Debugger die Datei bei Adresse CS:100 in den Speicher zu laden und legt BX:CX gemäß der Anzahl der geladenen Bytes fest. Wird der L-Befehl mit einem <Adresse>n-Parameter eingetragen, beginnt der Ladevorgang an der angegebenen Speicheradresse, und BX:CX wird gemäß der Anzahl der geladenen Bytes festgelegt. Wird L mit allen Parametern eingetragen, lädt der Debugger keine Datei, sondern absolute Disksektoren. Der Debugger nimmt <Wert>-Sektoren von dem bezeichneten <Laufwerk>. Die Laufwerkbezeichnung ist numerisch (0=A:, 1=B:, 2=C:). Der Debugger beginnt den Ladevorgang mit dem ersten angegebenen <Sektor> und fährt fort, bis die in <Wert> angegebene Sektorenzahl geladen ist.

4. Hat die Datei eine .EXE Erweiterung, wird sie zur Ladeadresse verschoben, die im Vorsatz der .EXE Datei angegeben ist. Bei .EXE Dateien ignoriert der Debugger stets den <Adresse>-Parameter des L-Befehls. Der Vorsatz wird von der .EXE Datei entfernt, bevor sie in den Speicher geladen wird. Die Größe einer .EXE Datei auf Disk weicht daher von ihrer Größe im Speicher ab.

5. Falls die mit dem N-Befehl benannte oder bei Aufruf bezeichnete Datei eine .HEX Datei ist, verursacht der L-Befehl ohne Parameter Beginn des Dateiladevorgangs an der Adresse, die in der .HEX Datei angegeben ist. Schließt der L-Befehl den <Adresse>-Parameter ein, fügt der Debugger die im L-Befehl angegebene Adresse zur Adresse in der .HEX Datei hinzu, um die Anfangsadresse zum Laden der Datei zu bestimmen.

Beispiele

Angenommen, folgender Befehl wird eingetragen:

```
A:DEBUG
>NFILE.COM
```

Zum Laden von FILE.COM nun folgendes eingeben:

```
L
```

Der Debugger lädt die Datei und zeigt den Debugger-Dialoghinweis an.

Sollen Teile einer Disk geladen werden, folgendes eingeben:

```
L04ba:100 2 0F 6D
```

Der Debugger lädt bei Speicher-Anfangsadresse 04BA:0100 109 (6D hex) Sektoren, die bei dem absoluten Sektor Nummer 15 beginnen. Sind die Sektoren geladen, erscheint wieder der Debugger-Dialoghinweis.

4.6.10 M-Befehl (Verschieben)

Funktion

Dieser Befehl verschiebt den in <Bereich> bezeichneten Speicherblock an die Stelle, die an der angegebenen <Adresse> beginnt.

Format

M<Bereich> <Adresse>

<Bereich>	entweder zwei <Adressen> oder eine <Adresse>, ein L und ein <Wert> (<Adresse> L <Wert>). Eine zweite <Adresse> muß höher als die erste und darf nur ein Offset sein.
<Adresse>	entweder eine alphabetische Segmentregister-bezeichnung oder eine 1- bis 4-stellige Segment- adresse, der ein Doppelpunkt und ein Offset-Wert folgen. Das Standardsegment ist DS.

Bemerkungen

Der M-Befehl führt ohne Verlust von Daten überschneidende Verschiebvorgänge aus (falls einige Ausgangs- und Zielbereiche gleiche Adressen tragen). Zuerst werden Daten verschoben, die überschrieben werden könnten. Bei Verschiebvorgängen von höheren auf niedrigere Adressen werden die Daten, die an der niedrigsten Adresse des Blocks beginnen, zuerst verschoben. Bei Verschiebvorgängen von niedrigeren auf höhere Adressen werden die Daten, die an der höchsten Adresse des Blocks beginnen, zuerst verschoben.

VORSICHT:
Der M-Befehl sollte nicht zur Änderung des Videospeichers benutzt werden, da dieser wortorientiert ist.

Beispiel

Angenommen, folgendes wird eingetragen:

```
MCS:100 110 CS:500
```

Der Debugger verschiebt zuerst Adresse CS:110 auf Adresse CS:510, dann CS:10F auf Adresse CS:50F usw., bis CS:100 auf Adresse CS:500 verschoben wird. Um das Ergebnis des Verschiebvorgangs zu überprüfen, den D-Befehl mit der gleichen <Adresse> wie für den M-Befehl eingetragen.

4.6.11 N-Befehl (Name)

Funktion

Dieser Befehl führt zwei verschiedene Funktionen aus: N kann benutzt werden, um eine Dateispezifikation für einen späteren L- oder W-Befehl zuzuweisen. N kann ebenfalls verwendet werden, um einer auszutestenden Datei Dateispezifikationen als Parameter zu geben.

Format

N<Filespez.>[/S...] [<Filespez.>[S...]...]

<Filespez.>	ein 1- bis 8-Zeichen langer Dateiname, dem wahlweise eine Laufwerkbezeichnung vorausgeht und wahlweise ein Punkt und eine 1- bis 3-Zeichen-Erweiterung folgt.
</S>	ein DOS Schalter, der für die angegebene Datei gesetzt wird.

Bemerkungen

1. Wird der Debugger aufgerufen, ohne daß eine auszutestende Datei benannt wird, muß der N-Befehl eingetragen werden, bevor der Debugger eine Datei laden kann.

2. Der N-Befehl kann sich auf vier verschiedene Speicherbereiche auswirken:

DS:5C FCB für Datei 1
DS:6C FCB für Datei 2
DS:80 Zeichenanzahl
DS:81 Alle eingegebenen Zeichen

Eine Beschreibung dieser Bereiche befindet sich in Abschnitt A.3.5.

3. Der Debugger setzt einen Dateisteuerblock (FCB) für die erste Dateispezifikation im N-Befehl bei DS:5C (siehe Abschnitt A.4.1). Bei Angabe einer zweiten Dateispezifikation setzt der Debugger einen FCB, der bei DS:6C beginnt. Die Stelle DS:80 enthält die Anzahl der Zeichen, die in den N-Befehl eingegeben werden (ausschließlich des ersten Zeichens N). Die Zeichen im N-Befehl (ausschließlich des Buchstabens N) beginnen bei DS:81. Diese Zeichen können Schalter und Begrenzungssymbole enthalten, die in jedem auf DOS Befehlsstufe eingetasteten Befehl gültig sind.

4. Die Funktionen, d. h. Zuweisen einer Dateispezifikation für L- und W-Befehle und Angabe von Dateispezifikations-Parametern, überschneiden sich. Folgender Debugger-Befehlsatz veranschaulicht dies:

```
>NFILE1.EXE
>L
>G
```

Die Doppelfunktion des N-Befehls verursacht folgendes:

a. N weist den erweiterten Dateinamen FILE1.EXE einem späteren Lade- oder Schreibbefehl zu.

b. N weist ebenfalls den erweiterten Dateinamen FILE1.EXE als ersten Dateispezifikations-Parameter zu, der von einem später auszutestenden Programm benutzt wird (siehe Absatz d.).

c. L lädt FILE1.EXE in den Speicher.

d. G führt FILE1.EXE aus. FILE1.EXE operiert auch als einziger Dateispezifikations-Parameter (d. h. FILE1.EXE führt aus wie bei Eintrag von FILE1.EXE auf DOS Befehlsstufe).

Folgender Befehlssatz ist zweckmäßiger:

```
>NFILE1.EXE
>L
>NFILE2.DAT FILE3.DAT
>G
```

In diesem Beispiel setzt N FILE1.EXE als Dateispezifikation für den folgenden L-Befehl. (Der DEBUG-Befehl kann für diese Funktion ebenfalls benutzt werden.) Der L-Befehl lädt FILE1.EXE in den Speicher. Der N-Befehl bezeichnet dann die von FILE1.EXE benutzten Parameter. Wenn der G-Befehl ausführt, läuft FILE1.EXE schließlich, als wäre FILE1 FILE2.DAT FILE3.DAT auf DOS Befehlsstufe eingetastet worden.

Wird zu diesem Zeitpunkt ein W-Befehl eingetragen, wird die Datei, die ausgetestet wird (FILE1.EXE), mit dem Namen FILE2.DAT gespeichert. Um solche unerwünschten Ergebnisse zu vermeiden, sollte ein N-Befehl immer vor einem L- oder W-Befehl ausgeführt werden.

Beispiel

Ein typischer Gebrauch des N-Befehls ist:

```
DEBUG PROG.COM
NPARAM1 PARAM2/C
G
```

In diesem Beispiel führt der G-Befehl die Datei im Speicher so wie bei Eintrag folgender DOS Befehlszeile aus:

```
PROG PARAM1 PARAM2/C
```

Prüfung und Fehlerbehebung stellen folglich ein normales Laufzeitumfeld für PROG.COM dar.

Der Debugger lädt PROG.COM bei DS:100. Diese Dateispezifikation wird in den FCB bei DS:5C gesetzt. Dann formatiert N FCBs für PARAM1 und PARAM2/C bei DS:5C bzw. DS:6C und überschreibt den FCB für PROG.COM.

4.6.12 O-Befehl (Ausgabe)

Funktion

Dieser Befehl sendet das bezeichnete <Byte> an die in <Wert> bestimmte Ausgabeanschlußstelle.

Format

O<Wert> <Byte>

<Wert>	eine 1- bis 4-stellige Hexadezimalzahl.
<Byte>	ein 2-stelliger in die Ausgabeanschlußstelle zu setzender Hexadezimalwert.

Bemerkungen

1. Eine 16-Bit Anschlußstellenadresse ist erlaubt.

2. Der O-Befehl ist nur bei byteorientierten und nicht bei wortorientierten Anschlußstellen, wie z. B. für die Videoschaltkarte, möglich.

Beispiel

Angenommen, folgendes wird eingegeben:

```
O2F8 4F
```

Der Debugger gibt den Bytewert 4F zur Ausgabeanschlußstelle 2F8 aus.

4.6.13 Q-Befehl (Abbrechen)

Funktion

Dieser Befehl beendet das Debugger-Programm.

Format

Q

Bemerkungen

Der Q-Befehl erhält keine Parameter und verläßt den Debugger, ohne die gerade bearbeitete Datei zu speichern. Der Q-Befehl kehrt dann zu dem Programm zurück, von dem der Debugger aufgerufen wurde.

Beispiel

Um das Debugger-Programm zu beenden, folgendes eingeben und dann Zeilenschaltung drücken:

Q

Das Debugger-Programm bricht ab und führt die Steuerung zu der Stufe zurück, auf welcher der Debugger aufgerufen wurde.

4.6.14 R-Befehl (Register)

Funktion

Dieser Befehl hat drei Funktionen: (1) R zeigt den Inhalt eines CPU-Registers an und ermöglicht Änderung des Inhalts. (2) R zeigt die acht Kennzeichen-Einstellungen an und ermöglicht, diese zu ändern. (3) R zeigt den Inhalt aller Register, Kennzeichen-Einstellungen und der nächsten auszuführenden Instruktion in decodiertem Format an.

Format

R[<Registername>]

<Registername>	entweder ein 2-Zeichen Registername oder das Zeichen F für Kennzeichen.

Bemerkungen

1. Wird kein <Registername> eingetragen, erstellt der R-Befehl einen Auszug des Sicherstellungsbereich für Register und zeigt den Inhalt aller Register und Kennzeichen an.

Bei Start enthalten die Segmentregister den unteren Teil des freien Speichers, der Instruktionsanzeiger enthält 0100H, der Stapelanzeiger enthält FFFEH, alle Kennzeichen gelöscht, und die übrigen Register enthalten Null.

2. Wird ein Registername eingetragen, zeigt der Debugger den 16-Bit-Wert dieses Registers hexadezimal an; dann erscheint ein Doppelpunkt als Dialoghinweis. Einen <Wert> eingeben, um das Register zu ändern, oder Zeilenschaltung drücken, falls keine Änderung vorgenommen werden soll.

3. Die einzigen zulässigen <Registernamen> lauten wie folgt (IP und PC beziehen sich auf den Instruktionsanzeiger):

AX	BP	SS
BX	SI	CS
CX	DI	IP
DX	DS	PC
SP	ES	F

Alle anderen Eintragungen für den <Registernamen> verursachen eine BR- Fehlermeldung (Falsches Register).

4. Wird F als <Registername> eingetragen, zeigt der Debugger acht alphabe- tische 2-Zeichencodes an. Um das Kennzeichen zu verändern, den entgegenge- setzten 2-Buchstabencode eintragen. Kennzeichen sind entweder gesetzt oder gelöscht.

Tabelle 4-2 enthält die Kennzeichen einschließlich der gesetzten und gelöschten Codes.

Tabelle 4-2. Kennzeichen und Kennzeichencodes

Kennzeichenname	Gesetzt	Gelöscht
Überlauf	OV	NV
Richtung	DN (Dekrement)	UP (Inkrement)
Unterbrechung	EI (Freigegeben)	DI (Gesperrt)
Vorzeichen	NG (Negativ)	PL (Plus)
Null	ZR	NZ
Zusatz-Übertrag	AC	NA
Parität	PE (Gerade)	PO (Ungerade)
Übertrag	CY	NC

5. Wird der RF-Befehl eingetragen, zeigt der Debugger die Kennzeichen in einer Reihe am Zeilenanfang, und zwar in der in Tabelle 4-2 dargestellten Reihenfolge an. Am Ende der Kennzeichenliste zeigt der Debugger einen Bindestrich (-) und dann den Debugger-Dialoghinweis (>) an. Neue Kennzeichenwerte können als alphabetische Paare in beliebiger Reihenfolge eingegeben werden. Es ist nicht notwendig, die Kennzeichen-Eintragungen durch Zwischenräume zu trennen. Um den R-Befehl zu verlassen, Zeilenschaltung drücken. Kennzeichen, für die keine neuen Werte eingetragen wurden, bleiben unverändert.

6. Falls mehr als ein Wert für ein Kennzeichen eingetragen wird, antwortet der Debugger mit einer DF-Fehlermeldung (Doppeltes Kennzeichen). Wird ein anderer als in Tabelle 4-2 enthaltener Kennzeichencode eingetragen, zeigt der Debugger eine BF-Fehlermeldung (Falsches Kennzeichen) an. In beiden Fällen werden die Kennzeichen bis zu dem Fehler in der Aufstellung geändert. Kennzeichen an und nach dem Fehler werden jedoch nicht geändert.

Beispiele

Angenommen, folgendes wird eingetragen:

```
R
```

Der Debugger zeigt alle Register, Kennzeichen und die decodierte Instruktion für die jeweilige Stelle an. Ist die Stelle z. B. CS:11A, zeigt der Debugger folgendes an:

```
AX=0E00 BX=00FF CX=0007 DX-01FF SP=039D BP=0000
SI=005C DI-0000 DS=04BA ES=04BA SS=04BA CS=04BA
IP=011A NV UP DI NG NZ AC PE NC
04BA:011A CD21    INT   21
```

Angenommen, folgendes wird eingetragen:

```
RF
```

Der Debugger zeigt folgende Kennzeichen an:

```
NV UP DI NG NZ AC PE NC -
```

Eine beliebige gültige Kennzeichenfolge mit oder ohne Zwischenräume eintragen. Zum Beispiel:

```
NV UP DI NG NZ AC PE NC - PLEICY
```

Der Debugger antwortet nur mit dem Debugger-Dialoghinweis. Um die Änderungen zu prüfen, entweder den R- oder RF-Befehl eingeben:

```
RF
NV UP EI PL NZ AC PE CY -
```

Andere Kennzeichenwerte eintragen oder Zeilenschaltung drücken, falls die Kennzeichen unverändert bleiben sollen.

4.6.15 S-Befehl (Suchen)

Funktion

Dieser Befehl sucht den bezeichneten <Bereich> nach der angegebenen <Liste> ab.

Format

S<Bereich> <Liste>

<Bereich>	entweder zwei <Adressen> oder eine <Adresse>, ein L und ein <Wert> (<Adresse> L <Wert>). Eine zweite <Adresse> muß höher als die erste und darf nur ein Offset sein.
<Liste>	eine Reihe von <Byte>-Werten oder Zeichen, die in Anführungszeichen eingeschlossen sind.

Bemerkungen

1. Die Liste kann eine oder mehrere Werte enthalten, die jeweils durch einen Zwischenraum oder ein Komma getrennt sind. Enthält die Liste mehr als einen Wert, zeigt der Debugger die Anfangsadresse jeder übereinstimmenden Bytekette im Bereich an. Enthält die Liste nur einen Wert, zeigt der Debugger die Adressen aller Bytes im <Bereich> an, die diesen Wert enthalten.

2. Erscheint der Debugger-Dialoghinweis ohne weitere Anzeige, wurde während des Suchvorgangs für die Liste im Bereich keine Übereinstimmung gefunden.

Beispiel

Angenommen, folgendes wird eingetragen:

```
SCS:100 110 41
```

Der Debugger erwidert ggf.:

```
04BA:0104
04BA:010D
>_
```

4.6.16 T-Befehl (Ablaufverfolgung)

Funktion

Dieser Befehl führt eine oder mehrere Instruktionen aus und zeigt den Inhalt aller Register und Kennzeichen nach Ausführung und die nächste Instruktion in decodiertem Format an.

Format

T[=<Adresse>] [<Wert>]

<Adresse>	entweder eine alphabetische Segmentregister-Bezeichnung oder eine 1- bis 4-stellige Segmentadresse, der ein Doppelpunkt und ein Offset-Wert folgen. Das Standardsegment ist CS.
<Wert>	eine 1- bis 4-stellige Hexadezimalziffer.

Bemerkungen

1. Wird eine wahlweise =<Adresse> eingetragen, findet die Ablaufverfolgung an der angegebenen Adresse statt. Ist die Segmentbezeichnung in der Adresse eingeschlossen, werden sowohl das CS-Register als auch der Instruktions- anzeiger bezeichnet. Ist die Segmentbezeichnung in der Adresse ausgelassen, wird nur der Instruktionsanzeiger angegeben.

2. Der wahlweise <Wert> veranlaßt den Debugger, die Anzahl der in <Wert> angegebenen Instruktionen auszuführen und deren Ablauf zu verfolgen.

3. Der Prozessor 8086 läßt bei Instruktionen, die Segmentregister abändern, keine Einzelschritte zu. Ein einzelner Aufruf des T-Befehls kann daher z. B. Ausführung von zwei Instruktionen auslösen, MOV SS, AX und MOV SP, DX. Der T-Befehl zeigt dann den Inhalt der Register und Kennzeichen nach Ausführung der zweiten und nicht der ersten Instruktion an. Auch wird nicht die zweite, sondern die darauffolgende Instruktion angezeigt.

Beispiele

Angenommen, folgendes wird eingegeben:

```
T
```

Der Befehl veranlaßt die Ausführung einer Instruktion (oder zweier Instruktionen, siehe letzte Bemerkung). Dann erscheint eine Anzeige der Register, Kennzeichen und der decodierten Instruktion für die Adresse, die nun durch den Instruktionsanzeiger angesprochen wird. Angenommen, die Adresse der nächsten Instruktion ist 04BA:011A; der Debugger antwortet ggf. mit folgender Anzeige:

```
AX=0E00 BX=00FF CX=0007 DX-01FF SP=039D BP=0000
SI=005C DI-0000 DS=04BA ES=04BA SS=04BA CS=04BA
IP=011A NV UP DI NG NZ AC PE NC
04BA:011A   CD21      INT     21
```

Nun folgendes eingeben:

```
T=011A 10
```

Der Debugger beginnt bei 011A im laufenden Segment zehn Instruktionen auszuführen und zeigt dann alle Register und Kennzeichen für jede Instruktion während der Ausführung an. Die Anzeige rollt ab, bis die letzte Instruktion ausgeführt hat. Dann hält die Anzeige an, und die Register und Kennzeichen für die letzten ausgeführten Instruktionen sind sichtbar. Mit KRTL + S kann die Anzeige an irgendeinem Punkt angehalten werden, wenn die Register und Kennzeichen für eine Instruktion durchgesehen werden sollen. Mit KRTL + Q wird die Anzeige wieder aufgenommen.

4.6.17 U-Befehl (Disassemblieren)

Funktion

Dieser Befehl disassembliert Bytes und zeigt die entsprechenden Quelldaten zusammen mit Adressen und hexadezimalen Bytewerten an.

Formate

1. U[<Adresse>]

   <Adresse> — entweder eine alphabetische Segmentregister-Bezeichnung oder eine 1- bis 4-stellige Segmentadresse, der ein Doppelpunkt und ein Offset-Wert folgen. Das Standardsegment ist CS.

2. U<Bereich>

   <Bereich> — entweder zwei <Adressen> oder eine <Adresse>, ein L und ein <Wert> (<Adresse> L <Wert>). Eine zweite <Adresse> muß höher als die erste und darf nur ein Offset sein.

Bemerkungen

1. Die Anzeige disassemblierter Codes ähnelt einer Liste für eine assemblierte Quelldatei.

2. Wird der U-Befehl ohne Parameter eingetragen, beginnt der Debugger bei CS:IP 20H Bytes zu disassemblieren, um die entsprechenden Instruktionen zu zeigen. Bei Eingabe des U-Befehls mit <Bereich>-Parameter disassembliert der Debugger alle Bytes im Bereich bis zu 20 Bytes. Falls sich im <Bereich> weniger als 20H Bytes befinden, zeigt der Debugger nur diese Bytes an.

3. Wird der U-Befehl nur mit dem Parameter <Adresse> eingetragen, beginnt der Debugger an der bezeichneten Adresse die Byte-Standardanzahl zu dis- assemblieren. Wird der U-Befehl mit den Parametern <Adresse> L <Wert> einge- tragen, beginnt der Debugger an der angegebenen Adresse alle Bytes auf den in <Wert> angegebenen Zeilen. Bei Eintrag des U-Befehls mit den Parametern <Adresse> L <Wert> wird die Standardbegrenzung (20H Bytes) aufgehoben.

4. Wurde der E-Befehl zur Änderung der Bytes in einigen Adressen benutzt, verändert der Disassembler die entsprechenden Quellinstruktionen. Der U-Befehl kann eingetragen werden, um die geänderten Stellen anzuzeigen und der disassemblierte Code benutzen werden, um die Quelldatei aufzubereiten.

5. Bei dem U-Befehl weicht die Syntax der Assembler-Sprachanweisungen etwas von der Standard-Assemblersprache 8086/8088 ab. Abschnitt 4.6.1, Bemerkung 4, geht auf die Unterschiede ein.

Beispiele

Angenommen, folgendes wird eingetragen:

```
U04BA:100
```

Der Debugger beginnt bei Adresse 04BA:0100 20 Bytes wie folgt zu disassemblieren:

```
04BA:0100    BA7406     MOV    DX, 0674
04BA:0103    OE         PUSH   CS
04BA:0104    1F         POP    DS
04BA:0105    E8F802     CALL   0420
04BA:0108    EB12       JMP    013C
04BA:010A    CD21       INT    21
04BA:010C    72D5       JB     0103
04BA:010E    50         PUSH   AX
04BA:010F    B44D       MOV    AH,4D
04BA:0111    CD21       INT    21
04BA:0113    2E         CS:
04BA:0114    A22F07     MOV    [072F],AL
04BA:0117    80FC01     CMP    AH,01
04BA:011A    7403       JZ     013F
04BA:011C    E98000     JMP    01BF
04BA:011F    58         POP    AX
```

Bei Eingabe von:

```
u04ba:0100 0108
```

erscheint:

```
04BA:0100    BA7406     MOV    DX, 0674
04BA:0103    OE         PUSH   CS
04BA:0104    1F         POP    DS
04BA:0105    E8F802     CALL   0420
04BA:0108    EB12       JMP    013C
```

Bei Eingabe von

```
u04ba:100 120
```

erscheint die Anzeige jedoch genau wie oben mit dem Befehl UCS:100. Wird jedoch der Befehl

```
UCS:100 L 20
```

eingetragen, werden alle Bytes in den 32 Instruktionszeilen von Adresse CS:100 bis Adresse CS:120 disassembliert und in beiden Anzeigeformaten dargestellt.

4.6.18 W-Befehl (Schreiben)

Funktion

Dieser Befehl schreibt die Datei, die ausgetestet wird, in eine Diskdatei.

Format

W[<Adresse> [<Laufwerk> <Sektor> <Wert>]]

<Adresse>	entweder eine alphabetische Segmentregister-Bezeichnung oder eine 1- bis 4-stellige Segmentadresse, der ein Doppelpunkt und ein Offset-Wert folgen. Das Standardsegment ist CS.
<Laufwerk>	ein 1-stelliger Hexadezimalwert.
<Sektor>	ein 1- bis 4-stelliger Hexadezimalwert, der zur Bestimmung des physischen Sektors auf der Disk, an dem der Schreibvorgang beginnt, benutzt wird.
<Wert>	eine 1- bis 4-stellige Hexadezimalzahl.

Bemerkungen

1. Die Datei muß mit dem Debugger-Aufruf oder N-Befehl benannt werden. Beide Methoden formatieren einen Dateinamen ordnungsgemäß im Dateisteuerungsblock bei DS:5C (siehe Abschnitt A.4.1, Dateisteuerblock-Definition). Falls ein späterer N-Befehl den ursprünglichen Dateinamen ändert, einen weiteren N-Befehl eingeben, damit der Dateiname seine ursprüngliche Form zurückerhält, bevor die ursprüngliche Datei geschrieben wird.

2. Der Debugger schreibt die ausgetestete Datei über die ursprüngliche Datei oder in eine neue Datei, falls die im Dateisteuerblock bei CS:5C benannte Datei nicht vorher bestand. Der Debugger schreibt .EXE Dateien jedoch nicht zurück. Wird eine Datei vor dem Schreiben auf Disk geladen und modifiziert, setzt der Debugger das richtige Datum und die korrekte Länge, um die modifizierte Datei zu speichern.

3. Werden keine Parameter eingetragen, definiert BX:CX die Anzahl der zu schreibenden Bytes, und die Anfangsadresse ist CS:100. Falls der G-, R- oder T-Befehl diese Werte geändert hat oder falls neue Werte benötigt werden, müssen sie vor Eintrag des W-Befehls ohne Parameter ausdrücklich neu gesetzt werden (dazu kann der R-Befehl benutzt werden).

4. Wird der W-Befehl mit Parametern eingetragen, beginnt der Schreibvorgang an der angegebenen Speicheradresse. W schreibt die Datei zum angegebenen Laufwerk. Die Laufwerkbezeichnung ist hier numerisch -- 0 für Laufwerk A, 1 für Laufwerk B, 2 für Laufwerk C usw. Der Debugger beginnt am angegebenen <Sektor> in die Datei zu schreiben und fährt fort, bis die in <Wert> angege- bene Sektorenanzahl geschrieben ist.

VORSICHT:
Bei Benutzung der <Sektor>- und <Wert>-Parameter ÄUSSERST vorsichtig sein, da ein Fehler die Datei an eine falsche Stelle auf der Disk schreiben und dort die Daten zerstören kann.

Beispiele

Angenomen, folgendes wird eingetragen:

```
W
```

Der Debugger schreibt die Datei auf Disk und zeigt dann nach Abschluß des Schreibvorgangs den Debugger-Dialoghinweis (>) an.

Angenommen, folgendes wird eingetragen:

```
WCS:100 1 37 2B
```

Der Debugger beginnt bei Adresse CS:100 den Inhalt des Speichers auf die Disk in Laufwerk B: zu schreiben. Die geschriebenen Daten beginnen an der physischen Sektornummer 37H der Disk und bestehen aus 2BH Sätzen. Nach Abschluß des Schreibvorgangs erscheint der Debugger-Dialoghinweis.

4.6.19 X-Befehl (E/A-Gerät ändern)

Funktion

Mit diesem Befehl wird E/A des Debuggers an ein anderes Gerät gesendet. Debugger-E/A geht normalerweise zum Monitor.

Format

X <Gerät>

<Gerät> ein reservierter Dateiname für ein Gerät

Bemerkungen

Der Debugger benutzt für seine Diagnostik-E/A keine Systemaufrufe, sondern ruft BIOS direkt auf. Da der Debugger BIOS für Diagnostik-E/A benutzt, kann der Debugger MSDOS selbst austesten. Durch Benutzung von BIOS läßt sich auch Debugger-E/A leicht auf ein anderes Gerät verschieben -- eine Eigenschaft, die zum Austesten von Grafik-Software von Nutzen sein kann.

Beispiel

Angenommen, folgendes wird eingetragen:

```
>X AUX
```

Der Debugger sendet E/A an das AUX-Gerät.

ANHANG A
PROZESSOR, BETRIEBSSYSTEM UND DATEIVERWALTUNG

A.1 EINFÜHRUNG

Dieser Anhang enthält einen technischen Überblick über den Prozessor Intel 8086 und das Microsoft Betriebssystems (DOS) bei Verwendung auf dem WANG PC. Die Angaben in diesem Anhang sind für Benutzer des WANG PC Assemblers, Debuggers oder Linkers und für Benutzer von kompilierten Sprachen von Nutzen. Der Prozessor 8086 ist der Zentralprozessor des WANG PCs.

A.2 ARCHITEKTUR-ÜBERBLICK

Dieser Abschnitt gibt einen Überblick über den Prozessor 8086; beschrieben werden Register, Speichersegmentierung, Status- und Steuerkennzeichen, Speicherorganisation und Adressengenerierung.

A.2.1 Register und Kennzeichen

Der Prozessor 8086 arbeitet mit zwölf 16-Bit-Registern. Acht davon sind allgemeine Register und vier Segmentregister. Ferner sind ein 16-Bit Instruktionsanzeiger (IP), sechs 1-Bit-Statuskennzeichen und drei Steuerkennzeichen vorhanden.

Allgemeine Register

Die allgemeinen Register fallen in zwei Gruppen: eine Gruppe von vier Datenregistern und eine Gruppe von vier Anzeige- und Indexregistern. Abbildung A-1 enthält Namen, Funktion und Gruppe der allgemeinen Register. Wie Abbildung A-1 zeigt, kann die obere und untere Hälfte der Datenregister getrennt adressiert werden. Jedes Datenregister kann also abwechselnd als 16-Bit-Register oder als zwei 8-Bit-Register operieren. Die anderen Register operieren immer als 16-Bit-Einheiten.

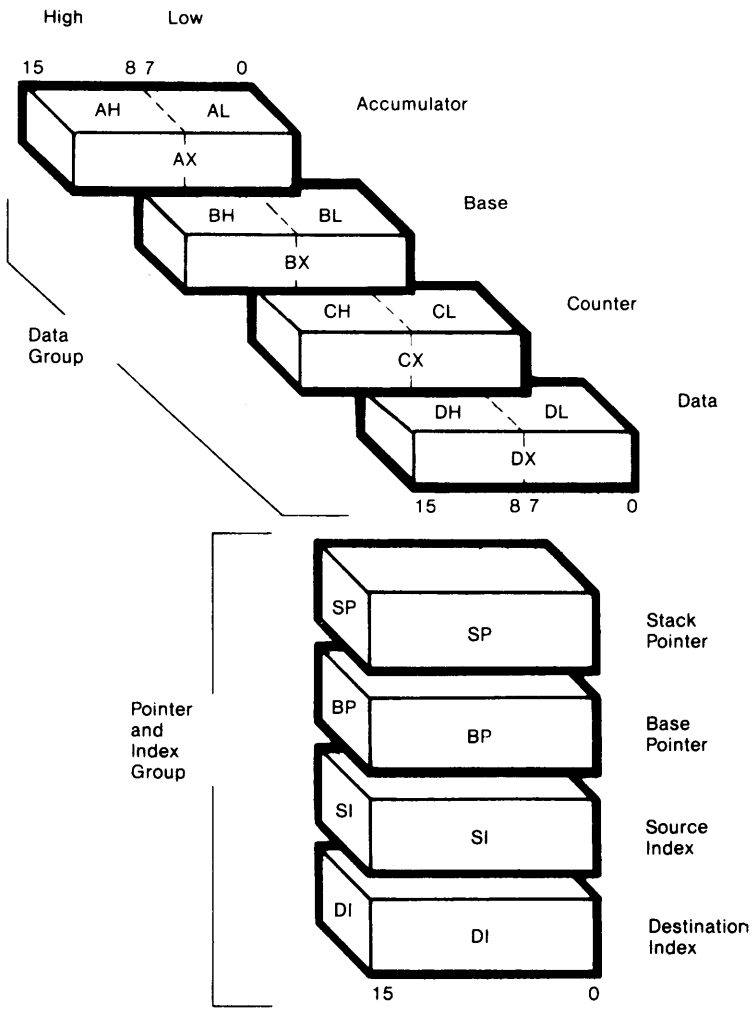


Abbildung A-1. Allgemeine Register

Segmentierung und Segmentregister

Der Prozessor 8086 hat einen physischen Adreßraum von einem Megabyte. Programme betrachten den Speicherraum von einem Megabyte als Sammlung logischer "Segmente". Ein Segment ist eine bis zu 64KB große Speichereinheit. Jedes Segment besteht aus zusammenhängenden Speicherstellen und ist eine unabhängige, getrennt adressierbare Einheit. Das System weist jedem Segment eine Basisadresse zu, die die Anfangsstelle im physischen Speicher darstellt. Alle Segmente beginnen auf den Paragraphengrenzen. Wie aus Abbildung A-2 zu ersehen ist, können die Segmente angrenzend, getrennt, teilweise überlagert oder vollständig überlagert sein. Mehr als ein logisches Segment kann die gleiche physische Speicherstelle enthalten.

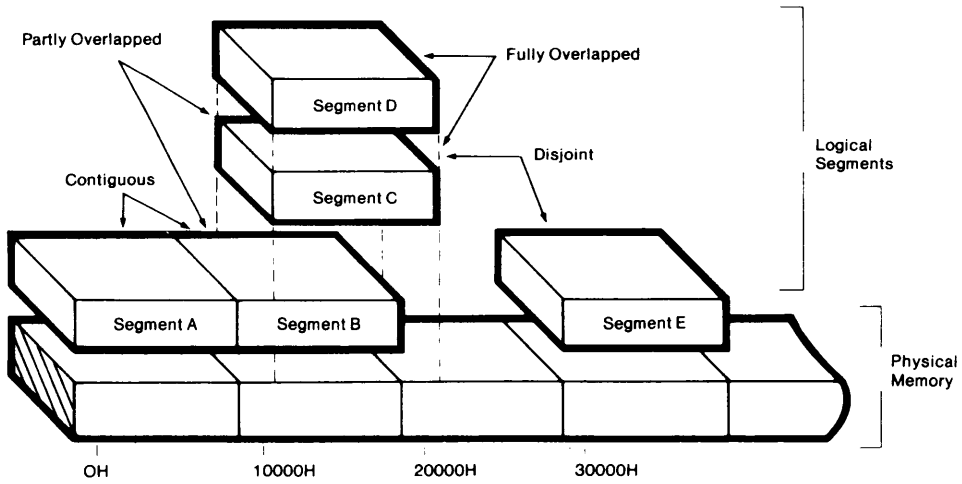


Abbildung A-2. Beziehung zwischen logischen Segmenten und physischem Speicher

Das System benutzt die vier Segmentregister zur Aufnahme der Basisadressen der gegenwärtig adressierbaren Segmente. Das CS-Register zeigt auf das laufende Codesegment; Instruktionen sind in diesem Segment enthalten. Das SS-Register zeigt auf das laufende Stapelsegment; Stapelvorgänge benutzen Speicherstellen in diesem Segment. Das DS-Register zeigt auf das laufende Datensegment; im allgemeinen enthält es Programmvariablen. Das ES-Register zeigt auf ein laufendes zusätzliches Segment, das im allgemeinen auch für den Datenspeicher benutzt wird.

Kennzeichen

Statuskennzeichen reflektieren die Eigenschaften der Resultate arithmetischer oder logischer Vorgänge. Unterschiedliche Maschineninstruktionen wirken sich auf die Kennzeichen verschieden aus. Im allgemeinen reflektieren Statuskennzeichen jedoch folgende Bedingungen:

1. Ist AF (das Zusatz-Übertragskennzeichen) gesetzt, hat ein Übertrag der vier wertniedrigsten Bits in die vier werthöchsten Bits oder ein Entleihen von den vier werthöchsten Bits in die vier wertniedrigsten Bits einer 8-Bit-Größe (wertniedrigstes Byte einer 16-Bit-Größe) stattgefunden. Dieses Kennzeichen wird bei dezimalen arithmetischen Instruktionen benutzt.

2. Ist CF (das Übertragskennzeichen) gesetzt, hat ein Übertrag von dem werthöchsten oder ein Entleihen in das werthöchste Bit des Resultats (8- oder 16-Bits) stattgefunden. Dieses Kennzeichen wird für Additions- und Subtraktions-Instruktionen benutzt. Rotations-Instruktionen können auch ein Bit im Speicher oder ein Register isolieren, indem es in das Übertragskennzeichen gestellt wird.

3. Ist OF (das Überlaufkennzeichen) gesetzt, ist ein arithmetischer Überlauf eingetreten, d. h. eine signifikante Ziffer verlorengegangen, weil die Größe des Resultats die Kapazität der Bestimmungsort-Speicherstelle überschritten hat. Eine Unterbrechung-bei-Überlauf-Instruktion steht in diesem Fall zur Verfügung.

4. Ist SF (das Vorzeichen-Kennzeichen) gesetzt, ist das werthöchste Bit des Resultats 1. Da negative Binärziffern in der Standard-Zweier-Komplement-Aufzeichnung dargestellt werden, gibt SF das Vorzeichen des Resultats an (0 = Positiv, 1 = Negativ).

5. Ist PF (Paritätskennzeichen) gesetzt, hat das Resultat gerade Parität, eine gerade Anzahl von 1-Bits.

6. Ist ZF (Null-Kennzeichen) gesetzt, ist das Resultat des Vorgangs 0.

Programme können die drei Steuerkennzeichen setzen und löschen, um Prozessorvorgänge wie folgt abzuändern:

1. Ist DF (das Richtungskennzeichen) gesetzt, werden Ketteninstruktionen automatisch dekrementiert, d. h. die Verarbeitung der Ketten verläuft von hohen auf niedrige Adressen bzw. von rechts nach links. Durch Löschen von DF werden Ketteninstruktionen automatisch inkrementiert oder von links nach rechts verarbeitet.

2. Ist IF (das Unterbrechungs-Freigabekennzeichen) gesetzt, kann die CPU externe (maskierbare) Unterbrechungsanforderungen erkennen. Bei Löschen von IF werden diese Unterbrechungen gesperrt. IF wirkt sich weder auf nicht maskierbare externe noch auf intern generierte Unterbrechungen aus.

3. Ist TF (das Auffangstellen-Kennzeichen) gesetzt, tritt der Prozessor zur Fehlerbehebung in den Einzelschrittmodus ein. In diesem Modus generiert die CPU automatisch eine interne Unterbrechung nach jeder Instruktion und erlaubt dem Debugger, ein Programm während der Ausführung nach jeder Instruktion zu prüfen.

Abbildung A-3 zeigt die Verwendung der Steuer- und Statuskennzeichen.

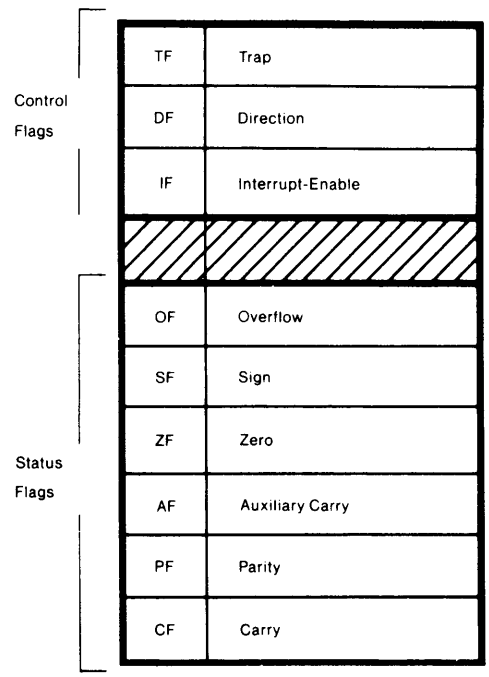


Abbildung A-3. Kennzeichen

A.2.2 Speicher

Dieser Abschnitt behandelt die Speicherorganisation und -Adressierung.

Speicherorganisation

Im Speicher sind Speicherstellen identische Anordnungen von 8-Bit-Bytes. Das System speichert immer Wortdaten mit dem signifikanteren Byte in der höheren Speicherstelle. In Abbildung A-4 ist z. B. der Wert des bei Adresse 724H gespeicherten Wortes 5502H.

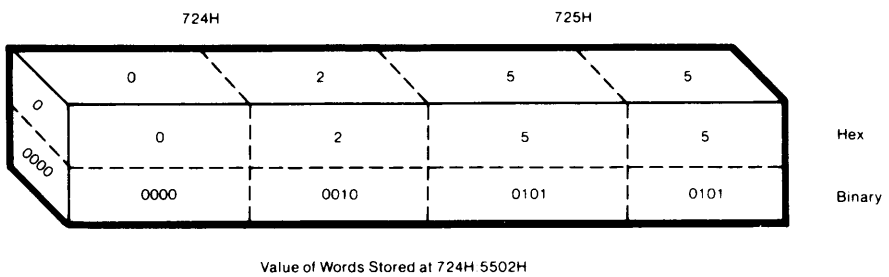


Abbildung A-4. Speichern von Wortvariablen

Doppelwort-Datenelemente, die benutzt werden, um Adressen außerhalb der Segmente zu halten, auf welche die Segmentregister gegenwärtig zeigen, enthalten einen Offset-Wert im niedrigeren adressierten Wort und eine Segment-Basisadresse im höheren adressierten Wort. Das höhere adressierte Byte eines Wortes enthält das signifikanteste Byte des Wortes. In Abbildung A-5 ist z. B. der Wert des bei Adresse 0004H gespeicherten Anzeigers Segment-Basisadresse 3B4CH und Offset 0065H.

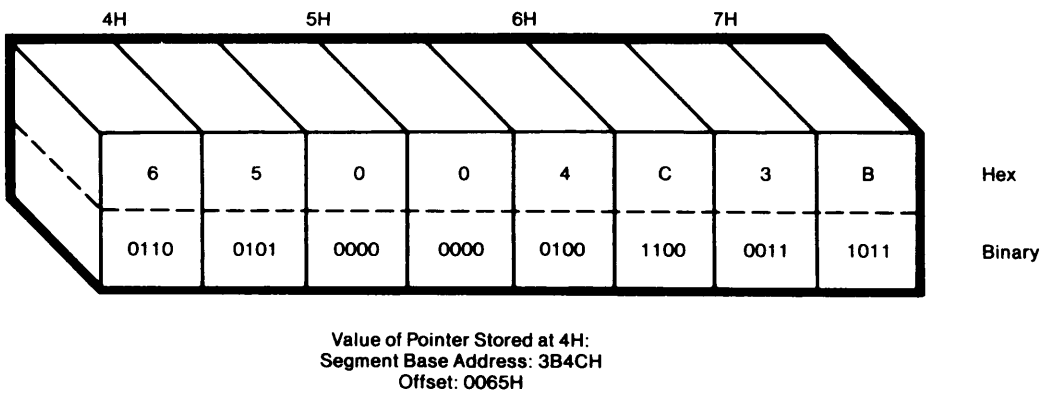


Abbildung A-5. Speichern einer Adresse in einem Doppelwort

Physische Adressengenerierung

Jede Speicherstelle hat im Grunde zwei Arten von Adressen, eine physische und eine logische. Eine physische Adresse ist der 20-Bit-Wert, der jedes Byte des Megabyte-Adreßraumes eindeutig identifiziert. Dieselbe physische Adresse kann verschiedene logische Adressen tragen.

Eine logische Adresse besteht aus einem Segment-Basiswert und einem Offset-Wert. Bei jeder Speicherstelle identifiziert der Segment-Basiswert das erste Byte eines Segments, das die Stelle enthält, und der Offset-Wert ist die Anzahl der Bytes vom Anfang des Segments bis zur jeweiligen Stelle. Das niedrigste adressierte Byte in einem Segment hat einen Offset von 0.

Im Speicher oder in den Registern gespeicherte Basis- und Offset-Werte sind vorzeichenlose 16-Bit-Größen. Um jedoch physische Adressen bis zu 1MB darzustellen, sind 20 Bits erforderlich. Da Segmente an 16-Byte-Grenzen beginnen, ist jede Segment-Basisadresse eine 20-Bit-Ziffer mit Null in den vier wertniedrigsten Bits. Zur Bezugnahme auf eine Segment-Basisadresse durch ein 16-Bit-Wort, stellt der Wert des Wortes die obersten 16 Bits der 20-Bit-Basisadresse dar. Das heißt, die Segment-Basisadresse entspricht dem Wert in einem Segment-Basisregister und ist um vier Binärpositionen nach links versetzt. Wenn z. B. ES den Wert 6789H enthält, dann zeigt ES auf das Segment, das an der physischen Adresse 67890H beginnt.

Eine physische Adresse wird von einem Basis:Offset-Paar abgeleitet, indem der 16-Bit-Offset zum Inhalt eines Segmentregisters, das um vier Positionen versetzt ist, hinzufügt wird (siehe Abbildung A-6). Die resultierende 20-Bit-Größe identifiziert eindeutig eine Stelle im physischen Speicher.

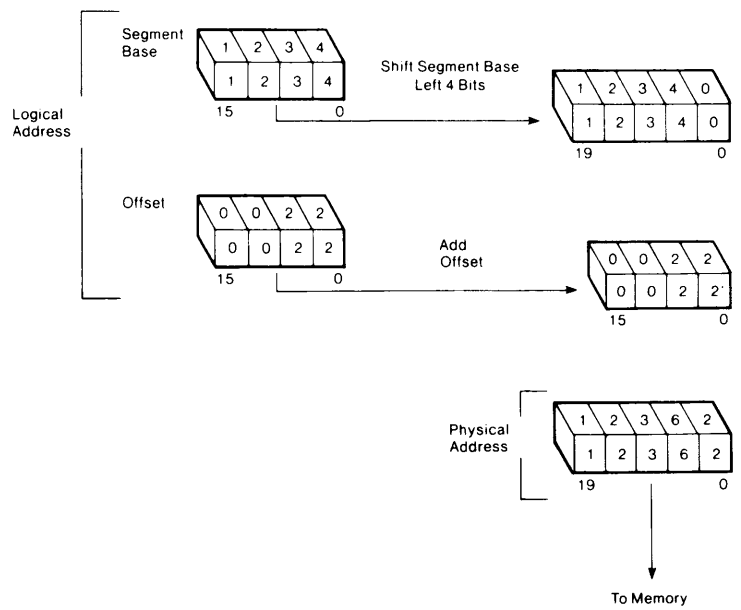


Abbildung A-6. Von logischer zu physischer Adressengenerierung

Mit logischen Adressen können Programme ohne Kenntnis über Code-Position im Speicher entwickelt werden. Dies erleichtert dynamische Verwaltung der Speicherresourcen. Um dynamisch verschiebbar zu sein, darf ein Programm seine Segmentregister nicht laden oder ändern und nicht direkt auf eine Stelle außerhalb des laufenden Codesegments übertragen. Mit anderen Worten, alle Offsets im Programm müssen sich auf feste Werte in den Segmentregistern beziehen.

A.3 DOS - TECHNISCHE DATEN

Dieser Abschnitt gibt einen technischen Überblick über WANG PC DOS, einschließlich DOS Befehlsprozessor und dessen Gebrauch der Systembetriebsmittel.

A.3.1 Inhalt einer DOS Diskette

Eine Systemdisk muß folgende Elemente enthalten, damit der DOS Befehlsprozessor geladen und benutzt werden kann:

- Einen Start-Sektor. Dies ist der erste Sektor auf allen mit dem FORMAT-Befehl formatierten Disks. Durch den FORMAT-Befehl wird dieser Sektor auf alle Disks gesetzt, um den Ablauf des Betriebssystems zu beginnen.
- Dateizuordnungstabellen (FATs). (Siehe Abschnitt A.4.6.)
- Diskverzeichnis. (Siehe Abschnitt A.4.5.)
- BIOS.SYS, das Grund-E/A-System. Dies ist der Hardware-abhängige Code, der alle Hardware-Unterbrechungen bearbeitet und die gesamte E/A für das Betriebssystem durchführt.
- MSDOS.SYS, der Betriebssystemnukleus. Dies ist eine höhere Anwenderprogramm-Schnittstelle. Sie enthält Daten- und Dateibearbeitungsroutinen und eine Anzahl von Funktionen, die leicht durch Anwenderprogramme zugänglich sind.
- CONFIG.SYS, eine System-Konfigurationsdatei. (Siehe Abschnitt A.3.2.)
- COMMAND.COM, der Befehlsprozessor. (Siehe Abschnitt A.3.3.)

Abbildung A-7 zeigt die Beziehung zwischen BIOS.SYS, MSDOS.SYS, COMMAND.COM und den Systemresourcen.

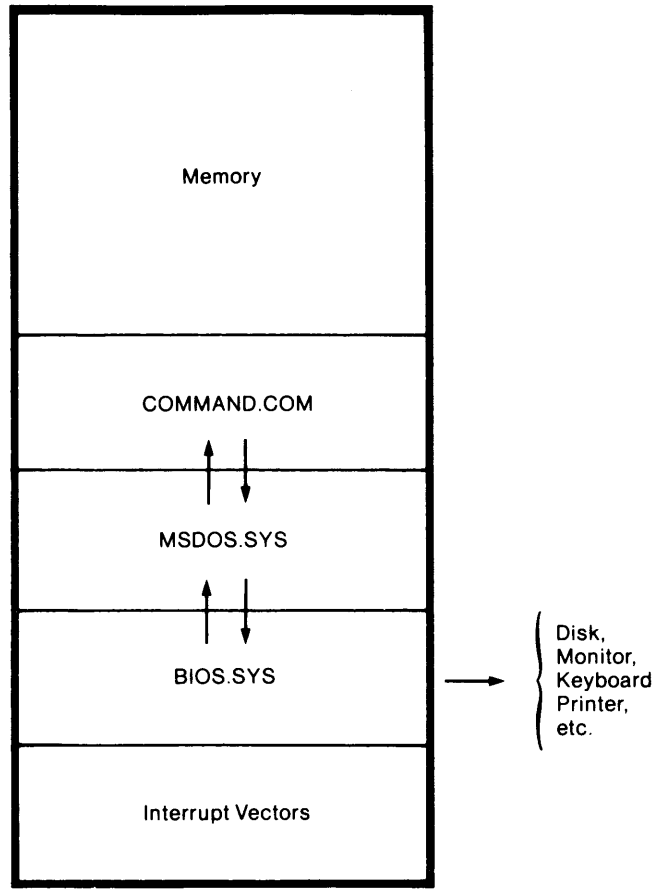


Abbildung A-7. DOS Betriebsmittel

A.3.2 DOS Initialisierung

Bei Start oder Wiederanlauf des Systems liest der Computer den Start-Sektor, lädt den Start-Code in den oberen Speicher und übergibt ihm die Steuerung. Der Start-Code liest das Diskverzeichnis, um sicherzustellen, daß sich BIOS.SYS und MSDOS.SYS auf der Disk befinden. Falls dies nicht der Fall ist, erscheint eine Fehlermeldung. Findet das Start-Programm diese Dateien, werden sie geladen und BIOS.SYS übernimmt die Steuerung. MSDOS.SYS belegt den ersten verfügbaren Speicherraum nach BIOS.SYS. BIOS.SYS und MSDOS.SYS können an irgendeiner Stelle auf der Disk erscheinen.

BIOS.SYS enthält am Anfang einen langen Aufruf an seinen Initialisierungscode, BIOSINIT. BIOSINIT befindet sich am oberen Ende von BIOS.SYS. Verschiedene BIOS Initialisierungen werden ausgeführt. Dann macht BIOS.SYS einen langen Sprung zur SYSINIT Routine.

Die SYSINIT-Routine initialisiert DOS und liest CONFIG.SYS. SYSINIT erstellt einen Programm-Segmentvorsatz (PSH) am untersten Segment für das Initialprogramm, dem DOS die Steuerung übergibt. Dann übergibt SYSINIT diesem Programm die Steuerung.

Die nachstehende Aufstellung enthält Befehle für die CONFIG.SYS Datei. Die Befehle können in beliebiger Reihenfolge erscheinen. Wang-spezifische Werte werden angeführt, wenn sie vom Standardwert abweichen.

BUFFERS = <Zahl>	Bestimmt die Anzahl der Sektorpuffer in der Systemliste. Enthält CONFIG.SYS mehrere BUFFERS-Befehle, gibt der Wert im letzten Befehl die Anzahl der Puffer an.
FILES = <Zahl>	Anzahl der eröffneten Dateien, auf die die XENIX Systemaufrufe zugreifen können. Der Standardwert ist 5. Eine Zahl von oder weniger als 5 wird ignoriert. Der Wert von FILES wurde von Wang Laboratories auf 10 festgelegt.
DEVICE = <Filespez.>	Veranlaßt die Installation des Geräte- antriebs in <Filespez.> in die System- liste.
BREAK = <ON oder OFF>	Ist ON (Standardwert OFF) angegeben, wird KTRL + C jedesmal an der Konsolen- eingabe gesucht, wenn die Steuerung zu DOS zurückkehrt. Der von Wang Laboratories festgelegte Wert ist ON.
SWITCHAR = <Zeichen>	Veranlaßt den Funktionsaufruf 37H, <Zeichen> als aktuellen Schalterdesig- nator zurückzugeben. Der Standardwert ist "/". Der Schalterdesignator wurde von Wang Laboratories auf "-" festge- legt.
AVAILDEV = <TRUE oder FALSE>	TRUE, der Standardwert, bedeutet, daß ein Gerät mit dem reservierten Namen <Gerät> oder mit /Gerät/<Gerät> ange- sprochen werden kann. FALSE bedeutet, daß nur /Gerät/<Gerät> auf das Gerät und <Gerät> alleinstehend auf eine Datei mit diesem Namen im aktuellen Verzeichnis verweist.

SHELL = <Filespez.>	Leitet die Ausführung des Haupt-Befehlsprozessors ab letzter Eintragung im <Filespez.> ein. Der von Wang Laboratories festgelegte Standardwert ist MENUDRVR.COM. Wird COMMAND.COM angegeben, werden die Menüs nicht geladen, und der DOS Befehlsprozessor wird zum Haupt-Befehlsprozessor.

CONFIG.SYS ist eine ASCII-Textdatei, deren Zeilen mit einer Zeilenschaltung und einem Wagenrücklauf enden. Ein typischer CONFIG.SYS Befehlssatz lautet wie folgt:

```
BUFFERS = 10
FILES = 10
DRIVER = Network
BREAK = ON
SHELL = a:command.com a: -p
```

-p ist ein Schalter, der anzeigt, daß COMMAND.COM permanent sein soll; COMMAND.COM könnte sonst später überschrieben werden.

WARNUNG:
Beim Modifizieren von CONFIG.SYS äußerst vorsichtig sein. Falls ein Fehler unterläuft, ist es möglich, daß das System nicht mehr gestartet werden kann. Stets Datensicherungskopien des ursprünglichen CONFIG.SYS aufbewahren.

A.3.3 DOS Befehlsprozessor

Der Befehlsprozessor (COMMAND.COM) besteht aus drei Abschnitten, die verschiedene Teile des Speichers belegen, und zwar einen residenten, einen Initialisierungs- und einen Übergangsabschnitt (siehe Abschnitt A.3.4, DOS Speicherverwendung).

Der residente Abschnitt befindet sich im Speicher direkt nach MSDOS.SYS und dessen Datenbereich. Dieser Abschnitt verwendet Routinen zur Bearbeitung der Unterbrechungen 22H (Abschluß), 23H (UMSCH + ANNULLIER), 24H (Abbruchfehler) und 27H (beenden, jedoch resident bleiben) und ebenfalls Code zum Wiederherstellen des Übergangsabschnitts. Dieser Abschnitt von COMMAND.COM bearbeitet auch alle normalen Diskfehler, einschließlich Anzeige von Fehlermeldungen und Interpretieren der auf den Dialoghinweis "Abbrechen, Wiederholen oder Ignorieren" erfolgenden Antwort.

Dem residenten Abschnitt folgt ein Initialisierungsabschnitt, der die Steuerung während des System-Einschaltvorgangs übernimmt. Dieser Abschnitt bestimmt die Adresse des Segments, in das die Programme geladen werden. Das erste COMMAND.COM Programm überlagert den Initialisierungsabschnitt, der nicht mehr benötigt wird.

Der Übergangsabschnitt, der am oberen Ende in den Speicher geladen wird, ist der Befehlsprozessor selbst mit Prozessoren für interne Befehle und Stapeldateien und einer Routine zum Laden und Ablauf der externen Befehle. (Externe Befehle sind Routinen mit den Dateinamenerweiterungen .COM oder .EXE). Dieser COMMAND.COM Abschnitt zeigt den DOS Dialoghinweis an, liest einen Befehl von der Endformat- oder Stapeldatei und führt den Befehl aus. Er erstellt einen PSH für externe Befehle direkt nach dem residenten Abschnitt von COMMAND.COM, lädt das externe Befehlsprogramm in dieses Segment, setzt die Abschluß- und UMSCH + ANNULLIER-Unterbrechungsadressen, so daß sie auf den residenten Abschnitt von COMMAND.COM zeigen und übergibt dann die Steuerung an das geladene Programm.

Der Übergangsabschnitt wird bei Beendigung eines Programms, durch das er überlagert wurde, wiederhergestellt. Falls der überlagerte Teil von COMMAND.COM nicht verfügbar ist, weil die Disk entfernt wurde, auf der er enthalten ist, oder falls der residente Abschnitt von COMMAND.COM auf einen nicht korrekten Übergangsabschnitt stößt, erscheint die Mitteilung, eine COMMAND.COM Disk in das Standardlaufwerk einzulegen und eine beliebige Taste zu drücken. Soweit Raum verfügbar ist, lädt COMMAND.COM .EXE Dateien, die für den oberen Teil des Speichers bestimmt sind, direkt unter dem Übergangsabschnitt, um beim Ladevorgang zu vermeiden, daß sich COMMAND.COM selbst überlagert.

A.3.4 DOS Speicherverwendung

Abbildung A-8 zeigt einen bei Ablauf von MSDOS.SYS typischen Speicherbelegungsplan. Alle Adressen sind physische Adressen in der Form NNNNNH. H bedeutet, daß es sich um einen Hexadezimalwert handelt.

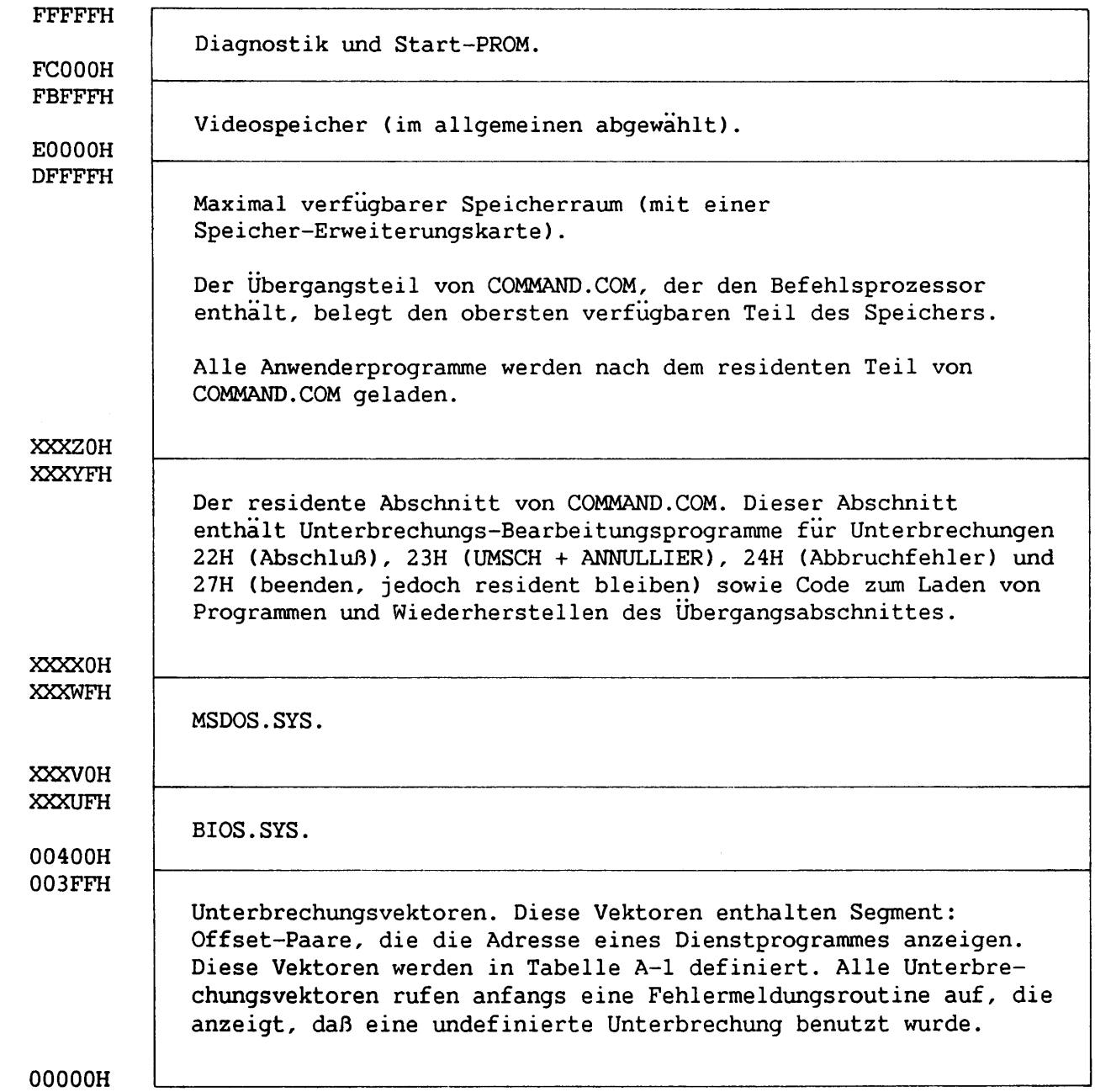


Abbildung A-8. DOS Speicherverwendung

Tabelle A-1 gibt Definitionen der Unterbrechungsvektoren im unteren Speicherteil.

Tabelle A-1. Unterbrechungsvektoren

Adresse	Verwendung
C0H - FFH	für Anwenderprogramme reserviert
80H - BFH	für BIOS reserviert
40H - 7FH	nicht definiert
20H - 3FH	für das Betriebssystem reserviert
14H - 1FH	für Emulator Intel 8087 reserviert
05H - 13H	von Intel reserviert
04H	ausschließlich zugeordnet - Überlauf-Unterbrechung
03H	ausschließlich zugeordnet - 1-Byte INT Instruktion
02H	ausschließlich zugeordnet - nicht maskierbare Unterbrechung
01H	ausschließlich zugeordnet - Einzelschritt-Unterbrechung
00H	ausschließlich zugeordnet - Divisionsfehler-Unterbrechung

A.3.5 Programm-Initialisierung

Wird ein externer Befehl erteilt, lokalisiert COMMAND.COM das Programm im untersten verfügbaren Bereich, direkt über dem residenten Abschnitt von COMMAND.COM. COMMAND.COM erstellt den PSH.

Der PSH speichert eine INT 20 Instruktion bei Offset 0. Der PSH speichert auch die Adressen für die UMSCH + ANNULLIER Unterbrechungen. Nach Beendigung gibt das Programm die Steuerung an COMMAND.COM zurück, und zwar entweder durch einen Sprung zu Offset 0, durch Erteilung einer INT 20 Instruktion, durch Erteilung einer INT 20 Instruktion, während das AH Register auf 0 gesetzt ist, oder durch Benutzung des Funktionsaufrufes 4CH.

Jede dieser Methoden veranlaßt, daß die Steuerung durch eine INT 20 Instruktion zum residenten Abschnitt von COMMAND.COM übergeht. COMMAND.COM stellt die Ausgangsadressen Abschluß und UMSCH + ANNULLIER vom PSH wieder her. Dann übergibt COMMAND.COM die Steuerung an die Abschlußadresse, die im beendenden Programm angegeben wird. Falls es sich um ein Programm handelt, das von einer Stapeldatei gelaufen wird, kehrt die Steuerung zur Stapeldatei zurück. Anderenfalls kehrt die Steuerung zum Übergangsabschnitt von COMMAND.COM zurück, der den DOS Dialoghinweis anzeigt und einen anderen Befehl von der Tastatur erwartet. Bei Beendigung über INT 20 oder 21, muß das Programm das CS Register gemäß dem PSH Segment setzen.

Abbildung A-9 zeigt das PSH Format. Die Speichergröße ist die Anzahl der Absätze. Das Wort bei Offset 6 enthält die Anzahl der Bytes, die im Segment verfügbar sind. Offsets sind hexadezimal.

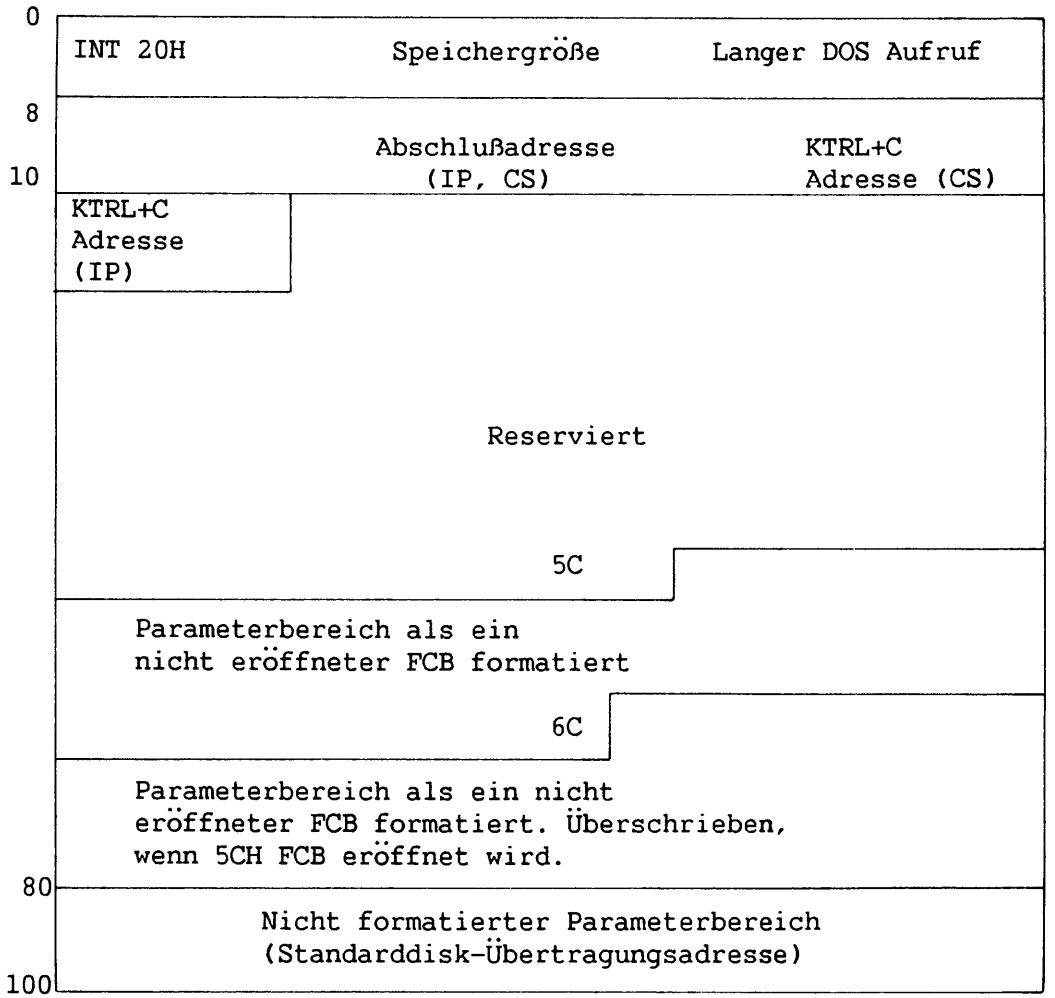


Abbildung A-9. Programm-Segmentvorsatz

Folgendes tritt ein, wenn die Steuerung auf ein Programm übergeht:

- Falls die Laufwerkbezeichnung des ersten oder zweiten Parameters unzulässig war, enthält das AL bzw. AH Register FFH. Anderenfalls enthalten sie 00H.
- Das Wort bei Offset 6 enthält die Anzahl der Bytes, die im Segment verfügbar sind.
- Das Wort bei Offset 2 enthält die gesamte Anzahl der Absätze im Speicher.

- Der Inhalt der Dateisteuerblöcke (FCBs) bei 5CH und 6CH wird von den ersten zwei Parametern des Befehls an definiert, der das Programm aufgerufen hat (siehe Abschnitt A.4.1).

- Die Standarddisk-Übertragungsadresse (DTA) ist 128 Bytes, die bei 80H im PSH beginnen (siehe Abschnitt A.4.2). Das Byte bei 80H enthält die Anzahl der Zeichen, einschließlich führender und eingeschlossener Trennsymbole, die dem Befehlsschlüsselwort folgen. Bei 81H beginnt ein nicht formatierter Parameterbereich, der die Zeichen selbst enthält.

Ein .COM Programm ist ein direktes Speicherbild, einschließlich Raum für nicht initialisierte Daten. Das gesamte Programm muß in ein Speichersegment geladen werden. Daher dürfen .COM Programme 64KB nicht überschreiten. Folgendes tritt ein, wenn ein .COM Programm die Steuerung erhält:

- Die vier Segmentregister enthalten das PSH Segment.

- Der Instruktionsanzeiger (IP) enthält 100H.

- Der Stapelanzeiger (SP) enthält das Ende des Programmsegments oder des Übergangsabschnitts von COMMAND.COM, je nachdem welches niedriger ist. Um einen Stapelspeicher von 100H zu ermöglichen, wird der Wert bei Offset 6 um diese Menge reduziert.

- Ein aus Nullen bestehendes Wort wird auf den Stapelspeicher geschoben.

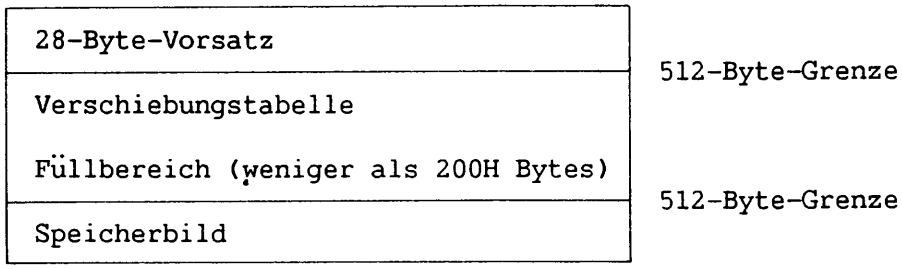
.EXE Programme sind Ausgabe vom Linker mit verschiebbaren Adressen. .EXE Programme können größer als 64KB sein. Der Linker erstellt einen 28-Byte-Vorsatz und eine Verschiebungstabelle von maximal 200H Bytes für jedes .EXE Programm. COMMAND.COM benutzt die Information im Vorsatz, um die Register aufzustellen, die das .EXE Programm lokalisieren. Die nachstehende Aufstellung zeigt das Format eines .EXE Dateivorsatzes, der als Assemblerstruktur definiert ist.

```
EXE file        STRUC
exe_signature       DW ?        ; muß "ZM" (5A4DH) enthalten
exe_len_mod 512     DW ?        ; Anzahl der auf der letzten
                                ; Seite enthaltenen Bytes
                                ; (untere 9 Bits der Länge)
exe_pages           DW ?        ; Anzahl der 512-Byte-Seiten
                                ; in der Datei einschließlich
                                ; Vorsatz
exe_rle_count       DW ?        ; Anzahl der Eintragungen in
                                ; der Verschiebungstabelle
exe_par_dir         DW ?        ; Anzahl der 16-Byte-Absätze
                                ; im Vorsatz
exe_min_BSS         DW ?        ; minimal benötigte Anzahl der
                                ; Absätze vor Ende des geladenen
                                ; Programms (für 2.0 und folgende
                                : DOS Versionen)
exe_max_BSS         DW ?        ; maximal benötigte Anzahl der
                                ; Absätze über dem Ende des
                                ; Programms. 0FFFFH zeigt an,
                                ; daß das Programm so niedrig wie
                                ; möglich liegt.
exe_SS              DW ?        ; Initialwert, der in SS vor
                                ; Ausführung zu laden ist.
                                ; Verschiebung muß diesen
                                ; Wert berichtigen.
exe_SP              DW ?        ; in SP vor Ausführung zu
                                ; ladender Anfangswert
exe_chksum          DW ?        ; Negativwert der Summe aller
                                ; Wörter in der Programmdatei
                                ; (gegenwärtig ignoriert)
exe_IP              DW ?        ; erster in IP vor Ausführung zu
                                ; ladender Offset
exe_CS              DW ?        ; in CS vor Ausführung zu
                                ; ladender Initialwert.
                                ; Verschiebung muß diesen Wert
                                ; berichtigen.
exe_rle_table       DW ?        ; relativer Byte-Offset der
                                ; Verschiebungstabelle vom
                                ; Anfang der Programmdatei.
                                ; Dies sollte 01CH sein.
exe_iov             DW ?        ; Nummer der Überlagerung.
                                ; Für den residenten Teil des
                                ; Programms ist iov = 0.

EXE FILE        ENDS
```

Die Verschiebungstabelle folgt dem festen Teil des Programmdatei-Vorsatzes. Die Eintragungen in der Verschiebungstabelle sind 4-Byte-Anzeiger auf Wörter in der Programmdatei, zu denen ein Verschiebungsfaktor hinzugefügt werden muß. Der Verschiebungsfaktor ist die physische Adresse des ersten Bytes des residenten Teils, der durch 16 geteilt wird. Der Verschiebungsfaktor muß zum ersten Wort jeder Tabelleneintragung hinzugefügt werden, bevor die Tabelleneintragung auf die eigentliche Stelle des Wortes zeigt.

Der residente Teil beginnt an der ersten 512-Byte-Grenze nach dem Ende der Verschiebungstabelle. Falls keine Verschiebung stattfindet, beginnt der residente Teil direkt nach dem Vorsatz. Die gesamte Programmdatei mit Verschiebungstabelle setzt sich wie folgt zusammen:



Folgendes tritt ein, wenn die Steuerung auf ein .EXE Programm übergeht:

- Die Register DS und ES enthalten das PSH Segment.
- Die Register CS, IP, SS und SP enthalten Werte, die von COMMAND.COM bei Verwendung der Information aus dem Dateivorsatz gesetzt werden.

A.3.6 Diskfehler

Wenn ein Disk-Lese- oder Schreibfehler auftritt oder die Dateizuordnungstabelle, FAT, beschädigt ist, überträgt DOS mit einer INT 24H (Abbruchfehler) Instruktion die Steuerung an ein Fehlerbearbeitungsprogramm. COMMAND.COM enthält das Standard-Fehlerbearbeitungsprogramm. Um ein anderes Fehlerbearbeitungsprogramm zu benutzen, den 24H Unterbrechungsvektor mit Funktionsaufruf 25H gemäß der Adresse des jeweiligen Codes setzen. DOS erteilt Fehlerinformation über Register und unterstützt Abbrechen, Wiederholen oder Ignorieren über Rückkehrcodes.

Wenn DOS einem Programm Steuerung übergibt, speichert es nicht den ursprünglichen Inhalt der Abbruchfehleradresse. Wird diese Unterbrechungsadresse geändert, muß das Programm vor Beendigung seinen ursprünglichen Inhalt zuerst speichern und dann wiederherstellen.

A.4 DATEI- UND DISKVERWALTUNG

Mit Hilfe von DOS können Dateien (über Unterbrechung 21H) erstellt, umbenannt, gelesen, geschrieben und gelöscht werden. Die Dateibearbeitung benutzt zwei Speicherbereiche und vier Diskspeicherbereiche. Die Speicherbereiche sind der Dateisteuerblock (FCB), dem ein erweiterter Dateisteuerblock vorausgehen kann, und die Disk-Übertragungsadresse (DTA). Die Diskspeicherbereiche bestehen aus dem reservierten Start-Sektor, dem Diskverzeichnis, der Dateizuordnungstabelle (FAT) und dem Dateiabschnitt. Speicherverwaltung für Dateien wird in Abschnitt A.4.1 und A.4.2 beschrieben. Die nachstehenden Unterabschnitte beschreiben die Diskverwaltung.

A.4.1 Dateisteuerblock-Definition

Dieser Unterabschnitt beschreibt den DOS FCB und erweiterten FCB. Jede Bezugnahme in der Beschreibung von DOS Funktionsaufrufen (Anhang C) an ein FCB, eröffnet oder nicht eröffnet, kann entweder einen normalen oder einen erweiterten FCB benutzen. Programme, die für MS-DOS Versionen vor 2.0 geschrieben werden, müssen jedoch FCBs benutzen.

DOS Dateisteuerblock

Der DOS Dateisteuerblock wird wie folgt formatiert:

Byte 0	Laufwerkanzeiger. 0 = Standardlaufwerk, 1 = Laufwerk A, 2 = Laufwerk B usw. Die maximale Laufwerkanzahl ist 64.
Byte 1-8	Dateiname. Besteht die Datei aus weniger als 8 Zeichen, muß der Name mit abschließenden Leerstellen links ausgerichtet sein.
Byte 9-11	Dateinamenerweiterung. Falls sie aus weniger als 3 Zeichen besteht, muß die Erweiterung mit abschließenden Leerstellen (20H) links ausgerichtet sein. Sie kann auch nur aus Leerstellen bestehen.
Byte 12-13	Aktueller Block. Dieses Wort (wertniedrigstes Byte zuerst) bestimmt den aktuellen Block von 128 Sätzen, bezogen auf den Anfang der Datei, in der sequentielle Disk-Lese- und Schreibvorgänge stattfinden. Null gibt den ersten Block der Datei an; eins den zweiten usw. Zusammen mit dem aktuellen Satzfeld (Byte 32), identifizieren diese Bytes einen bestimmten logischen Satz.
Byte 14-15	Größe des Satzes, mit dem gearbeitet werden soll. Dieses Wort sofort nach Eröffnung einer Datei füllen, wenn die vorgegebene logische Satzgröße (128 Bytes) nicht gewünscht wird. Bei Eröffnungs- und Erstellungsvorgängen wird dieses Feld bei 128 festgelegt. Ein Lese- oder Schreibversuch bei einem auf Null festgelegten Feld ändert es auch auf 128 ab.
Byte 16-19	Dateigröße. Dies ist die aktuelle, in Bytes ausgedrückte Größe der Datei. Anwenderprogramme können dieses Feld lesen, dürfen es jedoch nicht schreiben.
Byte 20-21	Erstellungs- oder letztes Schreibdatum. Bei allen Erstellungs- und Schreibvorgängen erhält dieses Feld das aktuelle Datum. Bei Eröffnung erhält dieses Feld das im Dateiverzeichnis eingetragene Datum. Um dieses Datum zu ändern, können Anwenderprogramme nach Schreiben in eine Datei, jedoch vor Beenden, dieses Feld modifizieren.

Das Format dieses 16-Bit-Feldes ist wie folgt: Bit 0-4, Tag des Monats; Bit 5-8, Monat des Jahres; Bit 9-15, Jahr minus 1980. Nur Nullen bedeutet kein Datum.

Byte 22-23 — Uhrzeit der Erstellung oder des letzten Schreibvorgangs. Dieses Feld wird auf die gleiche Weise wie das Datumfeld, Byte 20-21, fortgeschrieben. Das Format ist wie folgt: Bit 0-4, Sekunden/2; Bit 5-10, Minuten; Bit 11-15, Stunden.

Byte 24 — Kennzeichen. Bit 7 wird für eine Datei auf 0, für ein Gerät auf 1 gesetzt. Bit 6 wird auf 0 gesetzt, falls Änderungen vorliegen, die noch nicht geschrieben wurden. Bit 0 bis 5 enthalten den Geräte-ID.

Byte 25-26 — Diskadresse der ersten Zuordnungseinheit in der Datei.

Byte 27-28 — Diskadresse der letzten Zuordnungseinheit in der Datei.

Byte 29-31 — Diskadressen der Zuordnungseinheit, auf die zuletzt zugegriffen wurde und des Dateiverzeichnisses. Es handelt sich um 12-Bit-Adressen, die jeweils 1,5 Bytes belegen.

Byte 32 — Nächster Satz. Identifiziert den Satz innerhalb des aktuellen Blocks von 128 Sätzen, auf den ein sequentieller Lese- oder Schreibvorgang zugreifen kann. Siehe Byte 12-13, Aktueller Block.

Byte 32-35 — Zufallssatz. Dieses Feld muß nur dann bestimmt werden, wenn auf eine Datei mit einem Zufallslese- oder -schreibvorgang zugegriffen wird. Falls der Satz 64 Bytes oder größer ist, werden nur die ersten 3 Bytes als 24-Bit-Ziffer benutzt, die die Position eines Satzes in der Datei darstellen. Ist der Satz kleiner als 64 Bytes, werden alle 4 Bytes für eine 30-Bit-Ziffer zum selben Zweck benutzt. Dieses Feld ist somit groß genug, um jedes Byte in einer Datei bis zu 1 Megabyte zu adressieren.

Abbildung A-10 zeigt den Aufbaru des FCBs und des erweiterten FCBs. Offsets sind dezimal.

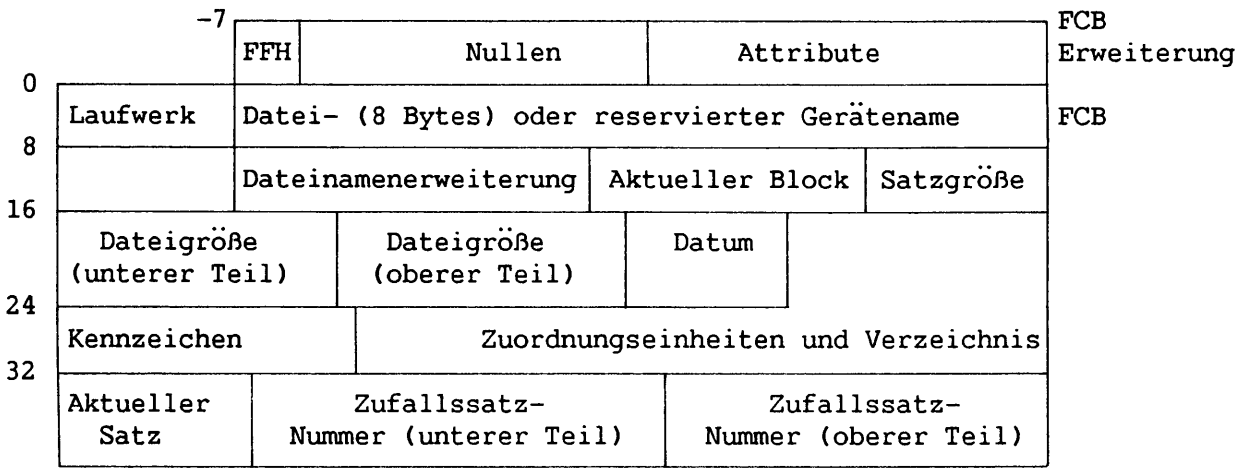


Abbildung A-10. Dateisteuerblock

Erweiterter Dateisteuerblock

Der erweiterte FCB ist ein besonderes Format, das verwendet wird, um Verzeichnisse nach Dateien abzusuchen, die von normalen Verzeichnis-Suchen ausgeschlossen sind (Systemdateien und verborgene Dateien). Das Format des erweiterten FCBs, das aus 7 Bytes vor einem normalen FCB besteht, sieht wie folgt aus:

FCB - 7	Kennzeichen. Ein hier gesetztes FF hex zeigt einen erweiterten FCB an.
FCB - 6 bis FCB - 2	Null-Feld.
FCB - 1	Attribut-Byte. Ist Bit 1 = 1, sind verborgene Dateien in Verzeichnis-Suchen eingeschlossen. Ist Bit 2 = 1, sind Systemdateien in Verzeichnis-Suchen eingeschlossen.

A.4.2 Disk-Übertragungsadresse

Die Disk-Übertragungsadresse (DTA) ist der Speicherbereich, der Daten für Disk-Lese- und -Schreibvorgänge enthält. Die DTA kann sich an jeder möglichen Stelle befinden. Das jeweilige Programm legt die DTA durch Benutzung des Funktionsaufrufs 1AH fest.

In einem Programm kann nur jeweils eine DTA in Effekt sein. Das Programm muß die DTA vor Durchführung von Disk-E/A setzen. Die bestimmte DTA bleibt solange für alle Diskvorgänge in Effekt, bis das Programm eine neue DTA durch einen weiteren 1A Funktionsaufruf bezeichnet. Wird die Steuerung an ein Programm übertragen, erstellt COMMAND.COM im PSH einen DTA Standardwert für 128 Bytes bei 80H (siehe Abschnitt A.3.5).

A.4.3 Diskettenformate

WANG PC BIOS unterstützt acht Diskettenformate, vier Formate mit 512-Byte-Sektoren und vier Formate mit 256-Byte-Sektoren. Dieser Abschnitt enthält die Spezifikationen dieser Formate einschließlich Inhalt des Start-Sektors. Falls die Spezifikationen für 512- und 256-Byte-Sektoren voneinander abweichen, erscheint zuerst der Wert für die 512-Byte-Sektoren; der Wert für die 256-Byte-Sektoren folgt direkt in Klammern.

Einseitig mit doppelter Aufzeichnungsdichte - SSDD (8 Sektoren pro Spur)

512 (256) Bytes pro Sektor
320 (640) Sektoren pro Diskette
163840 Bytes pro Diskette
FAT ID Byte =0FEH (0FAH)
2 FATS
1 (2) Sektor(en) pro FAT
1 (2) Sektor(en) pro Zuordnungseinheit
64 Diskverzeichnis-Eintragungen
32 Bytes pro Diskverzeichnis-Eintragung
Sektor 0 (0-1) = Ursektor
Sektor 1 (2-3) = Dateizuordnungstabelle (Kopie 1)
Sektor 2 (4-5) = Dateizuordnungstabelle (Kopie 2)
Sektor 3 bis 6 (6 bis 13) = Diskverzeichnis
Sektor 7 bis 319 (14 bis 639) = Datenbereich, nicht reserviert,
160256 Bytes

Einseitig mit doppelter Aufzeichnungsdichte - SSDD (9 Sektoren pro Spur)

512 (256) Bytes pro Sektor
360 (720) Sektoren pro Diskette
184320 Bytes pro Diskette
FAT ID Byte = 0FCH (0F8H)
2 FATS
2 (4) Sektoren pro FAT
1 (2) Sektor(en) pro Zuordnungseinheit
64 Diskverzeichnis-Eintragungen
32 Bytes pro Diskverzeichnis-Eintragung
Sektor 0 (0-1) = Ursektor
Sektor 1 bis 2 (2 bis 5) = Dateizuordnungstabelle (Kopie 1)
Sektor 3 bis 4 (6 bis 9) = Dateizuordnungstabelle (Kopie 2)
Sektor 5 bis 8 (10 bis 17) = Diskverzeichnis
Sektor 9 bis 359 (18 bis 359) = Datenbereich, nicht reserviert,
179712 Bytes

Doppelseitig mit doppelter Aufzeichnungsdichte - DSDD (8 Sektoren pro Spur)

512 (256) Bytes pro Sektor
640 (1280) Sektoren pro Diskette
327680 Bytes pro Diskette
FAT ID Byte = 0FFH (0FBH)
2 FATS
1 (2) Sektor(en) pro FAT
2 (4) Sektoren pro Zuordnungseinheit
112 Diskverzeichnis-Eintragungen
32 Bytes pro Diskverzeichnis-Eintragung
Sektor 0 (0-1) = Ursektor
Sektor 1 (2-3) = Dateizuordnungstabelle (Kopie 1)
Sektor 2 (4-5) = Dateizuordnungstabelle (Kopie 2)
Sektor 3 bis 9 (6 bis 19) = Diskverzeichnis
Sektor 10 bis 639 (14 bis 1179) = Datenbereich, nicht reserviert,
322560 Bytes

Doppelseitig mit doppelter Aufzeichnungsdichte - DSDD (9 Sektoren pro Spur)

512 (256) Bytes pro Sektor
720 (1140) Sektoren pro Diskette
368640 Bytes pro Diskette
FAT ID Byte = 0FDH (0F9H)
2 FATS
2 (4) Sektoren pro FAT
2 (4) Sektoren pro Zuordnungseinheit
112 Diskverzeichnis-Eintragungen
32 Bytes pro Diskverzeichnis-Eintragung
Sektor 0 (0-1) = Ursektor
Sektor 1 bis 2 (2 bis 5) = Dateizuordnungstabelle (Kopie 1)
Sektor 3 bis 4 (6 bis 9) = Dateizuordnungstabelle (Kopie 2)
Sektor 5 durch 11 (10 bis 23) = Diskverzeichnis
Sektor 12 bis 719 (24 bis 1139) = Datenbereich, nicht reserviert,
362496 Bytes

Start-Sektor

Abbildung A-11 führt die BIOS Parameterblock-Variablen auf, die der Start-Sektor für jedes Diskettenformat mit 512-Byte-Sektoren enthält. Abbildung A-12 enthält die gleiche Information für Formate mit 256-Byte-Sektoren.

TYP	BESCHREIBUNG	SSDD/8	SSDD/9	DSDD/8	DSDD/9
WORT	BYTES PRO SEKTOR	512 (200H)	512 (200H)	512 (200H)	512 (200H)
BYTE	SEKTOREN PRO ZUORDNUNGSEINHEIT	1	1	2	2
WORT	RESERVIERTE SEKTOREN (FÜR START BENÖTIGTE ANZAHL)	1	1	1	1
BYTE	ANZAHL DER FATS	2	2	2	2
WORT	ANZAHL DER URVERZEICHNIS-EINTRAGUNGEN	DIR 64 (40H)	64 (40H)	112 (70H)	112 (70H)
WORT	ANZAHL DER SEKTOREN/LOGISCHES BILD (d. h. ANZAHL DER SEKTOREN/FLOPPY)	320 (140H) BYTES= 163840	360 (168H) BYTES= 184320	640 (280H) BYTES= 327680	720 (2D0H) BYTES= 368640
BYTE	DATENTRÄGER-DESKRIPTOR (d. h. FATID)	0FEH	0FCH	0FFH	0FDH
WORT	ANZAHL DER SEKTOREN PRO FAT	1	2	1	2
WORT	SEKTOREN PRO SPUR	8	9	8	9
WORT	ANZAHL DER KÖPFE	1	1	2	2
WORT	ANZAHL DER VERBORGENEN SEKTOREN	0	0	0	0
WORT	ANZAHL DER FÜR START + FATS + VERZEICHNISSE RESERVIERTEN SEKTOREN	7 (07H)	9 (09H)	10 (0AH)	12 (0CH)

Abbildung A-11. Im Startsektor enthaltene BIOS Parameterblock-Variablen (512-Byte-Sektoren)

TYP	BESCHREIBUNG	SSDD/16	SSDD/18	DSDD/16	DSDD/18
WORT	BYTES PRO SEKTOR	256 (100H)	256 (100H)	256 (100H)	256 (100H)
BYTE	SEKTOREN PRO ZUORDNUNGSEINHEIT	2	2	4	4
WORT	RESERVIERTE SEKTOREN (FÜR START BENÖTIGTE ANZAHL)	2	2	2	2
BYTE	ANZAHL DER FATS	2	2	2	2
WORT	ANZAHL DER URVERZEICHNIS-EINTRAGUNGEN	64 (40H)	64 (40H)	112 (70H)	112 (70H)
WORT	ANZAHL DER SEKTOREN/LOGISCHES BILD (d. h. ANZAHL DER SEKTOREN/FLOPPY)	640 (280H) BYTES= 163840	720 (2D0H) BYTES= 184320	1280 (500H) BYTES= 327680	1440 (5A0H) BYTES= 368640
BYTE	DATENTRÄGER-DESKRIPTOR (d. h. FATID)	0FAH	0F8H	0FBH	0F9H
WORT	ANZAHL DER SEKTOREN PRO FAT	2	4	2	4
WORT	SEKTOREN PRO SPUR	16	18	16	18
WORT	ANZAHL DER KÖPFE	1	1	2	2
WORT	ANZAHL DER VERBORGENEN SEKTOREN	0	0	0	0
WORT	ANZAHL DER FÜR START + FATS + VERZEICHNIS RESERVIERTEN SEKTOREN	14 (0EH)	18 (12H)	20 (14H)	24 (18H)

Abbildung A-12. Im Startsektor enthaltene BIOS Parameterblock-Variablen (256-Byte-Sektoren)

A.4.4 Diskspeicherzuordnung

Der DOS Diskspeicherraum besteht aus vier Teilen:

- Reserviertem Sektor. Dieser Bereich enthält den DOS Start-Code.
- Dateizuordnungstabelle. FAT enthält Speicherstelleninformation für die Daten, aus denen jede Datei auf einer gegebenen Disk besteht.
- Diskverzeichnis. Das Verzeichnis enthält Information über jede Datei auf einer gegebenen Disk. Die Information besteht aus dem vollständigen Namen der Datei, der Größe, Attribute, Uhrzeit und Datum der letzten Modifikation und der Diskspeicherstelle des ersten Sektors.
- Dateiabschnitt. Der größte Teil des Diskspeicherraums ist für den Inhalt der Dateien reserviert.

Auf den mit der -S Option des FORMAT-Befehls formatierten Systemdisks sind die ersten vier Dateien im Diskverzeichnis BIOS.SYS, MSDOS.SYS, CONFIG.SYS und COMMAND.COM.

Eine einzelne Datei muß nicht unbedingt in zusammenhängenden Sektoren auf der Disk stehen, sondern kann verstreut sein, um den Diskspeicherraum besser auszunutzen. Durch die Verkettung von Dateisektoren kann auf eine Datei mit FAT und den im FCB angegebenen Feldern sequentiell oder direkt zugegriffen werden. Auf Sätze kann sequentiell mit dem sequentiellen Lese- oder Schreib-Funktionsaufruf zusammen mit dem aktuellen Block- und Satzfeld des FCBs zugegriffen werden. Auf Sätze kann mit dem direkten Lese- oder Schreib-Funktionsaufruf und den direkten Satzfeldern direkt zugegriffen werden. Die direkten Blockzugriffsfunktionen ermöglichen ebenfalls das Lesen oder Schreiben größerer Datenmengen mit einem Funktionsaufruf. Bei allen Methoden berechnet DOS selbst die korrekte Speicherstelle auf der Disk.

A.4.5 Diskverzeichnis

Der FORMAT-Befehl erstellt ein Verzeichnis auf jeder Diskette (siehe Abschnitt A.4.3). Im Disk- oder in untergeordneten Verzeichnissen gemachte Verzeichniseintragungen werden entsprechend Tabelle A-2 formatiert (Offset-Werte sind dezimal).

Tabelle A-2. Verzeichniseintragungen

Stelle	Bytes	Beschreibung
0	11	Dateiname und Erweiterung
11	1	Attributte. Tabelle A-3 definiert die Eintragungen in diesem Feld
12	10	Mit Nullen gefülltes Feld (für Erweiterung)
22	2	Uhrzeit der Erstellung oder des letzten Schreibvorgangs Bit 0-4 = Sekunden*2 5-10 = Minuten 11-15 = Stunden
24	2	Datum der Erstellung oder des letzten Schreibvorgangs Bit 0-4 = Tag 5-10 = Monat 9-15 = Jahr
26	2	Erste Zuordnungseinheit
28	4	Dateigröße in Bytes (max. 30 Bits)

Ist das erste Byte des Dateinamenfeldes E5, ist die Verzeichniseintragung frei. 00 zeigt das Ende des zugeordneten Verzeichnisses an.

Tabelle A-3 definiert die Eintragungen im Attribut-Byte eines Verzeichnisses.

Tabelle A-3. Datei-Attributcodes

Byte	Attribut
1H	Nur lesen
2H	Verborgene Datei
4H	Systemdatei
8H	Volume-ID
10H	Verzeichnis

Eine Datei kann eine beliebige Kombination der Attribute "Nur lesen", "verborgene Datei" und "Systemdatei" haben. Das "Volume-ID"- oder "Verzeichnis"-Attribut muß das einzige Attribut der Datei sein.

A.4.6 Dateizuordnungstabelle

DOS benutzt die Dateizuordnungstabelle, um Dateien Diskspeicherraum zuzuweisen, und zwar jeweils eine Zuordnungseinheit nach der anderen. Eine Kopie der FAT für die letzte in jedem Laufwerk benutzte Disk bleibt im internen Speicher erhalten, falls genügend Raum vorhanden ist. DOS schreibt die Dateizuordnungstabelle, wenn sich der Status des Dateiabschnitts auf der Disk ändert.

Die Dateizuordnungstabelle benutzt eine 12-Bit (1,5 Byte) Eintragung für jede Zuordnungseinheit auf der Disk. Die FAT-Eintragungsnummer Null wird in DOS als Dateiende-Auffangstelle benutzt und an BIOS weitergegeben, um zur Bestimmung des Diskformats beizutragen. Eintragung 1 wird für späteren Gebrauch reserviert. Die erste verfügbare Zuordnungseinheit ist die zugewiesene Eintragungsnummer zwei.

Jede FAT-Eintragung enthält drei hexadezimale Zeichen. Wird die Zuordnungseinheit nicht benutzt und ist verfügbar, enthält die Eintragung 000. Eintragungen größer als FF7H sind Dateiende-Markierungen. FF7H zeigt an, daß die Zuordnungseinheit einen beschädigten Sektor enthält. Alle weiteren hexadezimalen Zeichen stellen die relative Nummer der nächsten Zuordnungseinheit in der Datei dar. Die Verzeichniseintragung der Datei enthält die relative Zuordnungseinheitsnummer des ersten Sektors der Datei.

Um die absolute Spur und den absoluten Sektor eines Satzes zu ermitteln, die erste Zuordnungseinheitsnummer der Datei von der Verzeichniseintragung benutzen. Um die nächste Zuordnungseinheit zu finden, die FAT-Eintragung vom Verzeichnis mit 1,5 multiplizieren (Länge jeder FAT-Eintragung). Die ganze Zahl im Produkt ist ein Offset, der auf die FAT-Eintragung zeigt, welche die nächste Zuordnungseinheit aufzeichnet. Das Wort an dieser Eintragung nun mit der MOV Instruktion in ein Register verschieben. War die letzte Dateizuordnungseinheits-Nummer gerade, die wertniedrigsten 12 Bits des Registers beibehalten; war die Nummer ungerade, die werthöchsten 12 Bits beibehalten. Die resultierenden 12 Bits enthalten die Dateizuordnungseinheits-Nummer der nächsten Zuordnungseinheit in der Datei. Bestehen die 12 Bits alle aus 1 (FFFH), enthält die Datei keine weiteren Sektoren.

ANHANG B
MITTEILUNGEN

B.1 EINFÜHRUNG

Dieser Anhang enthält Linker-, Library-Verwalter- und Debugger-Fehlermeldungen, deren Ursache sowie entprechende Möglichkeiten zur Fehler- behebung.

B.2 LINKER-MITTEILUNGEN

Alle Fehler verursachen Abbruch des Linker-Vorgangs. Ist die Fehlerursache festgestellt und der Fehler behoben, muß der Linker daher noch einmal ablaufen.

Folgende Fehlermeldungen können während der Linker-Phase auftreten:

ATTEMPT TO ACCESS DATA OUTSIDE THE SEGMENT BOUNDS, POSSIBLY BAD OBJECT MODULE

Objektdatei wahrscheinlich falsch.

BAD NUMERIC PARAMETER

Numerischer Wert nicht in Ziffern. Nur Zeichen von 0 bis 9 benutzen.

CANNOT OPEN TEMPORARY FILE

Linker kann VM.TMP Datei nicht erstellen, weil Disk voll ist. Neue Disk einlegen. Nicht die Disk ändern, die list.MAP Datei erhält.

ERROR: DUP RECORD TOO COMPLEX

DUP Konstruktion in einem Assembler-Sprachmodul ist zu komplex. DUP Konstruktion im Assembler-Sprachprogramm vereinfachen.

ERROR: FIXUP OFFSET EXCEEDS FIELD WIDTH

Assembler-Sprachinstruktion benutzt eine kurze Adresse (nur Offset) für einen Programmteil, der vom Linker mehr als 64KB von der Basis entfernt positioniert wurde. Assembler-Sprachquelle aufbereiten und neu zusammenstellen.

INPUT FILE READ ERROR

Objektdatei wahrscheinlich falsch.

INVALID OBJECT MODULE

Objektmodul(n) fehlerhaft gebildet oder unvollständig (z. B. wenn der Assemblierungsvorgang angehalten wurde).

PROGRAM SIZE OR NUMBER OF SEGMENTS EXCEEDS CAPACITY OF LINKER

Gesamtgröße von 384KB und Segmentanzahl von 255 dürfen nicht überschritten werden.

REQUESTED STACK SIZE EXCEEDS 64KB

Größe von oder weniger als 64KB mit /STACK-Schalter angeben.

SEGMENT SIZE EXCEEDS 64K

Adressiersystem ist auf 64KB begrenzt.

SYMBOL DEFINED MORE THAN ONCE

Linker fand zwei oder mehr Moduln, die einen einzigen Symbolnamen definieren.

SYMBOL TABLE CAPACITY EXCEEDED

Zu viele lange Namen; Gesamtliste überschreitet 25KB.

TOO MANY EXTERNAL SYMBOLS IN ONE MODULE

Externe Symbole für jeden Modul sind auf 256 begrenzt.

TOO MANY GROUPS

Gruppenanzahl ist auf 10 begrenzt.

TOO MANY LIBRARIES SPECIFIED

Librarien-Anzahl ist auf 8 begrenzt.

TOO MANY PUBLIC SYMBOLS

Publik-Symbole sind auf 1024 begrenzt.

TOO MANY SEGMENTS OR CLASSES

Segmente und Klassen sind zusammen auf 256 begrenzt.

UNRESOLVED EXTERNALS: <Liste>

Aufgeführte externe Symbole haben keinen Definier-Modul unter den Moduln oder angegebenen Library-Dateien.

VM READ ERROR

VM.TMP Datei unlesbar. Diskproblem; nicht durch den Linker verursacht.

WARNING: NO STACK SEGMENT

Falscher Objektmodul oder ein Modulverknüpfungsversuch, den der Linker nicht bearbeiten kann, weil der Modul unverschiebbar (z. B. bereits verknüpft) ist.

WRITE ERROR IN TMP FILE

Kein Diskspeicherraum zur Vergrößerung der VM.TMP Datei.

WRITE ERROR ON RUN FILE

Im allgemeinen nicht genügend Diskspeicherraum für Programmdatei.

B.3 LIBRARY-VERWALTER-MITTEILUNGEN

Während des Library-Verwaltungsvorgangs können folgende Fehlermeldungen auftreten:

<Symbol> is a multiply defined PUBLIC. Proceed?

Ein neuer Modul definiert dasselbe Publik-Symbol wie ein schon in der Library enhaltener Modul. Die Entfernung der schon bestehenden Symbol-Definition ist zu bestätigen. Bei Verneinung bleibt die Library unbestimmt. Die Publik-Erklärung von einer der Objektmoduln entfernen, dann neu kompilieren oder neu zusammenstellen.

Allocate error on VM.TMP

Kein Speicherraum auf Disk bei Eröffnung der Datei.

Cannot create extract file

Kein Platz im Diskverzeichnis für die durch Kopieren einer Library-Datei (Stern-Befehl) zu erstellende Datei.

Cannot create list file.

Kein Platz für Listdatei im Diskverzeichnis.

Cannot nest response file

Antwortdatei kann keine Antwortdatei (<@Filespez.>) als Parameter enthalten.

Cannot open VM.TMP

Kein Speicherraum für VM.TMP im Diskverzeichnis.

Cannot write library file

Kein Speicherraum auf Disk.

Close error on extract file

Nicht genügend Speicherraum auf Disk für Datei, die durch Kopieren eines Moduls von der Library-Datei erstellt wurde (Stern-Befehl).

Error: An internal error has occurred

Wang Professional Computer Kundendienst anrufen.

Fatal Error: Cannot open input file

Objektdateiname entweder falsch gebildet oder im Diskverzeichnis nicht auffindbar.

Fatal Error: Module is not in the library

Ein Modul soll gelöscht werden, der nicht in der Library enthalten ist.

Input file read error

Falscher Objektmodul oder fehlerhafte Disk.

Invalid object module/library

Falscher Objektmodul und/oder falsche Library.

Library disk is full

Kein Speicherraum auf Disk.

Listing file write error

Kein Speicherraum auf Disk.

No library file specified

Keine Antwort auf Library-Datei-Dialoghinweis.

Read error on VM.TMP

Disk nicht zum Ablesen bereit. Plattenlaufwerktür offen oder Disk nicht richtig im Laufwerk.

Symbol table capacity exceeded

Zu viele Zeichen (ca. 30KB) in Publik-Symbolen.

Too many object modules

Mehr als maximal 500 Objektmoduln.

Too many public symbols

Mehr als maximal 1024 Publik-Symbole.

Write error on library/extract file

Nicht genügend Speicherraum auf Disk für eine Library-Datei oder eine Datei, die von einer Library gelöst wurde.

Write error on VM.TMP

Kein Speicherraum auf Disk.

B.4 DEBUGGER-MITTEILUNGEN

Während des Debug-Vorgangs können nachfolgende Fehlercodes erscheinen. Jeder Fehler bringt den damit verbundenen Debug-Befehl, nicht aber DEBUG selbst zum Abbruch.

FEHLERCODE	DEFINITION
BF	Ungültiges Kennzeichen (Bad Flag)

Zur Änderung eines Kennzeichens wurden Zeichen eingetragen, die nicht zu den gültigen Kennzeichenwertpaaren gehören (siehe Tabelle 4-2, Kennzeichen und Kennzeichencodes).

BP — Zu viele bedingte Programmstops (Too many Breakpoints)

Mehr als 10 bedingte Programmstops als Parameter für G-Befehl angegeben. G-Befehl mit 10 oder weniger Programmstops neu eintragen.

BR — Unzulässiges Register (Bad Register)

R-Befehl enthält einen unzulässigen Registernamen (siehe Abschnitt 4.6, R-Befehl).

DF — Doppelkennzeichen (Double Flag)

Zwei Werte für ein Kennzeichen eingetragen. Kennzeichenwert kann für jeden RF-Befehl nur einmal bestimmt werden.

ANHANG C
DOS UNTERBRECHUNGEN UND FUNKTIONSAUFRUFE

C.1 UNTERBRECHUNGEN

Die Unterbrechungsarten 20H bis 3FH, d. h. die absoluten Speicherstellen 80H bis FFH, sind für DOS Gebrauch reserviert.

Individuelle Unterbrechungsroutinen können mit Funktionsaufruf 25H (Unterbrechungsvektor festsetzen) geschrieben werden. Von der Unterbrechungsroutine über die IRET-Instruktion zurückkehren, die alle Register und Kennzeichen wiederherstellt. Bei Rückkehr ist es wichtig, den Unterbrechungsvektor wieder auf den ursprünglichen Wert festzulegen, damit er für andere Zwecke verfügbar ist.

Nachstehend werden die definierten Unterbrechungen zuerst alphabetisch, dann numerisch mit Erklärungen aufgelistet. Es handelt sich in allen Fällen um Hexadezimalwerte.

25	Absoluter Disk-Lesevorgang
26	Absoluter Disk-Schreibvorgang
24	Abbruchfehler
21	Funktionsanforderung
20	Programm abschließen
23	UMSCH + ANNULLIER Ausgangsadresse
27	Abschließen, jedoch resident bleiben

20 Programm abschließen. Auf diese Weise kann ein Programm verlassen werden. Dieser Vektor überträgt die Steuerung an die Logik in DOS, die die bei Eintritt in das Programm gültigen Werte der UMSCH + ANNULLIER Ausgangsadressen wiederherstellt. Alle Dateipuffer werden auf Disk geleert. Dateien, deren Länge sich geändert hat (siehe Funktionsaufruf 10H), sollten vor Unterbrechung geschlossen werden; anderenfalls enthält das Verzeichnis nicht die korrekte Länge. Bei Ausführung dieser Unterbrechung muß CS in Richtung des 100H Parameterbereichs zeigen.

21 Funktionsanforderung. Siehe Abschnitt C.2, Funktionsaufrufe.

22 Abschlußadresse. Die durch diese Unterbrechung dargestellte Adresse (88-8BH) übernimmt die Steuerung nach Abschluß eines Nebenprogramms. Soll ein Programm ein Nebenprogramm ausführen und die Steuerung danach zum Hauptprogramm zurückkehren, muß das Programm die Abschlußadresse vor Erstellung des Segments setzen, in welches das Nebenprogramm geladen wird. Anderenfalls wird bei Abschluß des Nebenprogramms auch das Hauptprogramm beendet. Diese Unterbrechung bestimmt nicht die Abschlußadresse des Aufrufprogramms, sondern nur die der Aufgaben, die durch dieses Programm aufgerufen wurden.

23 UMSCH + ANNULLIER Ausgangsadresse. Wird UMSCH + ANNULLIER während Tastatureingabe oder Videoausgabe gedrückt, erscheint ÜC an der Konsole, und eine 23H Unterbrechung wird ausgeführt. Bei Ausführung von Funktion 9H oder 0AH (gepufferte Ein- und Ausgabe) wird E/A am Anfang der Zeile fortgesetzt. Bei Unterbrechung enthalten alle Register den Wert, den sie z. Z. des ursprünglichen Aufrufs an DOS hatten. Wird keine normale UMSCH + ANNULLIER Programmbeendung gewünscht, kann das Programm auf eine besondere Routine zeigen. Behält diese UMSCH + ANNULLIER Routine alle Register bei, kann sie mit einer IRET-Instruktion (Rückkehr nach Unterbrechung) enden, um den Programmablauf fortzusetzen. Das UMSCH + ANNULLIER Bearbeitungsprogramm kann ohne Beschränkung alles, einschließlich DOS Funktionsaufrufe, ausführen, solange sich die Register bei Anwendung von IRET nicht ändern.

Erstellt das Programm ein neues Segment und lädt dann ein Nebenprogramm, das die UMSCH + ANNULLIER Adresse ändert, wird bei Beendung des Nebenprogramms und Rückkehr zum Hauptprogramm der ursprüngliche Wert der UMSCH + ANNULLIER Adresse wiederhergestellt.

24 Abbruchfehler. Tritt ein Abbruchfehler innerhalb von DOS auf, wird die Steuerung mit INT 24H übertragen. Bei Eintritt in das Fehlerbearbeitungsprogramm zeigt BP:SI auf den Gerätevorsatz des Gerätes, das den Fehler verursacht (Gerätevorsatzformat - siehe Anhang E).

Bei einem Plattenfehler (häufigste Ursache) ist das AH-Bit 7=0. Bit 7=1 bedeutet Zeichengerätfehler oder falsche Wiedergabe der FAT. Handelt es sich um einen Plattenfehler, umfassen Bits 0-2 folgendes:

Bit	0		0 Lese-, 1 Schreibvorgang
Bit	2	1	BETROFFENER DISKBEREICH
	0	0	reservierter Bereich
	0	1	Dateizuordnungstabelle
	1	0	Verzeichnis
	1	1	Datenbereich

AL, CX, DX und DS:BX werden für einen erneuten Übertragungsversuch mit INT 25H oder INT 26H gesetzt. Die Hardware setzt DI auf einen 16-Bit-Fehlercode.

Fehlercodes:

0	Schreibgeschützt
1	Unbekanntes Gerät
2	Laufwerk nicht bereit
3	Unbekannter Befehl
4	Datenfehler
5	Ungültige Länge der Laufwerkanforderungs-Struktur
6	Fehlersuche
7	Unbekannter Datenträger
8	Sektor nicht gefunden
9	Kein Papier im Drucker
A	Schreibfehler
B	Lesefehler
C	Allgemeines Diskversagen

Die Register sind für einen BIOS-Diskaufruf gesetzt, und der Fehlercode ist in der unteren Hälfte des DI-Registers, während die obere Hälfte unbestimmt bleibt. Das Fehlerbearbeitungsprogramm kann nur die Funktionsaufrufe 1 bis 12 durchführen. Alle anderen Aufrufe zerstören den DOS Stapelspeicher.

Anwender-Stapelspeicher (von oben):

SP
CS
FLAGS
AX
BX
CX
DX
SI
DI
BP
DS
ES
IP
CS
FLAGS

Wird IRET ausgeführt, reagiert DOS auf AL folgendermaßen:

(AL)=0	Fehler ignorieren
=1	Vorgang erneut versuchen (BEI DIESER OPTION DÜRFEN STAPEL DS, ES, BX, CX UND DX NICHT MODIFIZIERT WERDEN.)
=2	Programm abbrechen

25 Absoluter Disk-Lesevorgang. Die Steuerung wird direkt an DOS BIOS übertragen. Für diesen Eintrittspunkt sind Sätze und Sektoren von gleicher Größe. Anforderung:

(AL)	Laufwerknummer (0 = A, 1 = B usw.)
(CX)	Anzahl zu lesender Sektoren
(DX)	Anfang logischer Satznummer
(DS:BX)	Übertragungsadresse

Die Routine überträgt die angegebene Satzanzahl von dem bestimmten Laufwerk auf die Übertragungsadresse. Logische Satznummern werden geschaffen, indem jeder Sektor sequentiell ab Null und über Spurgrenzen hinaus numeriert wird. Auf Disketten mit neun Sektoren pro Spur ist die logische Satznummer 0 z. B. Spur 0, Sektor 1, während die logische Satznummer 12H Spur 2, Sektor 0, ist.

ANMERKUNG:
Bei Rückkehr sind die ursprünglichen Kennzeichen noch auf dem Stapel (aufgrund der INT-Instruktion). Dies ist notwendig, weil die Kennzeichen Rückkehrinformation enthalten. Es ist wichtig, den Stapel hervorzuholen, um unkontrolliertes Wachsen zu verhindern.

Dieser Aufruf zerstört alle Register mit Ausnahme der Segmentregister. Bei erfolgreicher Übertragung ist das Übertragkennzeichen (CF) Null. Bei nicht erfolgreicher Übertragung zeigen CF=1 und (AL) den Fehler folgendermaßen an:

Anzeige	Beschreibung
0	Schreibversuch auf schreibgeschützter Disk
2	Disk nicht bereit
4	Datenfehler
6	Fehlersuche
8	Sektor nicht gefunden
A	Allgemeines Diskversagen
C	Schreibfehler

26 Absoluter Disk-Schreibvorgang. Dieser Vektor ist das Gegenstück zu Unterbrechung 25 (siehe vorhergehende Beschreibung).

27 Abschließen, jedoch resident bleiben. Dieser Vektor wird für Programme benutzt, die resident bleiben sollen, nachdem BEFEHL wieder die Steuerung übernimmt. BEFEHL lädt ein solches Programm wie eine ausführende .COM Datei. Nach Initialisierung muß das Programm DX auf die der letzten Adresse folgende Adresse in dem Segment setzen, in dem es ausführt, und dann eine 27H Unterbrechung ausführen. BEFEHL betrachtet dann das Programm als DOS Erweiterung und überlagert es nicht bei Ausführung anderer Programme.

C.2 FUNKTIONSAUFRUFE

Eine Funktion wird durch Setzen einer Funktionszahl im AH-Register aufgerufen. Falls notwendig wird zusätzliche Information für eine spezifische Funktion in anderen Registern angegeben und eine 21H Unterbrechung ausgeführt. Wenn DOS die Steuerung übernimmt, wechselt es auf einen internen Stapel über. Mit Ausnahme von AX bleiben Anwender-Register erhalten, es sei denn, in spezifischen Aufrufen wird etwas anderes angegeben. Der Anwenderstapel muß zur Aufnahme des Unterbrechungssystems groß genug sein. Der Anwenderstapel sollte um 80H größer gestaltet werden, als vom Progamm benötigt wird.

MS DOS Versionen vor 2.0 führen Funktionsaufrufe 0 bis 2E aus. Diese Aufrufe sind mit 2.0 oder späteren Versionen kompatibel. Ausnahmen sind:

Programme, die mit Funktionen 27 oder 28 einen Anzeiger zur FAT geben, können nicht mehr benutzt werden. Weil FAT jetzt mit anderen Disk-Resourcen gesondert verarbeitet wird, gibt es dafür keine feste Stelle im Speicher. Funktionen 27 und 28 bestehen noch, können aber nur für das FAT-ID-Byte benutzt werden. Bei Rückkehr von 27 oder 28 zeigt ES:BX auf ein FAT-ID-Byte für das aktuelle Laufwerk.

ANMERKUNG:
Wird 27 und 28 für andere Zwecke als das Lesen dieses einen Bytes benutzt, ist Systemausfall möglich.

MS DOS Version 2.0 und darauffolgende Versionen führen die übrigen Funktionsaufrufe aus. Einige der 2.0 Aufrufe duplizieren Funktionsaufrufe vor Version 2.0, bieten aber zusätzliche Funktionsfähigkeit. Diese zusätzliche Funktionsfähigkeit läßt späteren Ausbau des Betriebssystems zu. Deshalb ist es ratsam, 2.0 Aufrufe zu benutzen, wenn die Möglichkeit dazu besteht.

Die E/A-Steuerung bei 2.0 und späteren Versionen benutzt einen 16-Bit-Dateibezeichner, der dem Gerät oder der Datei bei Eröffnung vom System zugeteilt wird. Nachfolgende Datei- oder Gerät-E/A-Funktionen müssen sich auf diesen Bezeichner beziehen. Funktionsaufrufe 3C, 3D und 45 senden den Dateibezeichner in AX. Standard-Dateibezeichner sind:

0 = Standard-Eingabegerät
1 = Standard-Ausgabegerät
2 = Standard-Fehlermeldungsgerät
3 = Standard-Zusatzgerät
4 = normaler Drucker

Bei Versionen vor 2.0 wurde Funktionsaufruf 6 (Direkte E/A-Konsole) zum Schreiben an den Bildschirm benutzt. Bei Versionen nach 2.0 sollte zum Schreiben an den Bildschirm Funktionsaufruf 40 (Datei schreiben) mit dem Standardausgabe-Dateibezeichner (1) benutzt werden.

Ein weiterer bedeutender Unterschied zwischen E/A-Funktionsaufrufen der Version 2.0 und früheren Versionen ist, daß die E/A-Datei 2.0 Verzeichnispfade benutzt, während Aufrufe vor 2.0 nur Dateinamen und Erweiterungen benutzen (siehe Wang Professional Computer Einführungshandbuch, Verzeichnispfade).

Mehrere der folgenden Beschreibungen beziehen sich auf ASCIZ Zeichenfolgen. Eine ASCIZ-Folge ist eine Reihe von ASCII-Zeichen, in der 0H das Folgenende darstellt.

Funktionsaufrufe ab 39 sind mit dem XENIX Betriebssystem kompatibel. XENIX ist ein Multitasking-Betriebssystem; Gebrauch dieser Aufrufe ermöglicht Upgrading zu diesem System.

Bei 2.0-Aufrufen erhält das AX-Register Fehlercodes und Rückkehrwerte. Fehlercode-Definitionen:

0 = Kein Fehler
1 = Fehler - Funktion unzulässig
2 = Fehler - Datei nicht gefunden
3 = Fehler - Pfad nicht gefunden
4 = Fehler - Zu viele eröffnete Dateien
5 = Fehler - Zugang verweigert
6 = Fehler - Bezeichner ungültig
7 = Fehler - Speicherblock beschädigt
8 = Fehler - Speicherraum nicht ausreichend
9 = Fehler - Block ungültig
A = Fehler - Umgebung ungültig (siehe Funktionsaufruf 4B)
B = Fehler - Format ungültig
C = Fehler - Zugang ungültig
D = Fehler - Daten ungültig
F = Fehler - Laufwerk ungültig
10 = Fehler - Aktuelles Verzeichnis
11 = Fehler - Nicht dasselbe Gerät
12 = Fehler - Keine weiteren Dateien

ANMERKUNG:
Wenn der Wert in AX entweder einen Fehlercode oder Rückkehrwert darstellen kann, handelt es sich bei gesetztem Übertrag-Kennzeichen um einen Fehlercode, sonst um einen Rückkehrwert.

Die Funktionsaufrufe werden nachstehend alphabetisch aufgelistet. Bei Funktionsdoppeln in Version 2.0 und früheren Versionen zeigt die Aufstellung, zu welcher Kategorie der Aufruf gehört.
DOS Unterbrechungen und Funktionsaufrufe

19	Aktuelle Disk
2F	Aktuelle DTA aufgreifen
47	Aktuelles Verzeichnis
3B	Aktuelles Verzeichnis ändern
2	Bildschirmausgabe
45	Dateibezeichner duplizieren
3D	Datei eröffnen (2.0)
F	Datei eröffnen (1.25)
3C	Datei erstellen (2.0)
16	Datei erstellen (1.25)
3F	Datei/Gerät lesen (2.0)
23	Dateigröße
42	Datei Lese-/Schreibanzeiger verschieben
13	Datei löschen
29	Dateinamen analysieren
3E	Datei schließen (2.0)
10	Datei schließen (1.25)
40	Datei schreiben (2.0)
17	Datei umbenennen (1.25)
57	Datei-Zeitangabe
2A	Datum aufgreifen
2B	Datum setzen
6	Direkte Konsolen-E/A
7	Direkte Konsolen-Eingabe
1A	Disk-Übertragungsadresse setzen
E	Disk wählen
D	Disk zurücksetzen
46	Doppel eines Bezeichners erzwingen
34	DOS Kritisches Abschnittskennzeichen aufgreifen
30	DOS Versions-Nummer aufgreifen
5	Druckerausgabe
44	E/A Gerätesteuerung
11	Erste Eintragung suchen
4E	Erste finden
9	Folge drucken
A	Gepufferte Tastatureingabe
37	Inkompatible Konfigurationsparameter ändern
38	Internationale Information
8	Konsolen-Eingabe ohne Rückmeldung
33	KTRL + C Auffang setzen oder aufgreifen
36	Leeren Diskraum aufgreifen
12	Nächste Eintragung suchen
4F	Nächste finden
26	Neues Programmsegment erstellen
4B	Programm ausführen
31	Prozedur beibehalten
4C	Prozedur verlassen
54	Prüfkennzeichen aufgreifen
4D	Rückkehrcode einer Nebenprozedur zurückholen
43	Schreibschutz ändern
14	Sequentielles Lesen (1.25)
15	Sequentielles Schreiben (1.25)
2E	Set/Reset-Prüfkennzeichen setzen

48	Speicherraum zuordnen
1	Tastatureingabe
B	Tastaturstatus prüfen
2C	Uhrzeit aufgreifen
2D	Uhrzeit setzen
56	Umbenennen (2.0)
35	Unterbrechungsvektor aufgreifen
25	Unterbrechungsvektor setzen
41	Verzeichniseintrag löschen
3A	Verzeichnis entfernen
39	Verzeichnis erstellen
C	Zeicheneingabe mit Pufferleerung
21	Zufallsablesen (1.25)
27	Zufallsblock ablesen
28	Zufallsblock schreiben
24	Zufallssatzfeld setzen
22	Zufallsschreiben (1.25)
4A	Zugewiesenen Block modifizieren
49	Zugewiesenen Speicher freigeben
4	Zusatzausgabe
3	Zusatzeingabe

Abschnitt C.2.1 und C.2.2 führen die Funktionsaufrufe numerisch auf und enthalten Erklärungen. Abschnitt C.2.1 listet Funktionsaufrufe vor Version 2.0 und Abschnitt C.2.2 Aufrufe der Version 2.0. Alle Werte sind hexadezimal.

C.2.1 Funktionsaufrufe vor Version 2.0

0 Programmabschluß. Die Ausgangsadressen Abschluß und UMSCH + ANNULLIER enthalten die Werte, die sie bei Eintritt in das beendende Programm hatten. Diese Routine leert alle Dateipuffer, setzt jedoch Verzeichnisse für nicht geschlossene Dateien, deren Länge sich geändert hat, nicht korrekt fest. Die Steuerung wird an die Endadresse übertragen.

1 Tastatureingabe. Die Routine wartet auf Eintragung eines Zeichens an der Tastatur, zeigt es am Bildschirm an und sendet es dann in AL. Es wird geprüft, ob das Zeichen ein KTRL + C (UMSCH + ANNULLIER) ist. Ist dies der Fall, führt die Routine Unterbrechung 23 aus.

2 Bildschirmausgabe. Die Routine sendet das Zeichen in DL zum Bildschirm. Tritt nach der Ausgabe ein UMSCH + ANNULLIER auf, führt die Routine Unterbrechung 23 aus.

3 Zusatzeingabe. Die Routine wartet auf ein Zeichen vom Zusatzeingabegerät und sendet es dann in AL.

4 Zusatzausgabe. Die Routine sendet das Zeichen in DL zum Zusatzgerät.

5 Druckerausgabe. Die Routine sendet das Zeichen in DL zum Drucker.

6 Direkte Konsolen-E/A. Ist FF in DL enthalten, kehrt AL mit dem Tastatur-Eingabezeichen zurück, falls eines bereit ist; sonst kehrt AL mit 00 zurück. Wenn DL nicht FF ist, setzt die Routine voraus, daß DL ein zulässiges Zeichen hat, das zum Bildschirm geschickt wird.

7 Direkte Konsolen-Eingabe. Die Routine wartet auf Eintragung eines Zeichens an der Tastatur und sendet es dann in AL. Diese Funktion ist Funktion 6 ähnlich, doch wird das Zeichen nicht auf UMSCH + ANNULLIER überprüft.

8 Konsolen-Eingabe ohne Rückmeldung. Diese Funktion ist Funktion 1 ähnlich, jedoch erscheint das eingetastete Zeichen nicht auf dem Bildschirm.

9 Folge drucken. Bei Eintritt muß DS:DX auf eine Zeichenfolge im Speicher zeigen, die durch ein $ (24) abgeschlossen ist. Jedes Zeichen in der Folge erscheint am Bildschirm in derselben Form wie bei Funktion 2.

A Gepufferte Tastatureingabe. Bei Eintritt zeigt DS:DX auf einen Eingabepuffer. Das erste Byte gibt an, wieviele Zeichen im Puffer enthalten sein können; die Zahl darf nicht Null sein. Die Routine liest Zeichen von der Tastatur und positioniert sie im Puffer ab dem dritten Byte. Der Computer setzt den Tastatur-Ablesevorgang und das Füllen des Puffers fort, bis Zeilenschaltung gedrückt wird. Füllt sich der Puffer um eins weniger als das Maximum, wird weitere Tastatureingabe ignoriert, bis Zeilenschaltung gedrückt wird. Die Routine setzt das zweite Byte des Puffers gemäß der Anzahl der erhaltenen Zeichen, ausschließlich Wagenrücklauf (0D), der immer das letzte Zeichen ist.

B Tastaturstatus prüfen. AL ist FF, wenn ein Zeichen von der Tastatur verfügbar ist; sonst ist AL 00.

C Zeicheneingabe mit Pufferleerung. Die Routine leert zuerst den Tastatur-Type-Ahead-Puffer. Falls AL 1, 6, 7, 8 oder 0A ist, wird die entsprechende DOS Eingabefunktion ausgeführt. Hat AL keinen dieser Werte, findet kein weiterer Vorgang statt, und AL sendet 00.

D Disk zurücksetzen. Die Routine leert alle Dateipuffer. Dateien, deren Größe sich verändert hat, werden erst ordnungsgemäß im Verzeichnis eingetragen, wenn sie geschlossen sind. Diese Funktion braucht nicht vor Diskwechsel aufgerufen zu werden, wenn alle Dateien, an die geschriebenen wurde, schon abgeschlossen sind.

E Disk wählen. Das in DL angegebene Laufwerk (0 = A, 1 = B usw.) wird zum Standardlaufwerk. AL sendet die Laufwerkanzahl.

F Datei eröffnen. Bei Eintritt zeigt DS:DX auf einen ungeöffneten Dateisteuerblock (FCB). Die Routine sucht in den Verzeichnissen nach der genannten Datei und setzt AL auf FF, falls sie die Datei nicht findet. Findet die Routine die Datei, setzt sie AL auf 00 und füllt den FCB auf folgende Art:

Falls der Laufwerkcode 0 ist (Standarddisk), ändert die Routine diesen auf den Code der tatsächlich benutzten Disk ab (A=1, B=2 usw.). Dies ermöglicht das Wechseln der Standarddisk, ohne spätere Vorgänge in dieser Datei zu beeinträchtigen. Die Routine setzt das höchstwertige Byte des aktuellen Blockfeldes. FCB Bytes 0E-0F (Größe des zu bearbeitenden Satzes) enthalten den System-Standardwert 80. Die Routine setzt aufgrund von Information aus dem Verzeichnis Dateigröße, Zeit und Datum im FCB fest.

Die Satzgröße (FCB Bytes 0E-0F) muß wie gewünscht festgelegt werden, falls der Standardwert 80 ungeeignet ist. Auch muß das Zufallssatzfeld und/oder aktuelle Block- und Satzfelder gesetzt werden.

10 Datei schließen. Diese Funktion ist nach Datei-Schreibvorgängen notwendig, um sicherzustellen, daß alle Verzeichnis-Informationen aktualisiert sind. Bei Eintritt zeigt DS:DX auf einen eröffneten FCB. Die Routine sucht die Verzeichnisse ab und vergleicht, wenn sie die Datei findet, deren Position mit der im FCB. Findet die Routine die Datei nicht, wird Diskwechsel vorausgesetzt, und AL sendet FF. Anderenfalls ändert die Routine das Verzeichnis auf den im FCB aufgezeichneten Dateistatus ab, und AL sendet 00.

11 Erste Eintragung suchen. Bei Eintritt zeigt DS:DX auf einen ungeöffneten FCB. Die Routine sucht im Verzeichnis nach dem ersten gleichwertigen Namen ("?" im Namen könnten auf gleichwertige Zeichen hinweisen). Wird kein Gleichwert gefunden, sendet AL FF. Anderenfalls setzt die Routine Stellen an der Disk-Übertragungsadresse:

1. Ist der zur Suche angegebene FCB ein erweiterter FCB, setzt die Routine das erste Byte auf FF, 5 Bytes auf Null, ein Byte gemäß dem Attribut-Byte vom Such-FCB, ein Byte gemäß der benutzten Laufwerknummer (A=1, B=2 usw.) und dann 32 Bytes gemäß der Verzeichniseintragung. Die Disk-Übertragungsadresse enthält auf diese Weise einen zulässigen, ungeöffneten erweiterten FCB mit denselben Suchattributen wie der Such-FCB.

2. Ist der zur Suche angegebene FCB ein gewöhnlicher FCB, setzt die Routine das erste Byte gemäß der benutzten Laufwerknummer (A=1, B=2 usw.), und die nächsten 32 Bytes enthalten die entsprechende Verzeichniseintragung. Auf diese Weise enthält die Disk-Übertragungsadresse einen zulässigen, ungeöffneten gewöhnlichen FCB.

   Siehe Abschnitt A.4.5, Diskverzeichnis.

12 Nächste Eintragung suchen. Nachdem Funktion 11 einen Gleichwert gefunden hat, wird bei Aufruf an Funktion 12 der nächste Gleichwert zu einer unklaren Anfrage gefunden ("?" im Suchdateinamen). Ein- und Ausgaben sind dieselben wie bei Funktion 11. Der reservierte Bereich

des FCBs enthält Informationen, die zur Fortsetzung des Suchvorgangs notwendig sind; deshalb darf der reservierte Bereich nicht modifiziert werden.

13 Datei löschen. Bei Eintritt zeigt DS:DX auf einen ungeöffneten FCB. Diese Funktion löscht alle gleichwertigen Verzeichniseintragungen. Wenn keine Verzeichniseintragungen übereinstimmen, sendet AL FF, anderenfalls 00.

14 Sequentielles Lesen. Bei Eintritt zeigt DS:DX auf einen geöffneten FCB. Die Routine lädt den vom aktuellen Block adressierten Satz (FCB Bytes C-D) und den aktuellen Satz (FCB Byte 1F) an der Disk-Übertragungsadresse und inkrementiert dann die Satzadresse. Bei Auftritt eines Dateiendes sendet AL entweder 01 oder 03. Rückgabe von 01 zeigt an, daß keine Daten im Satz waren. 03 zeigt einen teilweisen, mit Nullen ausgefüllten Satz-Lesevorgang. Rückgabe von 02 bedeutet, daß zum Lesen eines Satzes nicht genügend Platz im Disk-Übertragungssegment war, und die Übertragung daher abgebrochen wurde. AL sendet 00, wenn die Übertragung erfolgreich abgeschlossen ist.

15 Sequentielles Schreiben. Bei Eintritt zeigt DS:DX auf einen eröffneten FCB. Die Routine schreibt den vom aktuellen Block adressierten Satz und die aktuellen Satzfelder von der Disk-Übertragungsadresse. (Die Routine puffert Sätze, die weniger als einen Sektor einnehmen, für späteres Schreiben, wenn sich genug Daten für einen Sektor angesammelt haben.) Die Satzadresse wird dann inkrementiert. AL sendet 01, wenn die Disk gefüllt ist. Rückgabe von 02 bedeutet, daß nicht genug Platz zum Schreiben eines Satzes im Disk-Übertragungssegment war und die Übertragung daher abgebrochen wurde. AL sendet 00, wenn der Übergang erfolgreich abgeschlossen ist.

16 Datei erstellen. Bei Eintritt zeigt DS:DX auf einen ungeöffneten FCB. Die Routine sucht im Verzeichnis nach einer leeren Eintragung. Findet sie keine, sendet AL FF. Anderenfalls initialisiert die Routine die Eintragung auf eine Null-Länge-Datei und eröffnet die Datei (siehe Funktion F). AL sendet 00.

17 Datei umbenennen. Bei Eintritt zeigt DS:DX auf einen modifizierten FCB mit Laufwerkcode und Dateinamen an der üblichen Stelle und einem zweiten Dateinamen, der 6 Bytes nach dem ersten (DS:DX+11) in einem sonst reservierten Bereich beginnt. Die Routine ändert jedes Zeichen im ersten Dateinamen auf das Zeichen in der entsprechenden Position im zweiten ab (mit der Einschränkung, daß zwei Dateien nicht denselben Namen und dieselbe Erweiterung tragen können). Wenn "?" im zweiten Namen erscheinen, bleiben die entsprechenden Positionen im ursprünglichen Namen unverändert. AL sendet FF, falls kein Gleichwert gefunden wurde, anderenfalls 00.

19 Aktuelle Disk. AL kehrt mit dem Standard-Laufwerkcode zurück (0 = A, 1 = B usw.).

1A Disk-Übertragungsadresse setzen. Dieser Aufruf setzt die Disk-Übertragungsadresse auf DS:DX. DOS läßt bei Disküübertragung weder "Wraparound" innerhalb des Segments noch Überlauf in das nächste Segment zu.

21 Zufallsablesen. Bei Eintritt zeigt DS:DX auf einen eröffneten FCB. Die Routine setzt den aktuellen Block und Satz übereinstimmend mit dem Zufallssatzfeld und lädt dann den durch diese Felder adressierten Satz an der aktuellen Disk-Übertragungsadresse. Bei einem Dateiende sendet AL entweder 01 oder 03. 01 bedeutet, daß keine weiteren Daten verfügbar sind, 03, daß ein teilweiser, mit Nullen ausgefüllter Satz verfügbar ist. Bei Rückgabe von 02 war zur Ablesung eines Satzes nicht genügend Platz im Disk-Übertragungssegment; die Übertragung wurde daher abgebrochen. AL sendet 00, wenn die Übertragung erfolgreich abgeschlossen ist.

22 Zufallsschreiben. Bei Eintritt zeigt DS:DX auf einen eröffneten FCB. Die Routine setzt den aktuellen Block und Satz übereinstimmend mit dem Zufallssatzfeld und schreibt dann den durch diese Felder adressierten Satz von der Disk-Übertragungsadresse (gepuffert bei Sätzen, deren Größe nicht mit Sektorgrößen übereinstimmt). Ist die Disk gefüllt, sendet AL 01. Rückgabe von 02 bedeutet, daß nicht genug Platz zum Schreiben eines Satzes im Disk-Übertragungssegment war und die Übertragung deshalb abgebrochen wurde. AL sendet 00, wenn die Übertragung erfolgreich abgeschlossen ist.

23 Dateigröße. Bei Eintritt zeigt DS:DX auf einen ungeöffneten FCB. Die Routine sucht das Verzeichnis nach der ersten gleichwertigen Eintragung ab. Wird keine gefunden, sendet AL FF. Anderenfalls setzt die Routine das Zufallssatzfeld mit der Dateigröße (Satzgrößenfeld aufgerundet), und AL sendet 00.

24 Zufallssatzfeld setzen. Bei Eintritt zeigt DS:DX auf einen eröffneten FCB. Diese Funktion setzt das Zufallssatzfeld auf die gleiche Dateiadresse wie das aktuelle Block- und Satzfeld.

25 Unterbrechungsvektor setzen. Diese Funktion setzt die in AL angegebene Unterbrechungsart auf die 4-Byte-Adresse DS:DX.

26 Neues Programmsegment erstellen. Bei Eintritt hat DX eine Segmentnummer, an die ein neues Programm gesetzt wird. Die Routine kopiert den gesamten 100-Byte-Bereich an der Nullstelle im aktuellen Programmsegment auf die Nullstelle im neuen Programmsegment. Dann aktualisiert die Routine die Speichergrößen-Information an Stelle 6 und speichert die aktuellen Abschluß- und UMSCH + ANNULLIER-Ausgangsadressen im neuen Programmsegment, das bei OA beginnt.

27 Zufallsblock ablesen. Bei Eintritt zeigt DS:DX auf einen eröffneten FCB, und CX enthält eine Satzanzahl, die nicht Null sein darf. Die Routine liest die angegebene Satzanzahl (gemäß dem Satzgrößenfeld) von der im Zufallssatzfeld angegebenen Dateiadresse in die Disk-Übertragungsdresse. Erscheint ein Dateiende, bevor alle Sätze gelesen sind, sendet AL entweder 01 oder 03. Rückgabe von 01 zeigt das Dateiende an; die letzte Ablesung ist ein vollständiger Satz. 03 zeigt an, daß die letzte Ablesung ein Teilsatz ist. Tritt "Wrap-around" oberhalb von Adresse FFFF im Disk-Übertragungssegment auf, liest die Routine so viele Sätze wie möglich, und AL sendet 02. Wenn AL 00 sendet, wurden alle Sätze erfolgreich gelesen. CX kehrt auf

jeden Fall mit der Anzahl der tatsächlich gelesenen Sätze zurück; das Zufallssatzfeld und die aktuellen Block-/Satzfelder erhalten die Adresse des nächsten Satzes.

28 Zufallsblock schreiben. Diese Funktion, die Funktion 27 ähnelt, ist eine Schreibfunktion, bei welcher eine Schreibschutz-Bezeichnung benutzt wird. Reicht der Diskraum nicht aus, sendet AL 01, und der Schreibvorgang findet nicht statt. Wenn CX bei Eintritt Null ist, findet kein Schreibvorgang statt, die Routine setzt jedoch die Datei gemäß der im Zufallssatzfeld angegebenen Länge (gleichgültig ob länger oder kürzer als die aktuelle Datei) und weist Zuordnungseinheiten zu oder stellt sie nach Bedarf frei.

29 Dateinamen analysieren. Bei Eintritt zeigt DS:SI auf eine zu analysierende Befehlszeile; ES:DI zeigt auf einen Speicherteil, der einen ungeöffneten FCB empfangen soll. Der Abtaster ignoriert führende Tabs und Leerstellen. Wenn bei Eintritt Bit 0 von AL gleich 1 ist, ignoriert der Abtaster höchstens ein führendes Dateinamen-Trennsymbol einschließlich abschließender Tabs und Leerstellen. Die vier Dateinamen-Trennsymbole sind:

```
; , = +
```

Ist Bit 0 von AL gleich 0, bricht der Analysiervorgang bei Auftritt eines Trennsymbols ab. Die Routine sucht die Befehlszeile nach dem Dateinamen in der Form d:Filename.Ext. ab. Bei Auftritt setzt sie einen entsprechenden ungeöffneten FCB bei ES:DI. Der Wert der AL Bits 1, 2 und 3 bestimmt bei Eintritt, welche Maßnahmen zu treffen sind, falls Laufwerk, Dateiname bzw. Erweiterung fehlen. Ist das Bit Null und die Datei nicht in der Befehlszeile, erhält der FCB in allen Fällen einen festen Wert (0, Standardlaufwerk für das Laufwerkfeld; alle Leerstellen für das Dateinamen- und Erweiterungsfeld). Ist das Bit 1 und das Feld nicht in der Befehlszeile, ändert sich das Feld im Bestimmungsort-FCB bei ES:DI nicht. Befindet sich ein Stern (*) im Dateinamen oder in der Erweiterung, setzt die Routine alle übrigen Zeichen im Namen oder in der Erweiterung auf "?".

Folgende Zeichen sind in DOS Dateispezifikationen unzulässig:

```
" / [ ] + = ; ,
```

Steuerzeichen und Leerstellen erscheinen u. U. nicht als Elemente von Dateispezifikationen. Trifft der Abtaster auf eines dieser Zeichen oder den Punkt (.) oder Doppelpunkt (:) an einer unzulässigen Position, bricht der Analysiervorgang an dieser Stelle ab.

Tritt ein "?" oder "*" im Dateinamen bzw. in der Erweiterung auf, sendet AL 01, andernfalls 00. DS:SI kehrt zurück und zeigt auf das erste Zeichen nach dem Dateinamen.

2A Datum aufgreifen. Sendet das Datum in CX:DX. CX enthält das Jahr, DH den Monat (1=Jan, 2=Feb usw.) und DL den Tag. Wechselt der Uhrzeitgeber auf den nächsten Tag über, ändert sich das Datum entsprechend; die Anzahl der Tage in jedem Monat und Schaltjahre werden berücksichtigt.

2B Datum setzen. Bei Eintritt muß CX:DX ein zulässiges Datum mit dem gleichen Format tragen, wie es von Funktion 2A gesendet wird. Bei zulässigem Datum und erfolgreicher Festsetzung sendet AL 00. Ist das Datum ungültig, sendet AL FF.

2C Uhrzeit aufgreifen. Sendet die Tageszeit in CX:DX. Die Zeit wird durch vier 8-Bit-Binärgrößen folgendermaßen dargestellt: CH enthält Stunden (0-23), CL Minuten (0-59), DH Sekunden (0-59) und DL 1/100 Sekunden (0-99). Dieses Format läßt sich leicht zu einer druckbaren Form konvertieren; es eignet sich aber auch zum Kalkulieren, z. B. subtrahieren zweier Zeiten usw.).

2D Uhrzeit setzen. Bei Eintritt enthält CX:DX die Zeit im selben, wie von Funktion 2C gesendeten Format. Ist ein Zeitelement unzulässig, bricht der Festsetzvorgang ab, und AL sendet FF. Bei zulässiger Zeit sendet AL 00.

C.2.2 Funktionsaufrufe der Version 2.0

2E Set/Reset-Prüfkennzeichen setzen. Bei Eintritt muß DL 0 sein, und AL enthält das Prüfkennzeichen: 0 = nicht prüfen, 1 = nach Schreiben prüfen. Das E/A-System prüft dieses Kennzeichen bei jedem Schreibvorgang und deutet es entweder als "nur Schreiben" oder "nach Schreiben prüfen".

2F Aktuelle DTA aufgreifen. Die Disk-Übertragungsadresse wird in ES:BX gesendet.

30 DOS Versions-Nummer aufgreifen. Bei Rückkehr enthält AL-AH die zweiteilige Versionsbezeichnung. Deshalb ist z. B. bei MS-DOS 1.28 AL 1 und AH 28. Bei Versionen vor 1.28 ist AL 0. BH enthält einen Hersteller-Code, der für Wang 1 ist.

31 Prozedur beibehalten. Diese Routine ermöglicht es, eine Prozedur abzuschließen, und diese im Speicher beizubehalten. Bei Eintritt gibt der Anwender einen Ausgangscode in AL ein. DX enthält die in Absätzen angegebene Größe des Blocks, der zum Speichern des Ablaufs zugewiesen wurde. Der Block kann entweder kleiner oder so groß wie der ursprüngliche Zuordnungsblock der Prozedur sein. Die Routine gibt keine anderen zur Prozedur gehörenden Zuordnungsblöcke frei. Bei Rückkehr hat AX den vom Anwender angegebenen Ausgangscode. Dieser Code kann durch den Warte-Funktionsaufruf (4D) wieder zurückgeholt werden.

33 KTRL + C Auffang setzen oder aufgreifen. Mit dieser Routine kann KTRL + C Auffang für alle Funktionsaufrufe (in 1 bis 12 bereits einbegriffen) bestimmt werden. Es kann auch festgestellt werden, ob KTRL + C Auffang für alle Aufrufe gesetzt ist. Bei Eintritt enthält DL 0 (KTRL + C Auffang ausgeschaltet) oder 1 (KTRL + C Auffang eingeschaltet). AL hat einen Subfunktionscode: 0 = KTRL + C Auffang aufgreifen, 1 = KTRL + C Auffang setzen. Wenn AL bei Eintritt 0 enthält, hat DL bei Rückkehr 0 für KTRL + C Auffang (ausgeschaltet) oder 1 für KTRL + C Auffang (eingeschaltet).

34 DOS Kritisches Abschnittskennzeichen aufgreifen. In einem Unterbrechungs-Dienstprogramm zeigt diese Routine an, ob DOS in einem kritischen Abschnitt unterbrochen wurde. Bei Rückkehr ist ES:BX die Adresse einer Bytespeicherzelle in DOS. Ist das Byte Null, befindet sich DOS nicht in einem kritischen Abschnitt und kann durch die Unterbrechungsroutine aufgerufen werden. Ist das Byte nicht Null, sollte DOS als ununterbrechbar angesehen werden. DOS Aufrufe sollten aus Sicherheitsgründen nicht erteilt werden.

35 Unterbrechungsvektor aufgreifen. Bei Eintritt enthält AL die Nummer einer Unterbrechung. Diese Routine sendet die Adresse der Unterbrechungsroutine in ES:BX.

36 Leeren Diskraum aufgreifen. Bei Eintritt enthält DL das Plattenlaufwerk: 0 = Standardwert, 1 = A, 2 = B, 3 = C. Bei Rückkehr enthält BX die Anzahl freier Zuordnungseinheiten auf der Disk. DX enthält die Gesamtanzahl der Zuordnungseinheiten, CX die Anzahl der Bytes pro Sektor und AX die Sektoranzahl pro Zuordnungseinheit. Bei ungültiger Laufwerknummer wird AX auf -1 gesetzt.

37 Inkompatible Konfigurationsparameter ändern. Die Routine gibt einen Subfunktionscode in AL ein:

- 0 Bei Rückkehr enthält DL das DOS-Schaltzeichen. Der Standardwert ist '-'.
- 1 Das Schaltzeichen wird auf ein in DL eingegebenes Zeichen gesetzt.
- 2 DL wird auf das Geräteverfügbarkeits-Byte gesetzt. Ist dieses Byte 0, muß auf Geräte in Datei-E/A-Aufrufen durch Codierung von /Gerät/<Gerät> zugegriffen werden. Wenn dieses Byte nicht Null ist, sind die Geräte auf jeder Stufe der Verzeichnis-Hierarchie verfügbar (z. B. ist CON das Konsolengerät, nicht die Datei CON).
- 3 Das Geräteverfügbarkeits-Byte wird auf den in DL angegebenen Wert gesetzt.

Liegt der Subfunktionscode außerhalb des Bereichs, enthält AL bei Rückkehr FF.

38 Internationale Information. Bei Eintritt enthält Al einen geographischen Code: 0 = USA, 1 = Europa, 3 = Japan. Im Speicherblock, der von DS:DX adressiert ist, sendet diese Routine folgende landesabhängige Information:

Erstes Wort:	Datum-/Zeitformat
Drittes Byte:	ASCIZ Währungssymbol
Viertes Byte:	ASCIZ Tausender-Trennsymbol
Fünftes Byte:	ASCIZ Dezimal-Trennsymbol

Werte im Datum-/Zeitformatwort haben folgende Bedeutung:

0 - USA-Standard	h:m:s m/t/j
1 - Europäischer Standard	h:m:s t/m/j
2 - Japanischer Standard	j/m/t h:m:s

39 Verzeichnis erstellen. Diese Routine erstellt einen neuen Eintrag in einem Verzeichnis. Bei Eintritt zeigt DS:DX auf den ASCIZ-Namen eines Verzeichnispfades. Die Routine erstellt einen Verzeichniseintrag für den letzten Namen auf dem Pfad. Bei Rückkehr ist AX bei nicht gefundenem Pfad 3 und bei verweigertem Zugriff 5.

3A Verzeichnis entfernen. Bei Eintritt zeigt DS:DX auf den ASCIZ-Namen eines Verzeichnispfades, dessen letzter Eintrag ein Unterverzeichnis ist. Diese Routine entfernt den letzten Eintrag im Pfad vom Hauptprogramm-Verzeichnis. Das aktuelle Standardverzeichnis oder ein Unterverzeichnis mit Dateien kann nicht entfernt werden. Bei Rückkehr ist AX 3, wenn der Pfad nicht gefunden, 5, wenn Zugriff verweigert wurde, und 10, wenn der letzte Eintrag das aktuelle Standardverzeichnis ist.

3B Aktuelles Verzeichnis ändern. Bei Eintritt zeigt DS:DX auf den ASCIZ-Namen eines Verzeichnispfades. Dieser Pfad wird zum aktuellen Pfad. Fehlt ein Teil des angegebenen Pfadnamens, ändert sich das aktuelle Verzeichnis nicht. Bei Rückkehr ist AX bei nicht gefundenem Pfad 3 und bei verweigertem Zugriff 5.

3C Datei erstellen. Bei Eintritt zeigt DS:DX auf den Namen einer Datei. Besteht die Datei nicht, wird sie von dieser Routine im entsprechenden Verzeichnis erstellt und mit dem Lese-/Schreib-Zugriffsschutzcode versehen; 0 wird in CX eingegeben. Falls die Datei besteht, verkürzt die Routine deren Länge auf Null, um sie zum Schreiben vorzubereiten. Bei normaler Rückkehr enthält AX den Dateibezeichner. Anderenfalls ist AX bei nicht gefundenem Pfad 3, bei zu vielen eröffneten Dateien 4 oder bei verweigertem Zugriff 5.

3D Datei eröffnen. Bei Eintritt zeigt DS:DX auf den ASCIZ-Namen der zu eröffnenden Datei. AL enthält einen Zugriffsschutzcode. Die Codewerte und deren Bedeutungen sind:

0 Datei zum Ablesen eröffnet.

1 Datei zum Schreiben eröffnet.

2 Datei zum Lesen und Schreiben eröffnet.

Diese Routine setzt den Lese-/Schreibanzeiger zum ersten Byte der Datei und die Satzgröße der Datei auf 1 Byte. Bei normaler Rückkehr enthält AX einen Dateibezeichner. Bei weiterer E/A an die Datei muß der Dateibezeichner benutzt werden. Bei Initialisierung hat DOS die maximale Anzahl der in CONFIG.SYS definierten Dateien. AX ist bei nicht gefundener Datei 2, bei zu vielen eröffneten Dateien 4, bei verweigertem Zugriff 5 und bei ungültigem Zugriff C.

3E Datei schließen. Bei Eintritt enthält einen Dateibezeichner. Diese Routine schließt die zugeordnete Datei und leert die internen Puffer. Bei Rückkehr ist AX 6, wenn der Bezeichner ungültig war.

3F Datei oder Gerät lesen. Bei Eintritt enthält DX die Adresse eines Puffers, CX die Anzahl der Bytes für E/A und BX einen Dateibezeichner. Diese Routine überträgt die angegebene Byteanzahl von einer Datei oder einem Gerät in den Puffer. Nicht alle Bytes werden unbedingt gelesen. Nur eine Textzeile kann z. B. jeweils von der Tastatur abgelesen werden. AX sendet die Anzahl der gelesenen Bytes. Ist AX Null, hat das Programm versucht, ab Dateiende zu lesen. Ist AX 5, wurde der Zugriff verweigert. Ist AX 6, war der Dateibezeichner unzulässig. Der Computer benutzt für E/A normalisierte Anzeiger; Segment-"Wraparound" findet nicht statt.

40 Datei schreiben. Bei Eintritt enthält DS:DX die Adresse eines Puffers; CX enthält die Byteanzahl für E/A und BX einen Dateibezeichner. Diese Routine überträgt die angegebene Byteanzahl vom Puffer auf eine Datei oder ein Gerät. AX sendet die Anzahl geschriebener Bytes. Ein Fehler ist aufgetreten, wenn AX von CX abweicht. Ist AX 5, wurde der Zugriff verweigert; ist AX 6, war der Dateibezeichner unzulässig. Der Computer benutzt für E/A normalisierte Anzeiger; Segment-"Wraparound" findet nicht statt.

41 Verzeichniseintragung löschen. Bei Eintritt enthält DS:DX einen Anzeiger auf einen Dateinamen. Diese Routine löscht die Datei aus ihrem Verzeichnis. Ist die Datei gegenwärtig unter einem anderen Bezeichner eröffnet, findet kein Löschvorgang statt. Bei Rückkehr ist AX bei nicht gefundener Datei 2, bei verweigertem Zugriff 5.

42 Datei Lese-/Schreibanzeiger verschieben. Bei Eintritt enthält CX die werthöchsten und DX die wertniedrigsten 16 Bits eines 32-Bit Integer Offsets. BX enthält einen Dateibezeichner. AL enthält einen Subfunktionscode, der angibt, wie der Lese-/Schreibanzeiger verschoben wird. Bedeutung der Subfunktionscodes:

0 Anzeiger verschiebt die Offset-Bytes vom Dateianfang.

1 Anzeiger verschiebt sich zur aktuellen Position plus Offset.

2 Anzeiger verschiebt sich zum Dateiende plus Offset.

Bei normaler Rückkehr enthält DX:AX die neue Anzeigerposition. Anderenfalls enthält AX bei ungültiger Subfunktion 1 oder bei ungültigem Dateibezeichner 6.

43 Schreibschutz ändern. Bei Eintritt zeigt DX auf den ASCIZ-Namen einer Datei. AL enthält einen Subfunktionscode: 0 = aktuelles Schreibschutzattribut senden; 1 = Schreibschutzattribut setzen. CX enthält einen Code, der das neue Attribut angibt. Bedeutung der CX-Codes:

Bit	Bedeutung
0	0 = Ablesen und Schreiben; 1 = nur Ablesen
1	1 = Verborgene Datei
2	1 = Systemdatei
3	1 = Volume-ID
4	1 = Verzeichnis
5	1 = Datei, die erstellt oder in die geschrieben wird (Dieses Bit ist immer gesetzt, außer wenn von dieser Routine geändert.)

Bei Rückkehr ist AX bei ungültiger Subfunktion 1, bei nicht gefundenem Pfad 3 oder bei verweigertem Zugriff 5.

44 E/A Gerätesteuerung. Bei Eintritt enthält AL einen Subfunktionscode, der den durchzuführenden Vorgang angibt. Bedeutung der Subfunktionscodes:

0 Geräteinformation aufgreifen (in DX gesendet).
1 Geräteinformation setzen (wie in DX enthalten).
2 Anzahl der in CX angegebenen Bytes vom Gerätesteuerungskanal in DS:DX lesen.
3 Anzahl der in CS angegebenen Bytes von DS:DX zum Gerätesteuerungskanal schreiben.
4 Anzahl der in CX angegebenen Bytes in DS:DX von der Laufwerknummer in BL lesen: 0 = Standardwert, 1 = A, 2 = B, 3 = C.
5 Anzahl der in CX angegebenen Bytes von DS:DX zur Laufwerknummer in BL schreiben.
6 Eingabestatus aufgreifen.
7 Ausgabestatus aufgreifen.

Nur Subfunktionen 0, 6 und 7 sind für gewöhnliche Dateien zulässig. Alle Subfunktionen sind für Geräte zulässig.

Bei Subfunktionen 0, 1, 2, 3, 6 und 7 enthält BX bei Eintritt einen Dateibezeichner. Bei Subfunktionen 2, 3, 4 und 5 enthält DS:DX eine Pufferadresse, und CX enthält das Zählergebnis bei Eintritt. Bei Subfunktionen 4 und 5 enthält BL bei Eintritt eine Laufwerknummer: 0 = Standardlaufwerk, 1 = A, 2 = B, 3 = C usw. Bei Subfunktion 1 enthält DX Daten bei Eintritt. Bei Subfunktion 0 sendet DX Daten bei Austritt.

Subfunktionen 0 und 1 ermöglichen es, Geräteinformation entweder zu erhalten oder zu setzen. DX-Bits definieren diese Subfunktionen folgendermaßen (oberes Byte muß bei Subfunktion 1 Null sein):

BIT	15	14	13 - 8	7	6	5	4	3	2	1	0
	R	C		I	E	R	R	I	I	I	I
	E	T		S	O	A	e	S	S	S	S
	S	R	Reserviert	D	F	W	s	C	N	C	C
		L		E				L	U	D	I
				V				K	L	T	N
								Laufwerknummer			

ISDEV = 1, wenn dieser Kanal ein Zeichengerät ist, z. B. Drucker.
= 0, wenn dieser Kanal eine Diskdatei ist (Bits 8 bis 15 müssen 0 sein).

Wenn ISDEV = 1

EOF = 0, wenn Dateiende bei Eingabe erreicht ist.
RAW = 1, wenn KTRL + C Auffang für dieses Gerät ausgeschaltet ist.
= 0, wenn KTRL + C Auffang für dieses Gerät eingeschaltet ist.
ISCLK = 1, wenn dieses Gerät das Zeitgerät ist.
ISNUL = 1, wenn dieses Gerät das Nullgerät ist.
ISCOT = 1, wenn dieses Gerät das Konsolen-Ausgabegerät ist.
ISCIN = 1, wenn dieses Gerät die Konsolen-Eingabe ist.

KTRL = 0, wenn dieses Gerät keine Steuerfolgen von Subfunktionen 2 oder 3 ausführen kann.
= 1, wenn dieses Gerät Steuerfolgen von Subfunktionen 2 or 3 verarbeiten kann.

ANMERKUNG:
Sie können das KTRL-Bit nicht setzen.

Wenn ISDEV = 0
EOF = 0, wenn Kanal geschrieben ist.
Bits 0-5 ist die Blockgerätenummer für den Kanal (0 = A, 1 = B usw.).

Mit Subfunktionen 2 bis 5 können beliebige Steuerfolgen an ein Gerät geschickt oder von einem Gerät empfangen werden. Die Syntax ist dieselbe wie bei Lese- und Schreibaufrufen (3F bzw. 40), außer bei 4 und 5, die anstelle eines Dateibezeichners in BX eine Laufwerknummer in BL eingeben. Das KTRL-Bit muß 1 sein.

Subfunktionen 6 und 7 zeigen den Status (bereit oder nicht bereit) einer Datei an, die für Ein- oder Ausgabe eröffnet ist. Die Datei kann für ein Gerät oder eine Diskdatei eröffnet sein. Der Status wird folgendermaßen definiert:

Eingabe

AL = FF	Bereit. AL ist immer FF, bis das Dateiende erreicht ist.
= 0	Nicht bereit. Nachdem das Dateiende erreicht ist, ist AL immer 0, es sei denn, Funktionsaufruf 42 hat den aktuellen Anzeiger verschoben.

Ausgabe

AL = FF	Bereit. AL ist immer FF, auch wenn die Disk voll ist.

ANMERKUNG:
Diese Routine zeigt den E/A-Status zur Zeit des Systemaufrufs an. In zukünftigen Versionen kann es bei Rückkehr der Kontrolle vom System sein, daß der angezeigte Status nicht mit dem wahren Status des Gerätes oder der Datei übereinstimmt.

Bei Austritt sendet AX bei ungültiger Subfunktion 1, bei ungültigem Dateibezeichner 6, bei ungültigen Daten 0D und bei verweigertem Zugriff 5.

45 Dateibezeichner duplizieren. Bei Eintritt enthält BX einen Dateibezeichner. Bei normaler Rückkehr hat AX einen neuen Bezeichner, der sich auf die gleiche Datei und die gleiche Position bezieht. Anderenfalls enthält AX bei zu vielen eröffneten Dateien 4, oder bei ungültigem Dateibezeichner 6.

46 Doppel eines Dateibezeichners erzwingen. Bei Eintritt hat BX einen Dateibezeichner und CX einen neuen Bezeichner. Der neue Bezeichner bezieht sich auf die gleiche Datei wie der erste. Bestand schon eine Datei mit diesem neuen Bezeichner, wird die erste Datei geschlossen. Sendet AX 6, war der Dateibezeichner ungültig.

47 Aktuelles Verzeichnis. Diese Routine gibt das aktuelle Verzeichnis für ein Laufwerk an. Bei Eintritt enthält DS:SI einen Anzeiger auf einen 64-Byte-Bereich, der das aktuelle Verzeichnis enthält. DL enthält einen Laufwerkcode: 0 = Standardwert; 1 = A, 2 = B, 3 = C usw. Bei Rückkehr enthält AX 0F, wenn das Laufwerk ungültig war.

48 Speicherraum zuordnen. Bei Eintritt enthält BX die in Absätzen angegebene Größe des angeforderten Blocks. Bei normaler Rückkehr enthält AX:0 die Adresse eines freien Speicherblocks der angeforderten Größe. Anderenfalls ist AX 7, wenn der zugewiesene Bereich beschädigt ist, oder 8, wenn nicht genug Speicherraum verfügbar ist. Ist ein freier Block der angegebenen Größe nicht verfügbar, sendet BX den Wert des größten verfügbaren Blocks.

49 Zugewiesenen Speicher freigeben. Bei Eintritt hat ES die Adresse eines durch Funktionsaufruf 48 zugeordneten Speicherblocks. Dieser Block wird zum Systempool zurückgegeben. Bei Rückkehr enthält AX 7, wenn der freigestellte Bereich beschädigt, und 9, wenn der Block ungültig ist.

4A Zugewiesenen Block modifizieren. Bei Eintritt enthält ES die Absatzadresse eines durch Funktionsaufruf 48 zugeordneten Speicherblocks und BX eine neue, in Absätzen angegebene Größenspezifikation für den Block. Der Block wird entsprechend vergrößert oder verkleinert. Ist der angegebene Wert zu groß, enthält BX den größtmöglichen Wert. Bei Rückkehr ist AX 7, wenn der zugewiesene Bereich beschädigt, 8, wenn nicht genug Speicherraum vorhanden, oder 9, wenn die Adresse in ES ungültig ist.

4B Programm ausführen. Diese Routine ermöglicht einem Programm, ein anderes Programm in den Speicher zu laden und mit seiner Ausführung zu beginnen. Bei Eintritt zeigt DS:DX auf den ASCIZ-Namen der zu ladenden Datei. ES:BX zeigt auf einen Parameterblock für das Laden. AL enthält einen Subfunktionscode. Code-Bedeutung:

0 Programm laden und ausführen. Die Routine erstellt einen Programmvorsatz und setzt die Abschluß- und KTRL + C Adresse gemäß der diesem Aufruf folgenden Instruktion.

ANMERKUNG:
Wenn die Steuerung durch KTRL + C oder Programmabschluß zum Aufrufprogramm zurückkehrt, werden alle Register einschließlich des Stapelanzeigers geändert. Das hängt damit zusammen, daß die Steuerung vom ausgeführten Programm zurückgegeben wird und nicht vom System. Um den Stapel wieder zu erreichen, den SS:SP Wert an einer von CS adressierbaren Stelle speichern.

3 Programm laden; keinen Programmvorsatz erstellen und Ausführung nicht beginnen. Der angegebene Bereich erhält CS:IP und SS:SP des Programms.

Der Parameterblock hat für die verschiedenen Subfunktionscodes folgende Formate:

AL = 0 (Laden und Ausführen)

WORT Segmentadresse der Umgebung
DWORT Anzeiger auf die Befehlszeile bei 80H
DWORT Anzeiger auf Standard-FCB, der bei 5CH eingegeben sein soll
DWORT Anzeiger auf Standard-FCB, der bei 6CH eingegeben sein soll

AL = 3 (Lade-Überlagerung)

WORT Segmentadresse, an der die Datei geladen wird
WORT Verschiebungsfaktor für das Bild

Diese Routine dupliziert alle eröffneten Dateien eines Programms im Nebenprogramm. Ein Hauptprogramm könnte z. B. eine Reihe von Sätzen in eine Datei schreiben, die Datei als Standardeingabe eröffnen, eine Listdatei als Standardausgabe eröffnen und ein Sortierprogramm ausführen, das seine Eingabe von der Datei übernimmt, die als Standardeingabe eröffnet wurde, und sie in die Listdatei schreibt. Zur Steuerung der E/A-Geräte Funktionsaufrufe 0 bis C oder 44 benutzen.

Das Nebenprogramm empfängt auch eine "Umgebung" vom Hauptprogramm. Eine "Umgebung" ist ein Block von Textfolgen, die verschiedene Konfigurationsparameter übertragen. Der Block muß an einer Absatzgrenze beginnen, kürzer als 32K Bytes sein und mit 00H enden.

Umgebungs-Textfolgen haben gewöhnlich folgende Form:

```
Parameter=Wert
```

Enthält das Umgebungs-Adreßfeld Null, erhält das Nebenprogramm die unveränderte Umgebung des Hauptprogramms.

Bei Rückkehr ist AX bei ungültiger Subfunktion 1, bei nicht gefundener Datei 2, bei verweigertem Zugriff 5, bei nicht ausreichendem Speicherraum 8, bei ungültigem Parameterblock 9, bei ungültiger Umgebung 0A oder bei ungültigem Parameterblockformat 0B.

4C Prozedur verlassen. Diese Routine beendet die aktuelle Prozedur und übergibt der Aufrufprozedur die Steuerung. Außerdem kann in AL ein Rückkehrcode angegeben werden. Die Routine schließt alle zur Zeit offenen Dateien.

4D Rückkehrcode einer Nebenprozedur zurückholen. AX erhält den Rückkehrcode eines Nebenprogramms. Das wertniedrigste Byte dieses Codes wird von Funktionsaufruf 31 oder 4C gesendet. Das werthöchste Byte ist 0 bei normalem Abschluß (durch Funktionsaufruf 4C), 1 bei KTRL + C Abschluß, 2 bei Abbruchfehler und 3 bei Abschluß durch Funktionsaufruf 31 (beenden, jedoch resident bleiben). Der Ausgangscode wird nur einmal gesendet.

4E Erste finden. Diese Routine sucht nach der ersten Datei, die einem Dateinamen mit einem oder mehreren Allzweck-Zeichen entspricht und einen Nebensatz der Attribute enthält, die für die Suchdatei angegeben sind. Bei Eintritt zeigt DX auf einen Pfadnamen, dessen letzte Komponente Allzweck-Zeichen enthält. CX enthält die Attribut-Spezifikation (siehe Funktionsaufruf 43). Findet die Routine einen Gleichwert, schreibt sie einen Datenblock an der aktuellen Disk-Übertragungsadresse (siehe Abschnitt A.4). Die Daten werden folgendermaßen formatiert:

Byte	Bedeutung
0-20	Reserviert. Diese Bytes enthalten Suchattribut, Laufwerknummer (0 = Standardwert, 1 = A, 2 = B usw.), Dateinamen, Erweiterung und zusätzliche Informationen.
21	Attribut gefunden.
22-23	Zeit der Erstellung oder letzten Revision.
24-25	Datum der Erstellung oder letzten Revision.
26-27	Dateigröße (wertniedrigste Bytes).
28-29	Dateigröße (werthöchste Bytes).
30-42	Dateiname und Erweiterung ohne Interpunktion, links ausgerichtet und Leerräume, falls nötig. Das Feld endet mit 00H.

Bei Rückkehr ist AX bei nicht gefundener Datei 2, bei keinen weiteren Dateien im Verzeichnis 12.

Um weitere Dateien zu finden, Funktionsaufruf 4F benutzen.

4F Nächste finden. Diese Routine benutzt den von Funktionsaufruf 4E (Erste Finden) geschriebenen Datenblock, um die nächste Verzeichniseintragung mit entsprechendem Dateinamen und einem Nebensatz der angegebenen Attribute zu finden. Bei Eintritt muß die aktuelle Disk-Übertragungsadresse (siehe Abschnitt A.4) auf einen von Funktionsaufruf 4E gesendeten Datenblock zeigen. Bei Rückkehr ist AX 12, wenn keine weiteren Dateien im Verzeichnis sind.

54 Prüfkennzeichen aufgreifen. Diese Routine sendet den aktuellen Schreibprüfstatus in AL: 0 = Nicht Prüfen (Standardwert); 1 = Prüfen.

56 Umbenennen. Diese Routine benennt eine Datei um, und zwar entweder auf demselben oder einem anderen Pfad. Die Pfade müssen auf demselben Gerät sein. Bei Eintritt zeigt DX auf den Ursprungspfadnamen und DI auf den neuen Pfadnamen. Bei Austritt ist AX 3, wenn einer der Pfade nicht gefunden wurde, 5, wenn Zugriff verweigert wurde, und 11, wenn die Pfade nicht auf demselben Gerät waren.

57 Datei-Zeitangabe. Diese Routine sendet oder setzt Zeit und Datum des letzten Dateischreibvorgangs. Bei Eintritt enthält BX einen Dateibezeichner. AL enthält einen Subfunktionscode: 0 = Zeit und Datum senden; 1 = Zeit und Datum setzen. Ist der Subfunktionscode bei Eintritt 1, muß CX die Zeit und DX das Datum erhalten. Ist der Subfunktionscode 0, sendet CX die Zeit und DX das Datum. Bei Rückkehr ist AX bei ungültiger Subfunktion 1 und bei ungültigem Dateibezeichner 6.

C.3 CP/M-AUFRUFRICHTLINIEN

Ein zusätzlicher Mechanismus besteht für Programme, die CP/M-Aufrufrichtlinien entsprechen. Das CL Register erhält die Funktionsnummer. Andere Register erhalten die für die Funktionsspezifikation normalen Werte, und ein Intrasegmentaufruf zu Stelle 5 im aktuellen Codesegment findet statt. Diese Methode ist nur bei Funktionen verfügbar, die keinen Parameter in AL eingeben und deren Nummer 36 nicht unterschreitet. Der Mechanismus zerstört stets das AX Register; anderenfalls stimmt dieser Mechanismus mit den gewöhnlichen Funktionsanforderungen überein.

ANHANG D
BIOS INFORMATION

D.1 EINFÜHRUNG

Dieser Anhang enthält WANG PC BIOS Information für Programmierer. Einige der in diesem Anhang beschriebenen Funktionen sind u. U. nicht in der aktuellen BIOS-Version ausführbar. Rückkehrcode 02H bedeutet, daß eine angeforderte Funktion noch nicht ausgeführt ist.

Rückkehrcode 0 bedeutet immer erfolgreiche Beendung. Da zukünftige BIOS-Versionen u. U. über mehr Fehlercodes verfügen, muß das Programm so geschrieben werden, daß es auf ein Nicht-Null Resultat geprüft werden kann.

D.2 SOFTWARE/BIOS SCHNITTSTELLE

Der Eingangspunkt für die Software BIOS Schnittstelle ist Unterbrechung 88H. Die BIOS Schnittstelle empfängt eine Funktionsnummer im AL-Register und, falls notwendig, andere Parameter in anderen Registern. Folgende Funktionen sind von BIOS verfügbar:

Wert	Funktion
000H	ES:BX setzen, so daß es auf den Systemnamen zeigt. Beim WANG PC wird die Folge "Wang Professional Computer" durch ein Nullzeichen (0) beendet.
001H	ES:BX setzen, so daß es auf die System-Konfigurationstabelle zeigt.
002H	Ereignisauffangstelle setzen. BX enthält das Ereignis-verzeichnis, CX die Zählung der Ereignisse, die vor Ausgabe der Auffangstelle auftreten sollen, und DS:DX enthält die Adresse der Routine, die bei Auftritt der Auffangstelle aufgerufen werden soll. Bei Rückkehr enthält BX die Nummer der Auffangstellen-Warteschlange (QID).
003H	Ereignisauffangstelle löschen. BX enthält die QID-Nummer der zu löschenden Ereignisauffangstelle.
004H	Reserviert.
005H	Reserviert.

Wert	Funktion
006H	Zeichen zum Bildschirm senden. BL enthält das zum Bildschirm zu sendende Zeichen. Diese Funktion läßt Programme schneller am Bildschirm erscheinen als DOS. Zukünftig wird BH ein Verzeichnis enthalten, das anzeigt, an welchen Bildschirm eine Ausgabe gesendet werden soll, wenn Null der Standardwert ist. BH sollte Null sein, um zukünftige Kompatibilität zu gewährleisten.
007H	Ein oder mehrere Bytes zur Tastatur senden. Wenn CX 1 ist, enthält BL das für die Tastatur bestimmte Byte, anderenfalls zeigt DS:DX auf einen Zeichenpuffer, und CX enthält die Zählung. Diese Funktion ist für Programme mit akustischen Tastaturfähigkeiten vorteilhaft. Programmierer, die Akustik-Chip-Folgen zur Tastatur senden wollen, sollten die ganze 2- oder 3-Byte-Tastaturfolge in einem Aufruf senden. Dies verhindert, daß ein Tastenanschlag eine Unterbrechung verursacht, die gewöhnlich ein Klicken, Summzeichen oder FEST UMSCH LED zur Tastatur schicken würde. Ferner hat der Tastatur-Ausgabepuffer eine vorbestimmte Größe. Ist er voll, werden darauffolgende Zeichen ignoriert. Werden mehrere Bytes in einem Aufruf gesendet, nimmt der Puffer sie nur an, wenn alle erfaßt werden können. AL sendet 05H, wenn der Puffer nicht den ganzen Byte-Block erfassen kann.
008H	Reserviert.
009H	DMA Kanal setzen. Der in BL angegebene DMA Kanal wird für den in BH angegebenen Übertragungsmodus gesetzt. 000H bedeutet Übertragung prüfen, 001H Übertragung schreiben und 002H Übertragung lesen. Ist BH 0FFH, ist der Kanal gesperrt. Sonst ist CX die Anzahl der übertagenen Bytes (CX = 0 bedeutet 64KB) und DS:DX die Pufferadresse. Rückkehrcode 004H bedeutet, daß die Übertragung eine physische 64KB-Grenze überschreitet.
00AH	DMA Kanal zuordnen. Bei Austritt enthält BL die Nummer des zugeordneten DMA Kanals, falls einer verfügbar ist. Wenn keiner verfügbar ist, sendet AL 08H als Fehlercode. Bei Eintritt zeigt DS:DX auf die aufzurufende Routine (über einen langen Aufruf), wenn das Prozedurende für den Kanal angezeigt wird.
00BH	DMA Kanal freigeben. Bei Eintritt enthält BL den vorher zugeordneten DMA-Kanal.
00CH	KTRL + S/KTRL + Q Freigabe-Byte setzen oder löschen. BL enthält den Wert für das Freigabe-Byte. Ist BL nicht Null, verarbeitet BIOS KTRL + S und KTRL + Q sowohl für die Steuerkonsole als auch die serielle Anschlußstelle. Sonst werden diese Tasten als normale Tasten zum DOS gepuffert, von dem sie bearbeitet werden (Standardwert).

ANMERKUNG:
Funktion 0CH beeinflußt serielle E/A und E/A zur Steuerkonsole.

Bei Rückkehr enthält AL den Beendungscode für die Funktion. Folgende Werte sind für diesen Code bestimmt:

Wert	Beschreibung
000H	Funktion normal abgeschlossen
001H	Funktionsverzeichnis (AL) ungültig
002H	Angegebene Funktion nicht ausgeführt
003H	Ungültige(r) Parameter
004H	Vorgang nicht möglich
005H	Ereigniswarteschlange oder Tastaturpuffer voll
006H	QID ungültig
007H	Bildschirmverzeichnis ungültig
008H	Resource-Zuordnung verweigert
009H	Interner Fehler unbestimmbarer Art

D.3 BIOS AUFFANGBARE EREIGNISSE

Bei BIOS-Funktionen 002H und 003H (Auffangstellen setzen oder löschen) wird angegeben, welches Ereignis durch das Verzeichnis in BX aufzufangen ist. Die Auffangroutine bei DS:DX wird (über einen Fernaufruf) aufgerufen, wenn das Ereignis so oft auftreten ist, wie durch die Zahl in CX bestimmt wurde. Ist die Zahl in CX Null, setzt BIOS voraus, daß das Auffang-Bearbeitungsprogramm die Unterbrechung über die Hardware verarbeitet. Anderenfalls verarbeitet BIOS die Unterbrechung in der gewöhnlichen Weise und ruft dann die angegebene Auffangroutine auf.

ANMERKUNG:
Weil der Auffangroutinenaufruf innerhalb einer Unterbrechung auftritt, kann die Auffangroutine DOS nicht zur Ausführung von Funktionen aufrufen. Auch sollte die Auffangroutine nicht für längere Zeit laufen. Die Auffangroutine kann keine Unterbrechungen freigeben.

Wenn der Auffangroutinenaufruf stattfindet, enthält AX das Ereignisverzeichnis. Deshalb kann eine einzelne Auffangroutine für mehrere Bedingungen bestehen. Die Routine nimmt u. U. an, daß Stapelraum bis zu 64 Bytes verfügbar ist und daß alle Register erhalten bleiben. Bevor der Auffangroutinenaufruf stattfindet, wird die Zählprozedur zurückgestellt, damit der Auffang erneut beginnen kann. Die Auffangstelle kann nur mit der Funktion Auffang löschen (002H) gelöscht werden. Diese Auffangstellen sind demnach nicht einmalig, sondern treten wiederholt auf. Zum Löschen muß die QID-Nummer benutzt werden, die bei Setzen der Auffangstelle angegeben wird.

Folgende Ereignisse (von denen einige noch nicht ausführbar sind) können aufgefangen werden:

Index	Ereignis
0000H	10-ms-Zeitgeber (siehe Anmerkung 1)
0001H	Eingabezeichen für serielle Anschlußstelle bereit
0002H	Ausgabezeichen für serielle Anschlußstelle bereit
0003H	Datensatz-Änderungsstatus für serielle Anschlußstelle (siehe Anmerkung 2)
0004H	Parallel-Anschlußstellen-Eingabezeichen bereit
0005H	Parallel-Anschlußstellen-Ausgabezeichen bereit
0006H	Tastatur-Eingabezeichen bereit
0007H	Tastatur-Ausgabezeichen bereit
0008H	Unterbrechung 8087 (siehe Anmerkung 2)
0009H	KTRL-C- (oder UMSCH-ANNULLIER-) Auffangstelle (siehe Anmerkung 2)
000AH	Gedrückte Taste annullieren (siehe Anmerkung 2)
000BH	Floppy-Diskanforderung vollständig
000CH	Winchester-Diskanforderung vollständig
000DH	Paritätsfehler
000EH	E/A-Fehler (vom Optionsfach)
000FH	Unterbrechungsversuch unzulässig
0010H	Kanal 0 Hardware-Unterbrechung (siehe Anmerkung 2)
0011H	Kanal 1 Hardware-Unterbrechung (siehe Anmerkung 2)
0012H	Kanal 2 Hardware-Unterbrechung (siehe Anmerkung 2)
0013H	Kanal 3 Hardware-Unterbrechung (siehe Anmerkung 2)
0014H	Kanal 4 Hardware-Unterbrechung (siehe Anmerkung 2)
0015H	Kanal 5 Hardware-Unterbrechung (siehe Anmerkung 2)
0016H	Kanal 6 Hardware-Unterbrechung (siehe Anmerkung 2)
0017H	Kanal 7 Hardware-Unterbrechung (siehe Anmerkung 2)
0018H	DMA EOP auf DMA Kanal 1 (siehe Anmerkung 2)
0019H	DMA EOP auf DMA Kanal 2 (siehe Anmerkung 2)
001AH	DMA EOP auf DMA Kanal 3 (siehe Anmerkung 2)

Anmerkung:

1) Der 10-ms-Zeitgeber-Ereignisauffang kann nicht bei Nullzählung angefordert werden. In diesem Fall zeigt eine Fehlermeldung an, daß der Parameter ungültig ist.

2) Ereignisse werden jedesmal aufgefangen, ungeachtet der in CX angegebenen Zahl. Die Zahl in CX wird ignoriert.

D.4 SYSTEM-KONFIGURATIONSTABELLE

Die System-Konfigurationstabelle wird von BIOS fortgeschrieben und enthält Informationen über die aktuelle Systemkonfiguration. Ihre Position kann durch die Software BIOS Schnittstelle (Funktion 01H) ermittelt werden.

Das Format der System-Konfigurationstabelle ist wie folgt:

Offset	Name	Größe	Beschreibung
0000H	VERSION	Wort	BIOS-Version (bei werthöchstem Byte ist die Ziffer links vom Dezimalkomma, bei wertniedrigstem Byte rechts)
0002H	MEMSIZE	Wort	Speichergröße in Absätzen
0004H	Reserviert	Dwort	Für zukünftigen Gebrauch reserviert
0008H	SCRNCNT	Wort	Bildschirmzählung im System
000AH	SCRNPTR	Wort	Bildschirminformation-Blockanzeiger (0 oder mehrmals wiederholt)
mmmmH	DISKCNT	Wort	Plattenlaufwerkzählung im System (einschließlich zwei Floppies)
mmmnH	FLOPPY0	Wort	Floppy-Laufwerk 0 Steuerblockanzeiger
mmmoH	FLOPPY1	Wort	Floppy-Laufwerk 1 Steuerblockanzeiger
mmmpH	WINSTAT	Wort	Winchester-Steuerblockanzeiger (0 oder mehrmals wiederholt)

Für jede Videoschaltkarte besteht ein Bildschirminformations-Blockanzeiger (Offset 000AH).

D.4.1 Bildschirminformationsblock

Der Bildschirminformationsblock hat folgendes Format für jede Videoschaltkarte:

Offset	Name	Größe	Beschreibung
0000H	STATE	Byte	Aktiv/Art/Fach-Information
0001H	SCNOFF	Byte	Abtastregister-Offset
0002H	BUFSEG	Wort	Segment mit Rahmen-Pufferbild
0004H	COLORS	Byte	Aktuelle Vorder- und Hintergrundfarben (hohes Nibble: Vordergrund; niedriges Nibble: Hintergrund)
0005H	ROW	Byte	Aktuelle Cursorreihe (0 bis 24)
0006H	COL	Byte	Aktuelle Cursorspalte (0 bis 79)
0007H	ATTR	Byte	Aktuelle Attribute
0008H	AUXMOD	Byte	Ergänzungsmodus-Byte (bestimmt Schaltkarten-Unterbrechungsstatus)
0009H	AUXMD2	Byte	Zweites Ergänzungsmodus-Byte

Format des STATE-Bytes:

Bit(s)	Name	Beschreibung
3-0	SLOTID	Enthält die Fachnummer der gewählten Schaltkarte
6-4	TYPE	Enthält die Art der Videoschaltkarte: 0 - 320x225 Auflösung (niedrig) 1 - 640x225 Auflösung (niedrig) 2 - 800x300 Auflösung (mittlere Auflösung bei Grafiken) 3 - nur Zeichenschaltkarte mittlerer Auflösung 4 - 800x600 Auflösung (hoch)
7	AKTIVE	Falls gesetzt, zeigt den Standardbildschirm an.

Format des ATTR-Bytes:

Bit	Beschreibung
0	Blinken
1	Negativ-Darstellung
2	Leer
3	Fettdruck
4	Überstreichen
5	Unterstreichen
6	Tiefstellung
7	Hochstellung

D.4.2 Plattenlaufwerk-Steuerblöcke

Der Plattenlaufwerk-Steuerblock hat folgendes Format in Assemblerstruktur:

```
FCB       Struc
SxTYPE    Db   ?          ;Disktyp (siehe "Disktyp")
SxSTATE   Db   ?          ;Laufwerkstatus (siehe "Status-Byte"
SxBPS     Db   ?          ;Bytes pro Sektor
SxSPT     Db   ?          ;Sektoren pro Spur
SxHPC     Db   ?          ;Köpfe pro Zylinder
SxCPD     Dw   ?          ;Zylinder pro Disk
SxSPD     Dw   ?          ;Sektoren pro Disk
FCB       Ends
```

Disktyp

Das Disktyp-Byte (SxTYPE) zeigt an, über welche Laufwerkart der Steuerblock Informationen enthält. Die gegenwärtig definierten Arten sind 00H für Floppies, 01H für Winchester, 02H für RAM-Disks und FFH für andere.

Status-Byte

Das Status-Byte (SxSTATE) wird folgendermaßen codiert:

Bit(s)	Name	Bedeutung bei gesetztem Bit
7	EXIST	Laufwerk besteht
6	WP	Disk schreibgeschützt
5	ACTIVE	Laufwerk aktiv
4	SELECT	Laufwerk angewählt
3	MEDIA	Disk u. U. geändert
2	DISK	Disk im Laufwerk
1	MOTOR	Motor eingeschaltet
0	INIT	Laufwerk initialisiert

ANMERKUNG:
Obwohl sich einige Bits nicht auf alle Disktypen beziehen, werden sie im allgemeinen ihrer Bedeutung nach entsprechend gesetzt.

ANHANG E
GERÄTEANTRIEB

E.1 EINFÜHRUNG

Dieser Anhang beschreibt die Struktur und Funktion von Geräteantrieben und ermöglicht es, individuelle Antriebsroutinen zu schreiben oder auf bestehende zuzugreifen. Dies sollte nur von erfahrenen Programmierern unternommen werden.

E.2 GERÄTESCHNITTSTELLE

Die MS-DOS Geräteschnittstelle für Version 2 ist nicht für DOS Version 2 bestimmt, sondern dient auch als eine mehr allgemeine Schnittstelle für zukünftige DOS-Versionen.

Die neue Schnittstelle benutzt keine Sprungtabelle an fester Stelle wie vorhergehende Versionen, sondern verbindet Geräte in einer Kette. Diese Kette ermöglicht das Schreiben und Installieren neuer Antriebsroutinen für optionale Hardware.

Jede Antriebsroutine in der Kette bestimmt zwei Eingangsstellen: eine für eine Warteschlangenroutine und eine für ein Dienstprogramm. Die zwei Eingangstellen sind bei Version 2 nicht voll funktionsfähig; bei Version 2 wird einfach die Warteschlangenroutine und dann sofort das Dienstprogramm aufgerufen.

Doppelte Eingangsstellen sollen Multi-Tasking-Fähigkeiten zukünftiger MS-DOS Versionen ermöglichen. In Multi-Tasking-Umgebungen muß E/A mit CPU-Funktionen parallel ablaufen. Um dies zu erreichen, wird die Warteschlangen-routine zur Einreihung von Anforderungen und schneller Rückkehr abgerufen. Bei Warteschlangen-Anforderungen genügt es nicht mehr, E/A- Information durch die Register zu senden, da möglicherweise mehrere Anforderungen gleichzeitig warten. Deshalb benutzt die Warteschlangenroutine einen Datenblock zum Senden von Anforderungs-Information. Die Geräteantriebsroutine enthält einen Anzeiger auf den Datenblock, und der Aufruf an die Warteschlangenroutine reiht den Datenblockanzeiger in die Warteschlange ein.

Das Dienstprogramm ist dafür verantwortlich, die tatsächliche E/A durchzuführen, indem es Anforderungen von der Warteschlange verarbeitet, die durch die Einreihroutine aufgestellt wurde. Der Aufruf zum Dienstprogramm sendet keine Parameter. Das Dienstprogramm führt den Vorgang aufgrund des eingereihten Datenblocks aus. Ist eine Anforderung verarbeitet, stellt das Dienstprogramm Rückkehranweisungen auf und kennzeichnet die Anforderung als ausgeführt.

Zusätzlich zu der Anforderungskette für eine bestimmte Antriebsroutine, verbindet DOS den Datenblock in einer Globalkette aller ausstehenden E/A-Anforderungen. Die Antriebsroutine verbindet dann den Block mit seiner lokalen Anforderungskette für dieses Gerät. DOS sucht von Zeit zu Zeit die Globalkette nach Anforderungen ab, die als ausgeführt gekennzeichnet sind, und aktiviert ausstehende Verarbeitungsanforderungen. Das Geräte-Dienstprogramm nimmt Verarbeitungsanforderungen aus der lokalen Kette.

Im Gegensatz zu zukünftigen Versionen führt MS-DOS Version 2 die meisten dieser Eigenschaften nicht durch. Globale oder lokale Warteschlangen bestehen nicht. Nur eine Anforderung steht jeweils aus, und DOS wartet bis diese abgeschlossen ist. Für Version 2 genügt es, daß die Warteschlangenroutine die Adresse des Datenblocks an einer festen Stelle speichert und das Dienstprogramm diesen Block durch Ausführen der Anforderung verarbeitet und zurückkehrt. Zur Kompatibilität mit zukünftigen MS-DOS Versionen sollte jedoch der Antrieb den Block mit der lokalen Kette verbinden oder davon lösen. Kehrt die Steuerung vom Dienstprogramm an DOS zurück, nimmt DOS an, daß die Anforderung ausgeführt ist.

Antriebsroutinen können DOS Funktionen nur im INIT-Code abrufen (siehe Abschnitt E.4). Um Hardware-Unterbrechungen zu setzen, müssen bei Antriebsroutinen BIOS Software Schnittstellenroutinen benutzt werden (siehe Anhang D). Wenn Hardware-Unterbrechungen von BIOS abgerufen werden, brauchen Register nicht gespeichert zu werden, da dies von BIOS-Routinen ausgeführt wird.

E.3 FORMAT EINES GERÄTEANTRIEBS

Ein Geräteantrieb ist eine .COM Datei mit allen Codes, die zum Betreiben des Gerätes notwendig sind. Außerdem hat die Datei einen besonderen Vorsatz, der sie als Antriebsroutine identifiziert, die Warteschlangenroutine- und Dienst-Eingangsstelle bestimmt und verschiedene Attribute angibt.

Es gibt zwei grundsätzliche Gerätearten: Zeichengeräte und Blockgeräte. Zeichengeräte führen Zeichen E/A seriell aus. BIOS installiert vier Zeichengeräte mit den Namen CON, AUX, PRN und CLOCK. Diese Antriebe sind immer anwesend. Durch Eröffnen von FCBs kann E/A ausgeführt werden. Ferner können diesen oder anderen Geräten vom Benutzer bestimmte Namen, für die auch FCBs eröffnet werden können, zugewiesen werden.

Blockgeräte sind die Plattenlaufwerke im System. Durch diese werden Zufalls-E/A-Einheiten ausgeführt, d. h. Blöcke, deren physische Größe gewöhnlich mit der Sektorgröße übereinstimmt. Im Gegensatz zu Zeichengeräten tragen diese Geräte keine Namen. Deshalb können sie nicht direkt eröffnet werden. Stattdessen werden die Blockgeräte durch Laufwerkbuchstaben (A,B,C usw.) eingeteilt.

Zeichengeräte-Antriebsroutinen können keine Multi-Einheiten bestimmen, da Zeichengeräte nur einen Namen haben. Eine einzelne Blockgerät-Antriebsroutine kann jedoch für ein oder mehrere Plattenlaufwerke verantwortlich sein. Der Blockgeräteantrieb ALPHA kann z. B. für Laufwerke A, B, C und D verantwortlich sein, d. h. in ALPHA können vier Einheiten bestimmt sein, die vier Laufwerkbuchstaben einnehmen. Die Position des Antriebes in der Antriebskette bestimmt, welche Einheiten den Laufwerkbuchstaben entsprechen. Wenn Antrieb

ALPHA der erste Blockantrieb in der Gerätekette ist und 4 Einheiten bestimmt, sind diese vier Einheiten A, B, C und D. Ist BETA der zweite Blockantrieb und bestimmt drei Einheiten, sind diese drei BETA-Einheiten E, F und G usw.

Die theoretische Begrenzung für Blockgeräteinheiten ist 63 (2*6-1). Nach 26 sind die Laufwerkdesignatoren jedoch nicht mehr alphabetisch.

Der Geräteantriebsvorsatz beginnt bei CS:00 und hat folgendes Format:

DWORT Anzeiger auf nächsten Antriebsvorsatz (-1 bei letztem Gerät)
WORT Attribute Bit 15 = 1 bei Zeichen-, 0 bei Blockgerät Wenn Bit 15 1 ist, Bit 0 = 1 aktuelles sti Gerät Bit 1 = 1 aktuelle sto Ausgabe Bit 2 = 1 aktuelles NUL Gerät Bit 3 = 1 aktuelles CLOCK Gerät Bit 14 ist das IOCTL Bit (siehe unten) Bit 13 ist das NONIBM FORMAT Bit
WORT Anzeiger auf Eingangsstelle für Gerätewarteschlange
WORT Anzeiger auf Eingangsstelle für Gerätedienst
8-BYTE Zeichengerät-Namensfeld Zeichengeräte setzen einen Gerätenamen. 1-Byte Anzahl von Blockgeräteinheiten (von SYSINT gesetzt)

Der Anzeiger auf das nächste Gerät muß auf -1 gesetzt werden, da dies das letzte Gerät zur Zeit der Installation ist. DOS ändert dieses Feld, wenn auf andere Geräte gezeigt werden soll. Die Geräte-Eingangsstellen sind Wörter. Diese müssen Offsets der gleichen Segmentnummer sein, die benutzt wird, um auf diese Tabelle zu zeigen.

Die wichtigste Funktion des Attributfeldes ist es, dem System anzuzeigen, ob dieses Gerät ein Block- oder Zeichengerät (Bit 15) ist. Bestimmte Zeichengeräte werden von den meisten anderen Bits besonders behandelt. (Diese Bits sind für ein Blockgerät ohne Bedeutung.) Angenommen, ein neues Gerät soll das Standardeingabe- (STI) und Standardausgabegerät (STO) sein. Zusätzlich zur Installation des Antriebs müssen SYSINIT und DOS darüber informiert werden, daß das neue Gerät vor dem aktuellen STI und dem STO (das "CON"-Gerät) Vorrang hat. Zu diesem Zweck müssen Bits 0 und 1 auf Wert 1 gesetzt werden.

ANMERKUNG:
Obwohl ein NUL-Geräte-Attribut besteht, kann das NUL-Gerät nicht neu zugeordnet werden. Dieses Attribut zeigt DOS an, ob das NUL-Gerät in Gebrauch ist.

Das NONIBM-Format-Bit bezieht sich nur auf Blockgeräte und bewirkt Ausführung des Geräte-Aufrufes (BPB) "BIOS Parameter-Block aufgreifen" (siehe Abschnitt E.7.4).

Das IOCTL-Bit hat Bedeutung bei Zeichen- und Blockgeräten. Dieses Bit informiert DOS, ob das Gerät Steuerfolgen vom IOCTL-System-Aufruf empfangen kann (siehe Abschnitt C.2.2).

Durch IOCTL-Funktionen kann das Gerät Daten zu eigenen Zwecken selbst senden und empfangen (z. B. Baud Rate, Stop-Bits oder Formularlänge setzen), anstatt Daten über den Gerätekanal wie bei Standardlese- oder -schreibvorgängen zu senden. Die Interpretation der weitergegebenen Information hängt vom Gerät ab, aber die Daten werden gewöhnlich nicht wie normale E/A behandelt.

Kann der Antrieb keine Steuerfolgen verarbeiten, sollte dieses Bit zuerst auf 0 gesetzt werden. Bei einem Versuch, Steuerfolgen zu senden oder zu empfangen (über IOCTL-System-Aufruf), weist Null DOS an, einen Fehlercode an dieses Gerät zurückzugeben. Bei einem Gerät, das Steuerfolgen verarbeiten kann, sollte dieses Bit bei Initialisierung auf 1 gesetzt werden. Für Geräte dieser Art macht DOS Aufrufe zu IOCTL-Eingabe- und -Ausgabegerätfunktionen, um IOCTL-Ketten zu senden und zu empfangen.

Damit SYSINIT einen Geräteantrieb installieren kann, muß eine .COM Datei mit einem Geräteantriebsvorsatz erstellt werden. Das Link-Feld auf -1 setzen. SYSINIT ändert es u. U., nachdem der Antrieb installiert ist. Das Attributfeld und die Eingangstellen korrekt setzen. Ist das Gerät ein Zeichengerät, muß das Namensfeld ausgefüllt werden. Der Name kann ein beliebiger gültiger 8-Zeichen-Dateiname sein. SYSINIT installiert Geräte jedoch immer am Anfang der Geräteliste. Soll ein neues CON-Gerät installiert werden, braucht dieses daher nur CON genannt zu werden. Das neue Gerät geht dem früheren in der Liste voraus, mit dem Ergebnis, daß das frühere Gerät nicht mehr erscheint, weil die Gerätesuche nach der ersten Übereinstimmung endet. Es ist wichtig, die STI- und STO-Bits für ein neues CON-Gerät zu setzen. Ist das Gerät ein Blockgerät, füllt SYSINIT das Namensfeld mit der korrekten Einheitszählung aus.

ANMERKUNG:
Da SYSINIT den Antrieb an beliebiger Stelle installieren kann, d. h. der Antrieb nicht immer an derselben Stelle liegt, ist bei FAR-Speicher-Bezugnahme größte Vorsicht geboten.

E.4 INSTALLATION VON GERÄTEANTRIEBEN

Im Gegensatz zu früheren Versionen können bei MS-DOS Version 2 und zukünftigen Versionen neue Geräteantriebe zur Zeit des Einschaltens dynamisch installiert werden. Dies wird durch den SYSINIT-Modul ausgeführt, der die CONFIG.SYS Datei liest und verarbeitet. CONFIG.SYS muß die Zeile GERÄT = <Filename> enthalten, wobei <Filename> die Geräteantriebsdatei darstellt.
Geräteantrieb

INIT ist eine für jedes Gerät definierte Routine. Diese Routine wird durch SYSINIT zur Zeit der Installation aufgerufen, und danach nicht wieder. INIT zeigt das Doppelwort-Unterbrechungs-Adreßfeld im INIT Datenblock an (siehe Abschnitt E.7.7). Dies ist ein Anzeiger auf das erste freie Speicherbyte nach dem Geräteantrieb und muß gesetzt werden. SYSINIT benutzt diesen, um das nächste Gerät in der Folge, den Menü-Antrieb oder den Befehlsprozessor, zu installieren, wenn keine anderen Antriebe zu installieren sind. Dieser Anzeiger kann durch Fallenlassen des Initialisierungscodes zum Platzsparen benutzt werden. Die freie Byteadresse sollte das erste Byte der INIT-Routine sein.

SYSINIT installiert Blockgeräte auf dieselbe Weise wie Zeichengeräte. Die INIT-Routine für Blockgeräte setzt den INIT Datenblock (siehe Abschnitt E.7.7) zur Rückkehr des ersten freien Byteanzeigers und zusätzlicher Information. Es sendet die Anzahl der Geräteinheiten und bestimmt den Gerätebuchstaben. Wenn der höchste Gerätebuchstabe zur Zeit des Installieraufrufes F ist und die INIT Routine 4 als Einheitsanzahl angibt, erhalten die neuen Einheiten die Bezeichnungen G, H, I und J. Die Position des Antriebes in der Geräteliste und die Einheitsanzahl im Geräteantrieb (im ersten Byte des Gerätenamensfeldes gespeichert) bestimmen die Zuteilung von Einheiten zu Buchstaben.

Die INIT-Routine für Blockgeräte muß außerdem einen Anzeiger auf eine Gruppe von N Wortanzeigern auf BPBs setzen und senden, wobei N die Anzahl der definierten Einheiten ist. Ein BPB ist der INIT-Tabelle ähnlich, die vor MS-DOS Version 2 benutzt wurde. Der BPB enthält jedoch mehr Information. DOS benutzt diese Blöcke zum Aufbau eines Geräteparameterblocks (DPB) für jede Einheit, die jeweils einen BPB-Anzeiger hat. Sind alle Einheiten gleich, können alle Anzeiger auf das gleiche BPB zeigen und somit Platz sparen.

ANMERKUNG:
Die BPB-Anzeigergruppe muß geschützt sein (unterhalb des ersten freien, von der INIT-Routine gesendeten Bytes), weil das DPB an dem Byte beginnt, das durch das erste freie Byte bezeichnet wird.

BPB-Format:

WORT Sektorgröße in Bytes
BYTE Sektoren/Zuordnungseinheit
WORT Anzahl reservierter Sektoren vor der ersten FAT
BYTE Anzahl der FATS
WORT Maximale Zahl von Disk-Verzeichniseintragungen
WORT Gesamte Sektorenanzahl
BYTE Datenträger-Deskriptor
WORT Von jeder FAT belegte Sektorenanzahl

Die definierte Sektorgröße muß entweder bei oder unter der aktuellen Sektor-Maximalgröße liegen, da sonst die Installation nicht stattfinden kann.

Ein Teil der vom BPB gesetzten DPB-Information ist das Datenträger-Deskriptor-Byte. Dieses Byte hat für DOS keine Bedeutung, wird jedoch an Geräte gesendet, um anzugeben, welche DPB-Form von DOS für ein bestimmtes Laufwerk benutzt wird. Blockgeräte sind verschieden; sie können entweder programmierbar oder nicht programmierbar sein. Der Antrieb für ein nicht programmierbares Gerät würde eine Einheit (und damit ein DPB) für jede mögliche Datenträger-Laufwerkkombination bestimmen. Einheit 0 = Laufwerk 0 einseitig, Einheit 1 = Laufwerk 0 doppelseitig usw. Bei diesem Vorgang würden die Datenträger-Deskriptor-Bytes ohne Bedeutung sein. Der Antrieb für ein programmierbares Gerät würde mehrere Datenträger pro Einheit erlauben. In diesem Fall muß die von INIT gesendete BPB-Tabelle groß genug sein, um den größten unterstützten Datenträger unterzubringen. Antriebe für programmierbare Geräte geben durch das Datenträger-Byte an, welcher Datenträger sich gerade in einer Einheit befindet.

ANMERKUNG:
Soll das DPB die richtige Größe für ein programmierbares Gerät schreiben, muß es DOS ein ungültiges Datenträger-Byte geben.

Wenn die BIOS-Standardwertantriebe installiert sind, wird die vorbestimmte Geräteliste abgesucht. Blockantriebe werden, falls vorhanden, wie oben beschrieben installiert. Die Unterbrechungsadresse ändert sich jedoch nicht, weil die Antriebe in BIOS schon resident sind. Laufwerkbuchstaben werden in der gelisteten Reihenfolge zugewiesen. Daher muß das Laufwerk, daß das logische A sein soll, die erste Einheit des ersten Blockgerätes in der Liste sein. Die Reihenfolge der Zeichengeräte ist ebenfalls von Bedeutung. Beim Einschalten müssen mindestens 4 Zeichengeräte bestimmt, und diese die ersten vier Geräte einer der beiden Arten sein. Das erste bestimmte Zeichengerät wird zum STI, zum STO und zum Standard-Fehlerausgabegerät. Das zweite wird zum Satndard-Ergänzungsein- und Ausgabegerät, das dritte zum Standard-Listausgabegerät und das vierte zum Datum-/Zeitgerät (CLOCK). Deshalb muß die BIOS-Geräteliste folgendermaßen aussehen:

->CON-> AUX-> PRN-> CLOCK-> andere Block oder Zeichengeräte

E.5 GERÄTEANTRIEBSFUNKTIONEN

Ein Geräteantrieb bestimmt folgende Funktionen:

Befehls-Code	Funktion
0	INIT
1	DATENTRÄGER PRÜFEN (Nur Block, NOP oder Fehleranzeige bei Zeichen)
2	BPB AUFBAUEN " " " " "
3	IOCTL EINGABE (Nur aufgerufen, WENN GERÄT IOCTL HAT)
4	EINGABE (lesen)
5	NICHT-DESTRUKTIVE EINGABE OHNE WARTEN (Nur Zeichengeräte)
6	EINGABE STATUS " "
7	EINGABE LEEREN " "
8	AUSGABE (schreiben)
9	AUSGABE (schreiben) und Prüfen
10	AUSGABE STATUS " "
11	AUSGABE LEEREN " "
12	IOCTL AUSGABE (Nur aufgerufen, wenn Gerät IOCTL hat)

DOS-FATREAD-Logik erklärt die Funktionen BPB AUFBAUEN und DATENTRÄGER PRÜFEN. FATREAD Logik:

1. Den Laufwerkbuchstaben zum DPB-Anzeiger machen, indem DPB mit korrekter Antrieb-Einheitsnummer gesucht wird.

2. Datenträger-Kontrollroutine für Antriebseinheit aufrufen. DOS sendet sein aktuelles Datenträger-Deskriptor-Byte (vom DPB). Antworten auf die Aufrufe sind:

Code	Bedeutung
01H	Datenträger nicht geändert
FFH	Datenträger geändert
00H	Nicht sicher
Anderer	Fehler

Datenträger nicht geändert	Aktuelles DPB- und Datenträger-Byte korrekt; Datenträger-Kontrollroutine vollständig.
Datenträger geändert	Jeweiliger DPB und Datenträger falsch; Puffer für diese Einheit ungültig machen (siehe 3).
Nicht sicher	Sind modifizierte Puffer in dieser Einheit, muß angenommen werden, daß DBP und Datenträger korrekt und vollständig sind. Bei nicht modifizierten Puffern muß angenommen werden, daß Datenträger geändert wurde. Puffer für diese Einheit ungültig machen (siehe 3).
Fehler	Tritt ein Fehler auf, muß der Antrieb den Fehlercode im Statuswort des statischen Anforderungsvorsatzes entsprechend setzen.

ANMERKUNG:
Wird bei Initialisierung ein DPB für ein programmierbares Gerät gebildet und ein ungültiger Datenträger gesetzt, muß der Antrieb "Datenträger geändert" angeben, wenn er diesem ungültigen Datenträger-Byte begegnet.

3. "BPB Aufbauen" mit Datenträger und Puffer aufrufen.

   Der Antrieb muß nun den gerade in der Einheit befindlichen Datenträger bestimmen und einen Anzeiger zu einer BPB-Tabelle senden. Diese Tabelle ist bei INIT nötig, um einen korrekten DPB für die Einheit zu bilden (es wird jedoch kein Zwischenraum erstellt, in der Annahme, daß genug Platz reserviert wurde). Stellt sich heraus, daß das bestimmte Datenträger-Deskriptor-Byte in der Tabelle dasselbe ist wie das gesendete, bildet DOS keine neue Tabelle und benutzt stattdessen die bestehende. Deshalb braucht der Antrieb in diesem Fall nicht die anderen Eintragungen zu machen.

   Der Aufruf "BPB Aufbauen" setzt auch einen Anzeiger zu einem Puffer mit einem Sektor. Das NONIBM-FORMAT-Bit im Attributfeld bestimmt den Inhalt dieses Puffers. Wenn das NONIBM-FORMAT-Bit gesetzt ist, zeigt der Anzeiger auf einen 1-Sektor-Arbeitsbereich, der für beliebige Zwecke benutzt werden kann. Ist das Bit Null (Gerät ist IBM-Format kompatibel), enthält der Puffer den ersten Sektor der FAT. Das FAT-ID-Byte ist das erste Byte dieses Puffers. Die Stelle der FAT muß für alle Datenträger gleich sein, denn der erste FAT-Sektor muß gelesen werden, BEVOR der eigentliche BPB zurückgegeben wird.

Wenn bei Rückkehr das Fehlerkennzeichen im Statuswort gelöscht ist, wird DPB gesetzt, und die Funktion ist ausgeführt. Ist das Fehlerkennzeichen gesetzt, hätte BIOS unbekannte Datenträger setzen sollen.

E.6 STATISCHER ANFORDERUNGSVORSATZ

Der Datenblock für die Geräteantrieb-Unterbrechung besteht aus zwei Teilen: einem statischen Anforderungsvorsatz, der das gleiche Format für alle Anfragen hat, und einem Teil, der anforderungsspezifische Information enthält. Der zweite Teil ist von variabler Größe und Format. Der Antrieb bestimmt eine Eingangsstelle, und DOS sendet einen Anzeiger zum Datenblock in ES:BX.

ANMERKUNG:
Der Antrieb ist für die Erhaltung des Maschinenstatus zuständig.

Struktur des statischen Anforderungsvorsatzes:

BYTE Satzlänge Länge dieser Antriebsanforderungs-Struktur in Bytes
BYTE Einheitscode Untereinheit, für die der Vorgang bestimmt ist (Nebengerät). (Bei Zeichengeräten ohne Bedeutung.)
BYTE Befehlscode
WORT Status
8 Bytes hier für zwei DWORT-Verknüpfungen (DOS- und Geräte-warteschlange) reserviert.

Struktur des Statuswortes:

15	14 13 12 11 10	9	8	7 6 5 4 3 2 1 0
ERR	RESERVIERT	BUI	DON	FEHLERCODE (Bit 15 gesetzt)

Das Statuswort ist bei Eintritt Null. Das Antriebs-Dienstprogramm setzt es bei Rückkehr. Bit 8 ist das fertiggestellte (done) Bit, das auf Beendung des Vorgangs hinweist. Das Statuswort muß vorläufig immer vom Antriebscode gesetzt werden. Zukünftig wird es vom Dienstprogrammm gesetzt werden, um DOS die Beendung des Vorgangs zu melden.

Bit 15 ist das Fehlerbit. Wenn es gesetzt ist, zeigen die 8 wertniedrigsten Bits den Fehler folgendermaßen an:

0 Schreibschutzbruch
1 Unbekannte Einheit
2 Laufwerk nicht bereit
3 Unbekannter Befehl
4 CRC-Fehler
5 Falsche Strukturlänge der Antriebsanforderung
6 Fehler suchen
7 Unbekannter Datenträger
8 Sektor nicht gefunden
9 Kein Papier im Drucker
A Schreibfehler
B Lesefehler
C Allgemeines Versagen

Bit 9 ist das besetzte Bit, das nur durch Statusaufrufe gesetzt wird:

Zur Ausgabe auf Zeichengeräten: 1 bedeutet, daß eine Schreibanforderung auf Beendung einer gerade laufenden Anforderung wartet. 0 bedeutet, daß keine aktuelle Anforderung besteht, und eine Schreibanforderung sofort beginnen kann.

Zur Eingabe auf Zeichengeräten mit einem Puffer: 1 bedeutet, daß eine Leseanforderung an das physische Gerät geleitet wird. 0 bedeutet, daß Zeichen im Gerätepuffer sind und ein Ablesevorgang unmittelbar zurückgesendet wird. Es bedeutet weiterhin, daß der Anwender etwas eingetragen hat.

DOS setzt voraus, daß alle Zeichengeräte über einen Eingabe-Type-Ahead-Puffer verfügen. Geräte ohne diesen Puffer sollten immer 0 für das besetzte Bit senden. Ist dies nicht der Fall, wird DOS u. U. blockiert, während es auf Eingabe in einen nicht bestehenden Puffer wartet.

E.7 DATENBLOCKFORMATE

Dieser Abschnitt enthält die Datenblockformate für die verschiedenen Antriebsfunktionen:

E.7.1 Lesen oder Schreiben - ES:BX (einschließlich IOCTL)

13 BYTE Statischer Anforderungsvorsatz
BYTE Datenträger-Deskriptor vom DPB (bei Zeichengeräten ohne Bedeutung)
DWORT Übertragungsadresse
WORT Byte/Sektor Zählung
WORT Anfangssektornummer (bei Zeichengeräten ohne Bedeutung)

Der Antrieb muß außer dem Statuswort die Byte-Sektor-Zählung entsprechend der Anzahl der tatsächlich übertragenen Bytes oder Sektoren setzen.

ANMERKUNG:
Bei einem IOCTL E/A-Aufruf wird keine Fehlerprüfung ausgeführt. Der Antrieb muß jedoch die Rückkehr-Sektor(Byte)-Zählung gemäß der Anzahl der tatsächlich übertragenen Bytes korrekt setzen.

E.7.2 Nicht zerstörendes Lesen ohne Warten - ES:BX

13 BYTE Statischer Anforderungsvorsatz
BYTE vom Gerät gelesen

Sendet das Zeichengerät das besetzte Bit = 0 (Zeichen im Puffer), wird das nächstzulesende Zeichen gesendet. Dieses Zeichen wird NICHT vom Eingabepuffer entfernt (daher der Ausdruck "nicht zerstörendes Lesen"). Mit diesem Aufruf kann DOS das nächste Eingabezeichen feststellen.

E.7.3 Datenträgerprüfung - ES:BX

13 BYTE Statischer Anforderungsvorsatz
BYTE Datenträger-Deskriptor vom DPB
BYTE gesendet

Außer dem Statuswort muß der Antrieb das Rückkehr-Byte folgendermaßen setzen:

Rückkehr-Byte	-1	Datenträger geändert
	0	Unbekannt, ob Datenträger geändert
	1	Datenträger unverändert

E.7.4 BPB aufbauen - ES:BX

13 BYTE Statischer Anforderungsvorsatz
BYTE Datenträger-Deskriptor vom DPB
DWORT Übertragungsadresse (zeigt auf einen 1-Sektor-Arbeitsbereich oder dem ersten Sektor der FAT, entsprechend dem Wert des NONIBM FORMAT-Bits)
DWORT Anzeiger auf BPB

Wenn das NONIBM-Format-Bit des Gerätes gesetzt ist, zeigt die DWORT-Übertragungsadresse auf einen 1-Sektor-Puffer, der zu beliebigem Zweck benutzt werden kann. Ist das NONIBM-FORMAT-Bit 0, enthält dieser Puffer den ersten Sektor der FAT. In diesem Fall darf der Puffer nicht durch den Antrieb geändert werden. (Dieser Modus ist nützlich, wenn lediglich das FAT-ID-Byte abgelesen werden soll.)

Bei IBM-kompatiblem Format (NONIBM-Format-Bit = 0), muß der erste Sektor der ersten FAT in allen Datenträgern im gleichen Sektor sein, weil der Fat-Sektor gelesen wird, BEVOR der Datenträger bestimmt ist.

Außer dem Statuswort muß der Antrieb bei Rückkehr den Anzeiger auf den BPB setzen.

Die BPB bezügliche Information für einen bestimmten Datenträger wird im Einschalt-Sektor gespeichert. Das Format für den Einschalt-Sektor ist:

	3 BYTE Nah-Sprung auf Einschaltcode
	8 BYTES OEM-Name und -Version
B P B	WORT Bytes pro Sektor
	BYTE Sektoren pro Zuordnunsgseinheit
	WORT Reservierte Sektoren für Start
	BYTE Anzahl der FATs
	WORT Anzahl der Ursprungsverzeichnis-Eintragungen
	WORT Anzahl der Sektoren in logischem Bild
B P B	BYTE Datenträger-Deskriptor
	WORT Anzahl der FAT Sektoren
	WORT Sektoren pro Spur
	WORT Anzahl der Köpfe
	WORT Anzahl verborgener Dateien
	WORT Anzahl reservierter Sektoren für Start, FATS und Verzeichnis

Die Werte im Start-Sektor, die von den verschiedenen Diskettenformaten abhängen, sind in Abschnitt A.4.3 enthalten.

Die drei Wörter am Ende sind optional und werden von DOS nicht benötigt, da sie nicht Teil des BPBs sind. Sie helfen BIOS, den Datenträger zu verstehen. Sektoren pro Spur sind u. U. überflüssig (können anhand der Gesamtgröße der Disk berechnet werden). Die Kopfanzahl ist zur Unterstützung verschiedener Multi-Kopflaufwerke nützlich, die die gleiche Speicherkapazität, aber unterschiedliche Oberflächenanzahl haben. Die Anzahl verborgener Sektoren dient zur Unterstützung von Laufwerkunterteilungsplänen.

Gegenwärtig werden die Kennzeichen-Bits des Datenträger-Deskriptor-Bytes folgendermaßen bestimmt:

Bit	Bedeutung
0	Ein = doppelseitig Aus = einseitig
1	Ein = 16 Sektoren pro Spur Aus = 18 Sektoren pro Spur
2	Ein = 512-Byte Sektoren Aus = 256-Byte Sektoren

E.7.5 Statusaufrufe - ES:BX

13 BYTE Statischer Anforderungsvorsatz

Der Antrieb braucht nur das Statuswort zu setzen. Siehe Abschnitt E.6.

E.7.6 Flush Aufrufe - ES:BX

13 BYTE Statischer Anforderungsvorsatz

Dieser Aufruf weist den Antrieb an, alle ihm bekannten ausstehenden Anforderungen zu löschen. Der Aufruf dient hauptsächlich dazu, die Eingabewarteschlange bei Zeichengeräten zu leeren.

E.7.7 Init - ES:BX

13 BYTE Statischer Anforderungsvorsatz
BYTE Anzahl der Einheiten (nicht von Zeichengeräten benutzt)
DWORT Unterbrechungsadresse
DWORT Anzeiger auf BPB-Gruppe (nicht von Zeichengeräten gesetzt)

Der Antrieb setzt Einheitsanzahl, Unterbrechungsadresse und BPB-Anzeiger.

E.8 CONFIG.SYS ZUM INSTALLIEREN EINES ANTRIEBS MODIFIZIEREN

Die WANG PC Systemdiskette enthält eine sog. CONFIG.SYS Datei. CONFIG.SYS verfügt über eine Liste installationsspezifischer Parameter, die bei Systemstart benutzt werden. Um die Systemkonfiguration um ein Gerät zu erweitern, muß folgende Zeile zu CONFIG.SYS über den Editor hinzugefügt werden:

GERÄT = <Antriebsname>

<Antriebsname> ist der Geräteantriebs-Dateiname. Anstelle von GERÄT kann ANTRIEB als Schlüsselwort benutzt werden. Näheres über CONFIG.SYS ist in Abschnitt A.3 enthalten.

Geräteantrieb

Nachdem CONFIG.SYS aufbereitet ist, muß es und der Geräteantrieb auf die Systemdisk kopiert und das System neu gestartet werden. DOS installiert den Antrieb, sobald die Copyright-Anmerkung erscheint. Findet DOS die Antriebsdatei nicht, erscheint die Mitteilung, daß diese ungültig ist oder fehlt. Wird die Datei gefunden, lädt DOS sie in den unteren Speicher und ruft die INIT-Routine sofort auf.

Die INIT-Routine sollte Antriebsnamen und Versionsnummer auf dem Bildschirm anzeigen, damit der Anwender weiß, daß ein bestimmtes Gerät installiert worden ist.

WARNUNG:
CONFIG.SYS muß mit größter Vorsicht modifiziert werden; bei einem Fehler ist es u. U. nicht möglich, das System einzuschalten. Immer Kopien des ursprünglichen CONFIG.SYS aufheben.

ANHANG F
LOGISCHES TASTATURPROTOKOLL

F.1 EINFÜHRUNG

Dieser Anhang erklärt die logischen Tastenanschläge, die bei allen auf WANG PC laufenden Anwendungsprogrammen verfügbar sind. Logische Tastenanschläge sind unabhängig von den verschiedenen physischen Tastaturen, die für den WANG PC erhältlich sind. Die BIOS.SYS Datei enthält Tabellen zum Übersetzen von Codes, die von einer bestimmten physischen Tastatur empfangen werden, in die logischen Tastenanschläge, die in diesem Anhang beschrieben sind. Deshalb sind Angaben in den folgenden Tabellen für alle Sprachen gültig, für die WANG PC Tastaturen bestehen.

Es gibt drei Kategorien logischer Tastenanschläge: einfache Zeichen, Sonderzeichen und Tot-Tasten. 16-Bit-Codes bestimmen die Tastenanschläge. Bei einfachen Zeichen ist das werthöchste Byte 00H. Bei Sonderzeichen ist das werthöchste Byte 1FH, bei Tot-Tasten 1EH. Das System speichert das werthöchste Byte zuerst, sendet jedoch das Byte 00H nicht an den Tastaturpuffer.

F.2 EINFACHE ZEICHEN

Einfache Zeichen umfassen den druckbaren WISCII I Zeichensatz von 20H bis FDH und Steuerzeichen zwischen 01H und 1BH.

F.2.1 WISCII I Zeichensatz

Der WISCII I Zeichensatz (Tabelle F-1) ist der Standard- (nicht-wissenschaftliche) WISCII Zeichensatz von 20H bis FDH.

Tabelle F-1. WISCII I Zeichensatz

Dezimal	Hex	Zeichen	Beschreibung
032	20		Leerschritt
033	21	!	Ausrufezeichen
034	22	"	Doppel-Anführungszeichen
035	23	#	Nummernzeichen
036	24	$	Dollarzeichen
037	25	%	Prozentzeichen
038	26	&	Et-Zeichen
039	27	'	Einzel-Anführungszeichen
040	28	(	Linke runde Klammer
041	29	)	Rechte runde Klammer
042	2A	*	Stern
043	2B	+	Pluszeichen
044	2C	,	Komma
045	2D	-	Minuszeichen
046	2E	.	Punkt
047	2F	/	Schrägstrich
048	30	0	0
049	31	1	1
050	32	2	2
051	33	3	3
052	34	4	4
053	35	5	5
054	36	6	6
055	37	7	7
056	38	8	8
057	39	9	9
058	3A	:	Doppelpunkt
059	3B	;	Semikolon
060	3C	<	Kleiner-Zeichen
061	3D	=	Gleichheitszeichen
062	3E	>	Größer-Zeichen
063	3F	?	Fragezeichen
064	40	@	At-Zeichen
065	41	A	A
066	42	B	B
067	43	C	C
068	44	D	D
069	45	E	E
070	46	F	F
071	47	G	G
072	48	H	H

Tabelle F-1. WISCII I Zeichensatz (Forts.)

Dezimal	Hex	Zeichen	Beschreibung
073	49	I	I
074	4A	J	J
075	4B	K	K
076	4C	L	L
077	4D	M	M
078	4E	N	N
079	4F	O	O
080	50	P	P
081	51	Q	Q
082	52	R	R
083	53	S	S
084	54	T	T
085	55	U	U
086	56	V	V
087	57	W	W
088	58	X	X
089	59	Y	Y
090	5A	Z	Z
091	5B	[	Linke eckige Klammer
092	5C	\	Gegenstrich
093	5D	]	Rechte eckige Klammer
094	5E	↑	Aufwärtspfeil
095	5F	_	Unterstreichung
096	60	'	Anführungszeichen vorne
097	61	a	a
098	62	b	b
099	63	c	c
100	64	d	d
101	65	e	e
102	66	f	f
103	67	g	g
104	68	h	h
105	69	i	i
106	6A	j	j
107	6B	k	k
108	6C	l	l
109	6D	m	m
110	6E	n	n
111	6F	o	o
112	70	p	p
113	71	q	q
114	72	r	r

Tabelle F-1. WISCII I Zeichensatz (Forts.)

Dezimmal	Hex	Zeichen	Beschreibung
115	73	s	s
116	74	t	t
117	75	u	u
118	76	v	v
119	77	w	w
120	78	x	x
121	79	y	y
122	7A	z	z
123	7B	{	Linke geschweifte Klammer
124	7C	|	Vertikalstrich
125	7D	}	Rechte geschweifte Klammer
126	7E	~	Ungefähr
127	7F	¢	Cent
128	80	°	Grad
129	81	◆	Zentriersymbol
130	82	▶	Tab
131	83	◀	Zeilenschaltsymbol
132	84	→	Einrücksymbol
133	85	_	Dezi-Tab-Symbol
134	86	|	Format-Symbol
135	87	■	Stop-Symbol
136	88	!!	Notiz-Symbol
137	89	↕	Misch-Symbol
138	8A	↓	Abwärtspfeil
139	8B	↑	Aufwärtspfeil
140	8C	←	Linker Pfeil
141	8D	±	Plus/Minus-Zeichen
142	8E	¡	Umgekehrtes Ausrufezeichen
143	8F	¿	Umgekehrtes Fragezeichen
144	90	Â	A Zirkumflex
145	91	À	A Grave
146	92	Á	A Aigu
147	93	Ä	A Umlaut
148	94	Ã	A Tilde
149	95	Ö	Linker Pfeil
150	96	Å	A Angstrom
151	97	ö	Abwärtspfeil
152	98	Æ	AE Ligatur
153	99	C	C Cedille
154	9A	‡	Doppelkreuz
155	9B	•	Fetter Punkt

Tabelle F-1. WISCII I Zeichensatz (Forts.)

Dezimal	Hex	Zeichen	Beschreibung
156	9C	Ê	E Zirkumflex
157	9D	È	E Grave
158	9E	É	E Aigu
159	9F	Ë	E Tilde
160	A0	â	a Zirkumflex
161	A1	à	a Grave
162	A2	á	a Aigu
163	A3	ä	a Umlaut
164	A4	ã	a Tilde
165	A5	→	Rechter Pfeil
166	A6	a	a Angstrom
167	A7	†	Kreuz
168	A8	æ	ae Ligatur
169	A9	ç	c Cedille
170	AA	□	Offenes Quadrat
171	AB		Unveränderlicher Leerschritt
172	AC	ê	e Zirkumflex
173	AD	è	e Grave
174	AE	é	e Aigu
175	AF	ë	e Tilde
176	B0	Ğ	G Hacek
177	B1	IJ	IJ Ligatur
178	B2	I	I mit Punkt
179	B3	Î	I Zirkumflex
180	B4	Ì	I Grave
181	B5	Í	I Aigu
182	B6	Ï	I Trema
183	B7	L·L	Katalanische LL Ligatur
184	B8	Ñ	N Tilde
185	B9	Ô	O Zirkumflex
186	BA	Ò	O Grave
187	BB	Ó	O Aigu
188	BC	Ö	O Umlaut
189	BD	Õ	O Tilde
190	BE	Œ	OE Ligatur
191	BF	Ø	O mit Schrägstrich
192	C0	ğ	g Hacek
193	C1	ij	ij Ligatur
194	C2	ı	i ohne Punkt
195	C3	î	i Zirkumflex
196	C4	ì	i Grave
197	C5	í	i Aigu

Tabelle F-1. WISCII I Zeichensatz (Forts.)

Dezimal	Hex	Zeichen	Beschreibung
198	C6	ï	i Trema
199	C7	ŀ	Katalanische ll Ligatur
200	C8	ñ	n Tilde
201	C9	ô	o Zirkumflex
202	CA	ò	o Grave
203	CB	ó	o Aigu
204	CC	ö	o Umlaut
205	CD	õ	o Tilde
206	CE	œ	oe Ligatur
207	CF	ø	o mit Schrägstrich
208	D0	Þ	Isländischer Thorn
209	D1	Ð	Isländisches Eth
210	D2	Ý	Y Aigu
211	D3	Ş	S Cedille
212	D4	‘	Anführungszeichen vorne
213	D5	Û	U Zirkumflex
214	D6	Ù	U Grave
215	D7	Ú	U Aigu
216	D8	Ü	U Umlaut
217	D9	©	Urheberrecht-Symbol
218	DA	®	Eingetragenes Warenzeichen
219	DB	℞	Rezept-Symbol
220	DC	ª	hochgestelltes a (bei Nummernabkürzung)
221	DD	«	Europ. Anführungszeichen vorne
222	DE	§	Paragraphenzeichen
223	DF	¶	Absatzsymbol
224	E0	þ	Isländischer Thorn
225	E1	ð	Isländisches Eth
226	E2	ý	y Aigu
227	E3	ş	s Cedille
228	E4	’	Anführungszeichen hinten
229	E5	û	u Zirkumflex
230	E6	ù	u Grave
231	E7	ú	u Aigu
232	E8	ü	u Umlaut
233	E9	™	Warenzeichen
234	EA	¤	Internationales Währungssymbol
235	EB	↔	Doppelpfeil
236	EC	º	hochgestelltes o (bei Nummernabkürzung)

Tabelle F-1. WISCII I Zeichensatz (Forts.)

Dezimal	Hex	Zeichen	Beschreibung
237	ED	»	Europ. Anführungszeichen hinten
238	EE	ß	Beta
239	EF	·	Zentrierter Punkt
240	F0	£	Pfund Sterling
241	F1	ƒ	Florin
242	F2	¥	Yen
243	F3	¼	1/4
244	F4	½	1/2
245	F5	¾	3/4
246	F6	ˆ	Zirkumflex
247	F7	`	Grave
248	F8	´	Aigu
249	F9	¨	Umlaut (Diäresis)
250	FA	˜	Tilde
251	FB	¸	Cedille
252	FC	ˇ	Hacek
253	FD	˘	Kürzezeichen

F.2.2. Steuerzeichen

Zum Erstellen von Steuerzeichen, KTRL und eine andere Taste gleichzeitig drücken. Außer bei KTRL + [, bei dem [umgeschaltet sein muß, resultiert dasselbe Steuerzeichen, ob die zweite Taste umgeschaltet ist oder nicht. Tabelle F-2 enthält die Steuerzeichen von 01H bis 1BH.

Tabelle F-2. Steuerzeichen

Hex Wert	Tastenfolge
01	KTRL + A
02	KTRL + B
03	KTRL + C oder UMSCH + ANNULLIER
04	KTRL + D
05	KTRL + E
06	KTRL + F
07	KTRL + G
08	KTRL + H oder Rückschritt
09	KTRL + I oder Tab
0A	KTRL + J

Tabelle F-2. Steuerzeichen (Forts.)

Hex Wert	Tastenfolge
0B	KTRL + K
0C	KTRL + L
0D	KTRL + M oder Zeilenschaltung
0E	KTRL + N
0F	KTRL + O
10	KTRL + P
11	KTRL + Q
12	KTRL + R
13	KTRL + S
14	KTRL + T
15	KTRL + U
16	KTRL + V
17	KTRL + W
18	KTRL + X
19	KTRL + Y
1A	KTRL + Z
1B	KTRL + [

F.3 SONDERZEICHEN

Sonderzeichen enthalten verschiedene Codes wie Funktionstasten, Cursorsteuertasten, LÖSCH, HILFE, ANNULLIER und DRUCK. Es sollte jedoch beachtet werden, daß UMSCH, FEST UMSCH, KTRL und EXTRA keine zugeordneten Codes haben.

Es gibt 64 mögliche logische Funktionstasten bei Codes 1F80H bis 1FBF. Von diesen sind nur die ersten 32 (von 1F80H bis 1F9FH) definiert. Codes 1F61H bis 1F7AH sind auch undefiniert. Tabelle F-3 enthält die übrigen Sonderzeichen mit ihren umgeschalteten oder nicht umgeschalteten Hexadezimal-Werten.

Tabelle F-3. Sonderzeichentasten

Taste	Nicht umgeschaltet	Umgeschaltet
Annullieren	1FE0	0003
Hilfe	1FE1	1FF1
Glossar	1FE2	1FF2
Druck	1FE3	1FF3
Nordcursor	1FC0	1FD0
Südcursor	1FC1	1FD1
Ostcursor	1FC2	1FD2
Westcursor	1FC3	1FD3
Home	1FC4	1FD4
Ausführen	1FC5	1FD5
Einfügen	1FC6	1FD6
Löschen	1FC7	1FD7
Vorh.	1FC8	1FD8

Tabelle F-3. Sonderzeichentasten (Forts.)

Taste	Nicht umgeschaltet	Umgeschaltet
Nächs.	1FC9	1FD9
Löschen	1FCB	1FDB
Tab	0009	1FDC
Rücktab	1FCD	1FDD

F.4 TOT-TASTEN

Tot-Tastenfolgen sind Tastenanschläge, die - mit anderen Tastenanschlägen kombiniert - ein druckbares Zeichen erstellen. Tot-Tastencodes liegen zwischen 1E20 und 1EFF(Zwischencodes). Sie werden mit dem Code für kombinierte Tastenanschläge ersetzt, wenn die letzte Taste in der Folge gedrückt wird.

Die Wiederanlauftastenfolge illustriert den Gebrauch der Tot-Tasten. EXTRA + BEFEHL erstellt einen 1E Zwischencode. Der Zwischencode wird durch Druck von ANNULLIER mit dem Wiederanlaufcode ersetzt. Wird eine ungültige Tot-Tastenfolge eingegeben, empfängt der Tastaturpuffer den Code der zuletzt gedrückten Taste.

Eine Tastatur muß entweder in sichtbarem oder unsichtbarem Tot-Tastenmodus sein. Wenn Tot-Tasten in sichtbarem Modus gedrückt werden, werden die den Tasten entsprechenden Codes in den Puffer gesendet. Der Code kann mit einem druckbaren Symbol verbunden sein. Ist dies nicht der Fall, wird ein Tot-Tastenleerzeichen in den Puffer gesetzt. Wird die Tot-Taste in unsichtbarem Modus gedrückt, empfängt das Betriebssystem die Codes, speichert sie aber nicht im Puffer. Der einzige vom Puffer empfangene Code ist der kombinierte Code, der durch Druck der letzten Taste in der Tot-Tastenfolge generiert wird. Der Standard-Tastaturmodus für Tot-Tasten ist unsichtbar.

ANHANG G
BILDSCHIRMSTEUERUNG

G.1 EINFÜHRUNG

Wang BIOS erkennt eine Reihe von 1-Byte-Steuercodes und ANSI Standard-Umschaltcodefolgen, die spezifische Bildschirmvorgänge veranlassen. Abschnitt G.2 führt Steuercodes und Abschnitt G.3 Umschaltcodefolgen auf.

G.2 STEUERCODES

BIOS erkennt folgende 1-Byte-Steuercodes:

Code	Bedeutung
07H	Akustisches Signal
08H	Nicht zerstörender Rückschritt
09H	Cursor rechts (nicht zerstörend)
0AH	Zeilenvorschub (Scroll, falls notwendig)
0CH	Bildschirm leeren und Cursor Home
0DH	Wagenrücklauf
01BH	Umschaltcodefolgemodus

G.3 UMSCHALTCODEFOLGEN

In diesem Abschnitt sind die verschiedenen Umschaltcodefolgen, einschließlich ihrer Formate erklärt. Definitionen der Parameter in der Folge erscheinen nach den Formaten. Jede Folge endet mit einem Buchstaben, der die spezifische Umschaltcodefolge identifiziert.

In allen Folgen sind Parameter ASCII-codierte Dezimalzahlen mit den Ziffern 0 bis 9. BIOS ignoriert alle Leerzeichen im Feld und Steuerzeichen. Der Steuercode zum Zugriff auf den Umschaltcodefolgemodus ist 01BH.

G.3.1 Grafik-Attribute setzen

Funktion

Mit dieser Folge wird spätere Ausgabe mit den gewählten Attributen erstellt. Tabelle G-1 führt diese Attribute auf.

Format

ESC "[" Ps ";" ... ";" Ps "m"

Ps Werte von Tabelle G-1; Standardwert 0

Tabelle G-1. Grafik-Attribut-Umschaltcodefolgen

Wert	Attribute
0	Keine Attribute
1	Fettdruck
2	Tiefstellung
3	Hochstellung
4	Unterstreichen
5	Blinken
6	Überschreiben
7	Negativdarstellung
8	Unterdrückung
30	setzt Farbe 0 Vordergrund
31	setzt Farbe 1 Vordergrund
32	setzt Farbe 2 Vordergrund
33	setzt Farbe 3 Vordergrund
34	setzt Farbe 4 Vordergrund
35	setzt Farbe 5 Vordergrund
36	setzt Farbe 6 Vordergrund
37	setzt Farbe 7 Vordergrund
40	setzt Farbe 0 Hintergrund
41	setzt Farbe 1 Hintergrund
42	setzt Farbe 2 Hintergrund
43	setzt Farbe 3 Hintergrund
44	setzt Farbe 4 Hintergrund
45	setzt Farbe 5 Hintergrund
46	setzt Farbe 6 Hintergrund
47	setzt Farbe 7 Hintergrund
50	setzt Farbe 8 Vordergrund
51	setzt Farbe 9 Vordergrund
52	setzt Farbe 10 Vordergrund
53	setzt Farbe 11 Vordergrund
54	setzt Farbe 12 Vordergrund
55	setzt Farbe 13 Vordergrund
56	setzt Farbe 14 Vordergrund
57	setzt Farbe 15 Vordergrund

Tabelle G-1. Grafik-Attribut-Umschaltcodefolgen (Forts.)

Wert	Attribut
60	setzt Farbe 8 Hintergrund
61	setzt Farbe 9 Hintergrund
62	setzt Farbe 10 Hintergrund
63	setzt Farbe 11 Hintergrund
64	setzt Farbe 12 Hintergrund
65	setzt Farbe 13 Hintergrund
66	setzt Farbe 14 Hintergrund
67	setzt Farbe 15 Hintergrund
80	setzt Farbe 0 erhellt
81	setzt Farbe 1 erhellt
82	setzt Farbe 2 erhellt
83	setzt Farbe 3 erhellt
84	setzt Farbe 4 erhellt
85	setzt Farbe 5 erhellt
86	setzt Farbe 6 erhellt
87	setzt Farbe 7 erhellt
88	setzt Farbe 8 erhellt
89	setzt Farbe 9 erhellt
90	setzt Farbe 10 erhellt
91	setzt Farbe 11 erhellt
92	setzt Farbe 12 erhellt
93	setzt Farbe 13 erhellt
94	setzt Farbe 14 erhellt
95	setzt Farbe 15 erhellt

G.3.2 Standardbildschirm zum angegebenen Bildschirm schalten

Funktion

Diese Umschaltcodefolge wählt den angegebenen Bildschirm als den Standardbildschirm. Wird der angegebene Bildschirm nicht gefunden, bleibt der aktuelle Standardbildschirm erhalten.

Format

ESC "/" Ps "s"

Ps 1 = erster Bildschirm, 2 = zweiter Bildschirm usw.

G.3.3 Gewählte Modi und LED-Steuercodes setzen oder löschen

Funktion

Diese Folge löscht oder setzt die Modi in Tabelle G-2.

Format

ESC "[" Ps ";" ... ";" Ps {"h" / "l"}

Ps — Einer der Werte von Tabelle G-2; Standardwert ist 0.

"h" — Setzt gewählte Modi.

"l" — Löscht gewählte Modi.

Tabelle G-2. Bildschirmmodi und LED-Steuercodes

Wert	Modus
4	Einfügmodus. Bei aktivem Einfügmodus werden gedruckte Zeichen an der Cursorposition eingefügt.
5	Cursor ein/aus. l = ein, h = aus.
6	Soft Scroll (nur Niedrig-Auflösungskarte).
7	40/80 Spaltenmodus (nur Niedrig-Auflösungskarte). "h" wählt den 80-Spalten-, "l" den 40 Spalten-Modus.
8	Cursor-Anschlußmodus. In diesem Modus verschiebt sich der Cursor mit der Zeichenausgabe; ist der Modus gelöscht, bleibt der Cursor an seiner aktuellen Stelle.
9	Zeilenende-Autowrap-Modus. In diesem Modus wird der Cursor durch eine rechtswärtige Cursorbewegung vom rechten Rand auf den linken Rand der nächsten Zeile verschoben; ist der Modus gelöscht, wird das letzte Zeichen einer Zeile überschrieben.
10	Steuert LED 0 (FEST)
11	Steuert LED 1
12	Steuert LED 2
13	Steuert LED 3
14	Steuert LED 4
15	Steuert LED 5

ANMERKUNG:
Nur 80-Spalten-Monitoren lassen eine Auswahl von 40 oder 80 Zeichen pro Zeile zu. Im 80-Spalten-Modus können 4 Farben gleichzeitig angezeigt werden. Im 40-Spalten-Modus können bis zu 16 Farben gleichzeitig angezeigt werden.

G.3.4 Bildschirm ganz oder teilweise leeren

Funktion

Diese Folge löscht vom Bildschirmanfang bis zum Cursor, vom Cursor bis zum Bildschirmende oder leert den ganzen Bildschirm.

Format

ESC "[" Ps "J"

Ps einer der folgenden Werte (Standardwert ist 0):

Wert	Funktion
0	Löscht vom Cursor bis zum Bildschirmende (einschließlich des Zeichens über dem Cursor).
1	Löscht vom Bildschirmanfang bis zum Cursor.
2	Leert den ganzen Bildschirm und löscht den Home-Cursor.

G.3.5 Aktuelle Zeile ganz oder teilweise löschen

Funktion

Diese Folge löscht vom Cursor bis zum Zeilenende, vom Zeilenanfang bis zum Cursor oder die ganze Zeile.

Format

ESC "[" Ps "K"

Ps einer der folgenden Werte (Standardwert ist 0):

Wert	Funktion
0	Löscht vom Cursor bis zum Zeilenende (einschließlich des Zeichens über dem Cursor).
1	Löscht vom Zeilenanfang bis zum Cursor.
2	Löscht die ganze Zeile und Positionscursor bis Spalte 1 dieser Zeile.

G.3.6 Cursor an Reihe und Spalte setzen

Funktion

Diese Folge setzt den Cursor an eine bestimmte Reihe und Spalte.

Format:

ESC "[" Pr ";" Pc { "H" / "f" }

Pr — eine Zahl von 1 bis 24, die eine Reihe bezeichnet; der Standardwert für die obere Reihe ist 1.

Pc — eine Zahl von 1 bis 40 oder 1 bis 80 (je nach Breite des Bildschirmes), die eine Spalte bezeichnet. Der Standardwert für die linke Spalte ist 1.

ANMERKUNG:
"H" und "f" führen zum gleichen Ergebnis.

G.3.7 Cursor nach oben, unten, rechts oder links verschieben

Funktion

Diese Folgen verschieben den Cursor in einer bestimmten Richtung um eine angegebene Anzahl von Spalten oder Reihen.

Format

ESC "[" Pn { "A" / "B" / "C" / "D" }

Pn — Die Anzahl der Reihen oder Spalten, um die sich der Cursor verschiebt; Standardwert ist 1.

"A" — Verschiebt Cursor nach oben.

"B" — Verschiebt Cursor nach unten

"C" — Verschiebt Cursor nach rechts.

"D" — Verschiebt Cursor nach links.

G.3.8 Cursorposition speichern/Cursor an gespeicherte Position setzen

Funktion

Diese Folge speichert die aktuelle Cursorposition oder setzt den Cursor wieder an eine vorher gespeicherte Position.

Format

ESC "[" { "s" / "u" }

"s" Speichert die aktuelle Cursorposition.

"u" Setzt den Cursor wieder an die vorher gespeicherte Position.

ANMERKUNG:
Wenn die zweite Speicherfolge benutzt wird, bevor der Cursor wieder an die vorher gespeicherte Position gesetzt wird, überschreibt die zweite Speicherfolge die Information der ersten.

G.3.9 Zeile(n) bei aktueller Zeile einfügen

Funktion

Diese Folge fügt eine bestimmte Anzahl von Zeilen in der aktuellen Cursorzeile ein.

Format

ESC "[" Pn ";" Ps "L"

Pn Anzahl der einzufügenden Zeilen; der Standardwert ist 1.

Ps Beliebige Zahl. Bei 0 rückt die Folge den unteren Teil des Bildschirmes herunter. Bei einer Nicht-Null-Zahl verschiebt die Folge den oberen Teil des Bildschirmes nach oben.

G.3.10 Zeile(n) bei aktueller Zeile löschen

Funktion

Diese Folge löscht eine bestimmte Anzahl von Zeilen, die bei der aktuellen Cursorzeile beginnen.

Format

ESC "[" Pn ";" Ps "M"

Pn — Anzahl der zu löschenden Zeilen; der Standardwert ist 1.

Ps — Beliebige Zahl. Bei 0 (Standardwert) rückt die Folge den unteren Teil des Bildschirmes nach unten. Bei einer Nicht-Null-Zahl verschiebt die Folge den oberen Teil des Bildschirmes nach oben.

G.3.11 Teil des Bildschirmes abrollen lassen

Funktion

Diese Folge rollt einen bestimmten Teil des Bildschirmes um eine angegebene Anzahl von Zeilen nach oben oder unten.

Format

ESC "/" Ps ";" Pe ";" Pc ";" Pd "S"

Ps — Die Nummer der ersten abzurollenden Zeile; muß kleiner als Pe sein. Es gibt keinen Standardwert.

Pe — Die Nummer der letzten abzurollenden Zeile. Es gibt keinen Standardwert.

Pc — Die Anzahl der Zeilen, um die der bestimmte Bildschirmteil abgerollt werden soll. Der Standardwert ist 1.

Pd — Beliebige Zahl. Bei 0 (Standardwert) rollt der Bildschirm nach unten ab, bei einer Nicht-Null-Zahl rollt der Bildschirm nach oben ab.

G.3.12 Zeichen von aktueller Zeile löschen

Funktion

Diese Folge beginnt, an der aktuellen Cursorposition eine bestimmte Zeichenanzahl von einer Zeile zu löschen.

Format

ESC "[" Pn "P"

Pn Anzahl der zu löschenden Zeichen.

G.3.13 Auf Start-Konfiguration zurücksetzen

Funktion

Diese Folge setzt den Standardbildschirm auf die Start-Konfiguration zurück.

Format

ESC "[" "z"

G.3.14 Paletten-Eintragsattribute setzen

Funktion

Diese Folge setzt bestimmte Attribute für die gewählte Paletteneintragung (nur für Niedrig-Auflösungskarte).

Format

ESC "/" Ps ";" Pi ";" Pr ";" Pg ";" Pb "P"

Ps Eine Zahl von 0 bis 15, die eine Paletteneintragung anzeigt. Paletteneintragungen bei Systemstart folgen:

Eintrag	Farbe
0	Schwarz
1	Dunkelgrau
2	Weiß (hellgrau)
3	Weiß (für Hochintensität oder Fettdruck)
4	Blau
5	Hellblau
6	Braun
7	Gelb
8	Grün
9	Hellgrün
10	Magentarot
11	Helles Magentarot
12	Zyanblau
13	Helles Zyanblau
14	Rot
15	Hellrot

ANMERKUNG:
Eine Zahl von 0 bis 7 wählt die Attribute für die übrigen Parameter ab. Eine Ziffer von 8 bis 15 wählt das Attribut an. Es gibt keinen Standardwert. Tabelle G-3 führt die möglichen Attributkombinationen auf, wobei 8 das angewählte und 0 das abgewählte Attribut ist.

Pi	Intensität
Pr	Rot
Pg	Grün
Pb	Blau

Tabelle G-3. Kombinationen der Paletten-Attribute

Farbresultat	Intensität	Rot	Grün	Blau
Schwarz	0	0	0	0
Dunkelgrau	8	0	0	0
Weiß (hellgrau)	0	8	8	8
Weiß	8	8	8	8
Blau	0	0	0	8
Hellblau	8	0	0	8
Braun	0	8	8	0
Gelb	8	8	8	0
Grün	0	0	8	0
Hellgrün	8	0	8	0
Magentarot	0	8	0	8
Helles Magentarot	8	8	0	8
Zyanblau	0	0	8	8
Helles Zyanblau	8	0	8	8
Rot	0	8	0	0
Hellrot	8	8	0	0

ANHANG H
DRUCKER-ZEICHENSÄTZE, CODES UND UMSCHALTCODEFOLGEN

H.1 EINFÜHRUNG

In diesem Anhang befindet sich WANG PC Drucker-Information für Programmierer. Zeichensätze, Steuercodes, Umschaltcodefolgen und Rückkehrcodes für jeden Drucker werden aufgeführt. Näheres über den Typenraddrucker-Zeichensatz ist in Abschnitt H.2, über den Matrixdrucker-Zeichensatz in Abschnitt H.3 enthalten.

Mit Funktionsaufruf 44H (Ein-/Ausgabesteuerung), Subfunktion 3, kann festgestellt werden, ob der Typenraddruckerantrieb geladen ist. Ist dies der Fall, wird die Mitteilung "PC DW20 Druckerantrieb" in den Speicher gelesen. Das Format des Ein-/Ausgabesteuerbefehls lautet:

Byte 0FFH (der Befehl zur Anzeige der identifizierenden Mitteilung)

Dwort Segment und Adresse der Mitteilungsposition

Wort Länge des für die Mitteilung reservierten Speichers. Die Länge sollte mindestens 20 Bytes sein. Bei Rückkehr reflektiert dieses Wort die tatsächliche Länge der Mitteilung; die angezeigte Längenangabe geht jedoch nicht über die eingetragene hinaus.

H.2 TYPENRADDRUCKER

Dieser Abschnitt gibt Zeichensatz, Steuercodes, Umschaltcodefolgen und Rückkehrcodes für den Typenraddrucker an (siehe Wang Professional Computer Typenraddruckerhandbuch).

H.2.1 Typenraddrucker-Zeichensatz

Der Wang Typenraddrucker benutzt für Textsymbole den Zeichensatz WISCII I (ohne FF). Der WISCII I Zeichensatz wird in Tabelle H-1 dargestellt.

Tabelle H-1. WISCII I Zeichensatz

Wertniedrigste 4 Bit Hex-Ziffer / Werthöchste 4 Bit Hex-Ziffer		$0\,0\,0\,0$	$0\,0\,0\,1$	$0\,0\,1\,0$	$0\,0\,1\,1$	$0\,1\,0\,0$	$0\,1\,0\,1$	$0\,1\,1\,0$	$0\,1\,1\,1$	$1\,0\,0\,0$	$1\,0\,0\,1$	$1\,0\,1\,0$	$1\,0\,1\,1$	$1\,1\,0\,0$	$1\,1\,0\,1$	$1\,1\,1\,0$	$1\,1\,1\,1$
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0			LEERSTELLE	0	@	P	`	p	°	Â	â	Ğ	ğ	Þ	þ	£
0001	1			!	1	A	Q	a	q	♦	À	à	Ŭ	ij	Đ	ð	*f*
0010	2			"	2	B	R	b	r	▶	Á	á	İ	ı	Ý	ý	¥
0011	3			#	3	C	S	c	s	◀	Ä	ä	Î	î	Ş	ş	¼
0100	4			$	4	D	T	d	t	→	Ã	ã	Ì	ì	‘	’	½
0101	5			%	5	E	U	e	u	⌙	←	→	Í	í	Û	û	¾
0110	6			&	6	F	V	f	v	|	Å	å	Ï	ï	Ù	ù	ˆ
0111	7			’	7	G	W	g	w	■	↓	†	L·L	I·I	Ú	ú	`
1000	8			(	8	H	X	h	x	!!	Æ	æ	Ñ	ñ	Ü	ü	´
1001	9			)	9	I	Y	i	y	↕	Ç	ç	Ô	ô	©	™	¨
1010	A			*	:	J	Z	j	z	↓	‡	□	Ò	ò	®	¤	~
1011	B			+	;	K	[	k	{	↑	●	Unveränderl. Leerstelle	Ó	ó	₧	↔	¸
1100	C			,	<	L	\	l	|	←	Ê	ê	Ö	ö	ª	º	˛
1101	D			-	=	M	]	m	}	±	È	è	Õ	õ	«	»	˝
1110	E			.	>	N	↑	n	~	¡	É	é	Œ	œ	§	ß	LEERSTELLE VOR 5 EINHEITEN
1111	F			/	?	O	_	o	¢	¿	Ë	ë	Ø	ø	¶	.	TOT-TASTE

H.2.2 Typenraddrucker-Steuercodes und Umschaltcodefolgen

Folgende 1-Byte-Steuercodes werden von Typenraddruckerantrieben erkannt:

Code	Bedeutung
07H	Akustisches Signal
08H	Nichtzerstörender Rückschritt
09H	Horizontaler Tab
0AH	Zeilenvorschub
0BH	Vertikaler Tab
0CH	Formularzuführung
0DH	Wagenrücklauf
01BH	Umschaltcodefolgemodus

Tabelle H-2 führt die vom Typenraddruckerantrieb erkannten Standard-Umschaltcodefolgen auf. Die Umschaltcodefolge (ESC) ist 01BH. Für alle folgenden ist der Parameter (P) eine ASCII-codierte Dezimalziffer von 0 bis 9 mit den Vorzeichen + oder - . Falls nichts anderes angegeben ist, stellt P Tausendstel eines Zolls dar. Der Standardwert für Ziffern ohne Vorzeichen ist positiv. Alle Leer- (020H) und Steuerzeichen werden in einer Umschaltcodefolge ignoriert.

Tabelle H-2. Typenraddrucker-Umschaltcodefolgen

Folge	Bedeutung
ESC "[" "z"	Stellt Drucker zurück
ESC "[" "a"	Stellt Standardwert wieder her
ESC "[" "b"	Leert Walze ohne neue Papierzufuhr
ESC "[" Ps "c"	Setzt linken Rand
ESC "[" Ps "d"	Setzt Zeilenvorschubabstand
ESC "[" Ps "e"	Setzt Formularlänge
ESC "[" Ps "i"	Setzt teilweisen Zeilenvorschub

Tabelle H-2. Typenraddrucker-Umschaltcodefolgen (Forts.)

Folge	Bedeutung
ESC "[" Ps "v"	Wählt Papierbehälter Ps
ESC "[" Pi "o"	Wählt Zeichen-pro-Zoll in tausendstel Zoll. Wählt Proportionalschrift, wenn Pi 0 ist
ESC "[" Ps ";"...";" Ps "p"	Setzt horizontale Tabpositionen in tausendstel Zoll ab linkem Rand
ESC "[" Ps ";"...";" Ps "q"	Setzt vertikale Tabpositionen in tausendstel Zoll ab Seitenanfang
ESC "[" ";"...";" Ps "h"	Setzt die durch den Wert von Ps gewählten Modi
ESC "[" Ps ";"...";" Ps "l"	Löscht die durch den Wert von Ps gewähl- ten Modi

Wert	Modus
0	(l) Drucker abwählen, (h) Drucker anwählen
1	Autozeilenvorschub nach Wagenrücklauf; h = Autozeilenvorschub; l kein Autozeilenvorschub
3	Löscht den aktuellen Inhalt des Puffers bei Empfang des externen Seitenanfangsbefehls

ESC "[" Ps ";"...";" Ps "m"	Setzt das durch den Ps-Wert gewählte Druckattribut
ESC "[" Ps ";"...";" Ps "n"	Löscht das durch den Ps-Wert gewählte Druckattribut; z. B. ist die Folge für Fettdruck mit Unterstreichung 1BH [1;4m.

Wert	Bedeutung
0	Alle Attribute
1	Überschreiben
2	Tiefstellung
3	Hochstellung
4	Unterstreichung
6	Überschreiben mit Schrägstrich
7	Doppel-Unterstreichung

H.2.3 Typenraddrucker-Rückkehrcodes

Tabelle H-3 führt Hexadezimal-Rückkehrcodes für Typenraddrucker auf.

Tabelle H-3. Typenraddrucker-Rückkehrcodes

Hex	Bedeutung
01	Abdeckung offen
02	Kein Papier
04	Kein Farbband
08	Drucker abgewählt
10	Top of Form (Seitenanfang) gedrückt
20	Papierzufuhr leer
40	Papierzufuhrstau
80	Drucker ausgeschaltet

H.3 MATRIXDRUCKER

In diesem Abschnitt sind Zeichensatz, Steuercodes, Umschaltcodefolgen und Rückkehrcodes für den Matrixdrucker enthaltem (siehe Wang Professional Computer Matrixdruckerhandbuch).

H.3.1 Matrixdrucker-Zeichensatz

Tabelle H-4 zeigt den Matrixdrucker-Zeichensatz mit entsprechenden Hexadezimalcodes. Tabelle H-5 enthält Zeichennamen und Dezimalwerte für Gebrauch bei Druckprogrammen.

Tabelle H-4. Matrixdruckerzeichen und Hexadezimalcodes

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0				0	@	P	`	p				Ğ	ğ	Þ	þ	£
1			!	1	A	Q	a	q	£		à	IJ	ij	Ð	à	ƒ
2			"	2	B	R	b	r	¨		á	İ	ı	Ŷ	ŷ	¥
3			#	3	C	S	c	s	´		ä	ş	î	Ş	ş	µ
4			$	4	D	T	d	t	`		ã	ì	i	Á	↓	½
5			%	5	E	U	e	u	§	┐	Ë	í	í	Ô	ô	¼
6			&	6	F	V	f	v	r	├	å	ï	ï	Ù	ù	Å
7			'	7	G	W	g	w		┤	±	Ш	Н	Ú	ú	Á
8			(	8	H	X	h	x		┬	æ	Ñ	ñ	Ü	ü	Ä
9			)	9	I	Y	i	y		└	ç	Ò	ô	Ø	←	Á
A			*	:	J	Z	j	z		┘	¡	ò	ò	°	Ij	Æ
B			+	;	K	[	k	{			¿	ó	ó	¢	→	Ç
C			,	<	L	\	l	|			ê	ö	ö	₁	°	ê
D			-	=	M	]	m	}		—	è	õ	õ	●	₁	è
E			.	>	N	↑	n	~		┴	é	Œ	œ	§	ß	É
F			/	?	O	_	o			┼	è	Ø	ø	¶	·	

Tabelle H-5. Matrixdruckerzeichen und Hexadezimal-Codes

DEZ	HEX	CHR	DEZ	HEX	CHR	DEZ	HEX	CHR	DEZ	HEX	CHR	DEZ	HEX	CHR	DEZ	HEX	CHR
32	20	Leerstelle	64	40	@ Zeichen	96	60	Anführung auf	160	AO	Leer	192	CO	g Hacek	224	EO	Thorn
33	21	!	65	41	A	97	61	a	161	A1	a Grave	193	C1	ij Ligatur	225	E1	Eth
34	22	"	66	42	B	98	62	b	162	A2	a Aigu	194	C2	i ohne Punkt	226	E2	y Aigu
35	23	#	67	43	C	99	63	c	163	A3	a Umlaut	195	C3	i Zirkumflex	227	E3	s Cedille
36	24	$	68	44	D	100	64	d	164	A4	a Tilde	196	C4	i Grave	228	E4	Abwärtspfeil
37	25	%	69	45	E	101	65	e	165	A5	E Trema	197	C5	i Aigu	229	E5	u Zirkumflex
38	26	&	70	46	F	102	66	f	166	A6	a Angstrom	198	C6	i Trema	230	E6	u Grave
39	27	'	71	47	G	103	67	g	167	A7	Plus/Minus	199	C7	ll Ligatur	231	E7	u Aigu
40	28	(	72	48	H	104	68	h	168	A8	ae Ligatur	200	C8	n Tilde	232	E8	u Umlaut
41	29	)	73	49	I	105	69	i	169	A9	c Cedille	201	C9	o Zirkumflex	233	E9	Linker Pfeil
42	2A	*	74	4A	J	106	6A	j	170	AA	Umgek. Ausruf	202	CA	o Grave	234	EA	Währung
43	2B	+	75	4B	K	107	6B	k	171	AB	Umgek Frage	203	CB	o Aigu	235	EB	Rechter Pfeil
44	2C	,	76	4C	L	108	6C	l	172	AC	e Zirkumflex	204	CC	o Umlaut	236	EC	o hochgest.
45	2D	-	77	4D	M	109	6D	m	173	AD	e Grave	205	CD	o Tilde	237	ED	a Zirkumflex
46	2E	.	78	4E	N	110	6E	n	174	AE	e Aigu	206	CE	oe Ligatur	238	EE	Beta
47	2F	Schrägstrich	79	4F	O	111	6F	o	175	AF	e Trema	207	CF	o Schrägstrich	239	EF	Punkt
48	30	Null	80	50	P	112	70	p	176	BO	G Hacek	208	DO	THORN	240	FO	Pfund Sterling
49	31	Eins	81	51	Q	113	71	q	177	B1	IJ Ligatur	209	D1	ETH	241	F1	Florin
50	32	Zwei	82	52	R	114	72	r	178	B2	I mit Punkt	210	D2	Y Aigu	242	F2	YEN
51	33	Drei	83	53	S	115	73	s	179	B3	I Zirkumflex	211	D3	S Cedille	243	F3	1/4
52	34	Vier	84	54	T	116	74	t	180	B4	I Grave	212	D4	A Aigu	244	F4	1/2
53	35	Fünf	85	55	U	117	75	u	181	B5	I Aigu	213	D5	U Zirkumflex	245	F5	3/4
54	36	Sechs	86	56	V	118	76	v	182	B6	I Trema	214	D6	U Grave	246	F6	A Zirkumflex
55	37	Sieben	87	57	W	119	77	w	183	B7	LL Ligatur	215	D7	U Aigu	247	F7	A Grave
56	38	Acht	88	58	X	120	78	x	184	B8	N Tilde	216	D8	U Umlaut	248	F8	A Umlaut
57	39	Neun	89	59	Y	121	79	y	185	B9	O Zirkumflex	217	D9	A Tilde	249	F9	A Angstrom
58	3A	Doppelpunkt	90	5A	Z	122	7A	z	186	BA	O Grave	218	DA	Grad	250	FA	AE Ligatur
59	3B	Semikolon	91	5B	L. eck. Klammer	123	7B	L. geschw. Kl.	187	BB	O Aigu	219	DB	Cent	251	FB	C Cedille
60	3C	Weniger als	92	5C	Gegenstrich	124	7C	Vertikalstrich	188	BC	O Umlaut	220	DC	a hochgest.	252	FC	E Zirkumflex
61	3D	Gleich	93	5D	R. eck. Klammer	125	7D	R. geschw. Kl.	189	BD	O Tilde	221	DD	Fetter Punkt	253	FD	E Grave
62	3E	Größer als	94	5E	Aufwärtspfeil	126	7E	Ungefähr	190	BE	OE Ligatur	222	DE	Paragraph	254	FE	E Aigu
63	3F	Fragezeichen	95	5F	Unterstreichung	127	7F	LÖSCH	191	BF	O Schrägstrich	223	DF	Neuer Absatz	255	FF	LÖSCH

DEZ=DEZIMAL
HEX=HEXADEZIMAL
CHR=ZEICHEN

H.3.2 Matrixdrucker-Steuercodes und Umschaltcodefolgen

Die Steuercodes für den Matrixdrucker sind dieselben wie die für den Typenraddrucker (Abschnitt H.2.2). Tabelle H-6 führt die Umschaltcodefolgen für den Matrixdrucker auf. Die Umschaltcodefolge (ESC) ist 01BH. Für alle folgenden ist der Parameter (P) eine ASCII-codierte Dezimalziffer von 0 bis 9 mit den Vorzeichen + oder - . Falls nichts anderes angegeben ist, stellt P Tausendstel eines Zolls dar. Der Standardwert für Ziffern ohne Vorzeichen ist positiv. Alle Leer- (020H) und Steuerzeichen werden in einer Umschaltcodefolge ignoriert.

Tabelle H-6. Matrixdrucker-Umschaltcodefolgen

Folge	Bedeutung
ESC "[" "z"	Stellt den Standardwert wieder her und leert den Puffer
ESC "[" "a"	Stellt den Standardwert wieder her und leert den Puffer
ESC "[" Ps "c"	Setzt linken Rand bei 10 Zeichen pro Zoll
ESC "[" Ps "d"	Setzt Zeilenvorschubabstand
ESC "[" Ps "e"	Setzt Formularlänge. Ps wird zu einem geraden Zollinkrement aufgerundet.
ESC "[" Ps "i"	Erteilt teilweisen Zeilenvorschub
ESC "[" Pi "o"	Zeichen pro Zoll wählen. Pi ist 16,5, 10 oder 5 Zeichen pro Zoll in Tausendsteln.
ESC "[" Ps ";" ... ";" Ps "h"	Unterdrückt Autozeilenvorschub nach Wagenrücklauf. Ps = 1.
ESC "[" Ps ";" ... "; "Ps "l"	Löscht Unterdrückung des Autozeilenvorschubs nach Wagenrücklauf. Ps = 1.
ESC "[" Ps ";" ... ";" Ps "m"	Setzt das durch Ps-Wert gewählte Druckattribut.
ESC "[" Ps ";" ... ";" Ps "n"	Löscht das durch Ps-Wert gewählte Druckattribut.

Tabelle H-6. Matrixdrucker-Umschaltcodefolgen (Forts.)

Wert	Bedeutung
0	Alle Attribute (Ausnahme: "m" setzt keine Hoch- oder Tiefstellung)
1	Überschreiben
2	Tiefstellung
3	Hochstellung
4	Unterstreichung
6	Autoschrägstrich
7	Doppel-Unterstreichung

H.2.3 Matrixdrucker-Rückkehrcodes

Tabelle H-7 führt Hexadezimal-Rückkehrcodes für den Matrixdrucker auf.

Tabelle H-7. Matrixdrucker-Rückkehrcodes

Hex	Bedeutung
02	Kein Papier
04	Fehler
08	Drucker abgewählt

ANHANG I
CODE KONVERTIERUNG VON IBM PC ZU WANG PC

I.1 EINFÜHRUNG

Programme in BASIC lassen sich mühelos vom IBM Personal Computer zum Wang Professional Computer konvertieren. Die nachfolgenden Abschnitte geben Richtlinien für diese Konvertierung.

I.2 TASTATUR, DRUCKER UND BILDSCHIRMPROTOKOLLE

Keine Syntaxmodifikationen sind nötig, um IBM BASIC Programme auf dem Wang PC ablaufen zu lassen. Der Microsoft BASIC Sprachübersetzer ist mit dem IBM PC Advanced BASIC Sprachübersetzer funktionell kompatibel. Es bestehen jedoch Unterschiede in Tastatur und Protokollanzeige. Wenn bestehende Programme konvertiert oder neue entwickelt werden, ist es vorteilhaft, Variablen zu benutzen, die Tasten und Anzeigewerte im Code darstellen.

Soll ein Programm z. B. einen vertikalen Balken auf beiden Maschinen anzeigen, kann im Programm eine Variable VERTBAR$ bestimmt werden. Auf dem IBM PC gleicht VERTBAR$ CHR$(179). Auf dem WANG PC wird es jedoch auf CHR$(134) gesetzt und zur Anzeige des vertikalen Balkens einfach PRINT VERTBAR$ codiert. Bei dieser Methode braucht nicht jeder Befehl in der für Textanzeige verantwortlichen Software geändert zu werden.

Diese Methode kann auch bei der Tastatur angewendet werden. Angenommen, ein Programm soll das Drücken der PF01-Taste (wenn als Soft-Taste gesperrt) prüfen. Eine PF01KEY$ Variable im Programm bestimmen. Auf dem IBM PC wird PF01KEY$ gleichwertig mit CHR$(059) gesetzt, auf dem WANG PC jedoch mit CHR(128). Dann kann die Variable, die den Tastenwert von dem INKEY$-Befehl erhält, mit der PF01KEY$ Variable anstatt eines hartcodierten Tastenanschlagwertes für eine bestimmte Maschine, verglichen werden.

Diese Methode kann einen Schritt weitergeführt werden, indem ein Standardvorsatz für jede Maschine erstellt wird, einschließlich aller Tasten- und Anzeigewerte, die bei einer bestimmten Maschine bestehen. Dieser Standardvorsatz könnte in jedes Programm eingefügt werden, das zur Verbesserung von Lesbarkeit und Übertragbarkeit der Software entwickelt wurde.

I.2.1 Tastaturwerte

Tastenwerte auf dem IBM PC sind mit denen auf dem WANG PC gleich. Dieser Vergleich beschränkt sich auf die bei beiden Machinen verfügbaren Tasten, da einige nur auf der WANG PC-, andere nur auf der IBM PC-Tastatur verfügbar sind (siehe Wang Professional Computer Einführungshandbuch).

Für bestimmte Tasten oder Tastenkombinationen, die nicht als Standard-WISCII-Code gespeichert werden können, sendet der BASIC INKEY$-Befehl einen erweiterten Code. Der Wert CHR$(031) zeigt an, daß ein erweiterter Code folgt. Wenn das erste erhaltene Zeichen CHR(031) ist, steht ein zweiter Code im Tastenanschlagpuffer aus. Ausführung eines weiteren INKEY$-Befehls sendet diesen zweiten Code unmittelbar. Folgendes ist eine Aufstellung der WISCII-Codes (Dezimal) für dieses zweite Zeichen und die damit verbundenen Tasten.

Zweiter Code	Bedeutung
067	UMSCH/ANNULLIER
128-143	PF01 bis PF16 (EINRÜCK bis GEHE NACH)
144-159	PF17 bis PF32 (UMSCH/EINRÜCK bis UMSCH/GEHE NACH)
192	Nordcursor
193	Ostcursor
194	Südcursor
195	Westcursor
196	Home
197	AUSF
198	EINRÜCK
199	LÖSCH
200	VORH
201	NÄCHS
203	LÖ REST
205	Rücktab
208	UMSCH/Nordcursor
209	UMSCH/Ostcursor
210	UMSCH/Südcursor
11	UMSCH/Westcursor
212	UMSCH/Home
13	UMSCH/AUSF
214	UMSCH/EINFÜGEN
215	UMSCH/LÖSCH
216	UMSCH/VORH
217	UMSCH/NÄCHS

Zweiter Code	Bedeutung
219	UMSCH/LÖSCH
220	UMSCH/Tab
221	UMSCH/Rücktab
224	ANNULLIER
225	HILFE
226	GL(Glossar)
227	DRUCK
241	UMSCH/HILFE
242	UMSCH/GL(Glossar)
243	UMSCH/DRUCK

I.3 TEXTDATEIEN VON IBM-PC AUF WANG-PC ÜBERTRAGEN

Zwischen Wang und IBM PC besteht Datenträger-Kompatibilität. Der WANG PC kann SSDD oder DSDD 5,25-Zoll-Disketten lesen, die auf dem IBM PC formatiert sind. Weil beide Systeme praktisch das gleiche Betriebssystem benutzen, sind die Diskettenformate vollkommen kompatibel. Deshalb kann der WANG PC alle durch DOS IBM PC erstellten Quellprogramme und Dateien mühelos lesen. BASIC Quellprogramme, die in das ASCII-Format zu übertragen sind, sollten gespeichert werden. Der WANG PC BASIC Compiler kann Quelldateien nicht in komprimiertem Format lesen.

Eine weitere Methode zur Übertragung von Quellprogrammen und Dateien zwischen den beiden Maschinen ist möglich, wenn beide Systeme mit Asynchron-Kommunikations-Hardware und -Software ausgestattet sind. Wang und IBM bieten beide Asynchron-Kommunikationsfähigkeiten. Auf dem IBM PC ist dies jedoch eine Option. Der Wang PC verfügt über eine Standard-RS-232-C Kommunikations-anschlußstelle.

Zum Übertragen von Textdateien auf den WANG PC IBM-asynchrone Kommunikations-Support-Software benutzen. Es folgen einige Beschränkungen, die durch das Protokoll für die IBM-asynchrone Kommunikations-Support-Software bedingt sind:

- Dateien müssen textlich sein und dürfen 254 Zeichen pro Zeile nicht überschreiten.
- BASIC Quellprogramme müssen im ASCII-Format sein.
- Jede Zeile in der Datei muß mit CR/LF enden.
- Jede Datei muß mit KTRL + Z abschließen.
- Ein Vollduplex-Protokoll wird angewendet [bei XON (KTRL + Q) und XOFF (KTRL + S)], um Kommunikationspuffer-Überlauf während Datenaufnahme zu verhindern.

Das nachstehende Beispielsprogramm in BASIC illustriert, wie Textdateien vom IBM PC auf den Wang PC durch asynchrone Kommunikation übertragen werden können. Das Programm setzt eine Zeile von 300-Bits-pro-Sekunde (bps), gerade Parität, sieben Datenbits und einen Eingabepuffer von 256 Bytes voraus. Erhaltene Zeichen werden auf dem Bildschirm angezeigt und können auf ihre Zulässigkeit geprüft werden.

Code-Konvertierung von IBM PC zu Wang PC

```
10  CLS:WIDTH 80: KEY OFF: DEFINT A-Z
20  LOCATE 10,10: INPUT "(T)transmit or (R)receive "; XMIT$
30  LOCATE 12,10: INPUT "ASCII Filename "; NAM$
40  IF XMIT$ = "t" OR XMIT$ = "T" GOTO 150
50  REM LINES 60-130 RECEIVE FILE FROM COM PORT
60  OPEN NAM$ FOR OUTPUT AS #1
70  OPEN "com1:300,E,7,1" AS #2
80  IF LOC(2)=0 THEN GOTO 80
90  WHILE NOT EOF(2)
100 LINE INPUT #2,A$
110 PRINT #1, A$
120 WEND
130 CLOSE: GOTO 220
140 REM LINES 150-210 TRANSMIT FILE TO COM PORT
150 OPEN NAM$ FOR INPUT AS #1
160 OPEN "com1:300,E,7,1" AS #2
170 WHILE NOT EOF(1)
180 LINE INPUT #1, A$
190 PRINT #2, A$
200 WEND
210 CLOSE
220 LOCATE 20,30: PRINT "Transfer Complete"
230 END
```

ANHANG J
INHALT DER WANG PC SYSTEMSOFTWARE-DISKETTE

In diesem Anhang wird der Inhalt der WANG PC Systemsoftware-Diskette aufgeführt, die Wang Laboratories, Inc. ab 24. Januar 1983 mit allen Systemen mitliefert. Spur 0, Sektor 1 der Diskette enthält den Startsektor. Zwei Kopien der FAT und ein Diskverzeichnis folgen dem Startsektor. Das Diskverzeichnis und die FATs werden in Abschnitt A.4 behandelt. Die Dateinamen auf der Systemsoftware-Diskette mit entsprechenden Versionsnummern (N/A = nicht zutreffend) und Beschreibung der Funktionen folgen:

Dateiname	Version	Beschreibung
BIOS.SYS	1	Basic E/A System
MSDOS.SYS	2	Diskbetriebssystem
CONFIG.SYS	N/A	Standard-Konfigurationsdatei (für MENU.COM)
CONFIG.BAK	N/A	Modifizierte Konfigurationsdatei (für COMMAND.COM)
DW20DRVR.COM	1	DW20 Druckerantrieb
CONFGEDT.COM	1	Konfigurationsdatei-Editor
COMMAND.COM	2	Befehlsprozessor
CHKDSK.COM	2.1	Disk-Utility prüfen
FORMAT.COM	2	Disk-Initialisierungs-Utility
FILCOM.COM	1	Dateivergleich Utility
PRINT.COM	1	Print Spooler Utilility
UTILITY.COM	1	System Utilities Programmlader
MENU.COM	1	Menü-Anzeige-Programm
MENUDRVR.COM	1	Lader für MENU.COM und Dateien, die über Menüs aufgerufen werden
MENUICMP.COM	1	Menü-Editor
WPCONV.COM	1	Textdatei zu WP Dokument Konvertierungs-Utility
CONVRSRC.WPS	1	WP Dokument Konvertierungsmitteilungen
KEYBOARD.WPS	1	Tastatur-Übersetzungstabelle
BASIC.EXE	1	GW Basic Interpretierer
UTILITY.MSG	1	System-Utilities-Mitteilungen
MENU.MSG	1	Datum/Zeit-Bildschirm und Menü-Mitteilungen
UTILITY.DAT	1	Utility-Menü-Daten für MENU.COM
MENU.DAT	1	Hauptmenü-Daten für MENU.COM
APPMENU.DAT	1	Anwendungsmenü-Daten für MENU.COM
PRGDVMEN.DAT	1	Programmentwicklungsmenü-Daten für MENU.COM
TCMENU.DAT	1	Kommunikationsmenü-Daten für MENU.COM
APPMENU.HLP	1	Anwendungsmenü-Hilfebildschirme

Dateiname	Version	Beschreibung
MENU.HLP	1	Hauptmenü-Hilfebildschirme
TCMENU.HLP	1	Kommunikationsmenü-Hilfebildschirme
PRGDVMEN.HLP	1	Programmentwicklungsmenü-Hilfebildschirme
UTILITY.HLP	1	System-Utilities-Menü-Hilfebildschirme
PRNDW20.BCF	1	DW20 Stapelbefehls-Datei für CONFGEDT.COM
DAISY06	1	Zeichensatztabelle für Typenraddrucker

ANHANG K
ABKÜRZUNGEN

Abkürzung	Bedeutung
AF	Zusätzliches Übertragkennzeichen
ANSI	American National Standards Institute
ASCII	American Nationaler Standard Code for Information Interchange
ASCIZ	ASCII-Folge mit Hexadezimal-Null als Folgenende-Anzeiger
AUX	Zusatzgerät
BIOS	Basic E/A-System
BPB	BIOS Parameterblock
CF	Übertragkennzeichen
CHR	Zeichen
CON	Konsolengerät
CRC	Zyklische Blockprüfung
CS	Codesegment (Register)
DEZ	Dezimal
DF	Richtungskennzeichen
DMA	Direkter Speicherzugriff
DOS	Diskbetriebssystem
DPB	Geräteparameterblock
DS	Datensegment (Register)
DTA	Disk-Übertragungsadresse
DWORT	Doppelwort
ES	Zusätzliches Segment (Register)
ESC	Umschaltcode
FAT	Dateizuordnungstabelle
FCB	Dateisteuerblock
H (suffix)	Hexadezimal
HEX	Hexadezimal
IF	Unterbrechungskennzeichen
Int	Software-Unterbrechung
IOCTL	Ein-/Ausgabesteuerung (DOS Funktionsaufruf)
IP	Instruktionsanzeiger (Register)
IRET	Rückkehr nach Unterbrechung (Assembler-Instruktion)
LED	Leuchtdiode
OF	Überlaufkennzeichen
PF	ParitätsKennzeichen
PRN	Druckergerät
PROM	Programmierbarer Festspeicher
PSH	Programmsegmentvorsatz
QID	Warteschlangen-ID-Nummer

Abkürzung	Bedeutung
SF	Vorzeichen-Kennzeichen
SP	Stapelanzeiger (Register)
SS	Stapelsegment (Register)
STI	Standard-Eingabegerät
STO	Standard-Ausgabegerät
TF	Auffangskennzeichen
WISCII	Wang International Standard Code for Information Interchange
ZF	Null-Kennzeichen