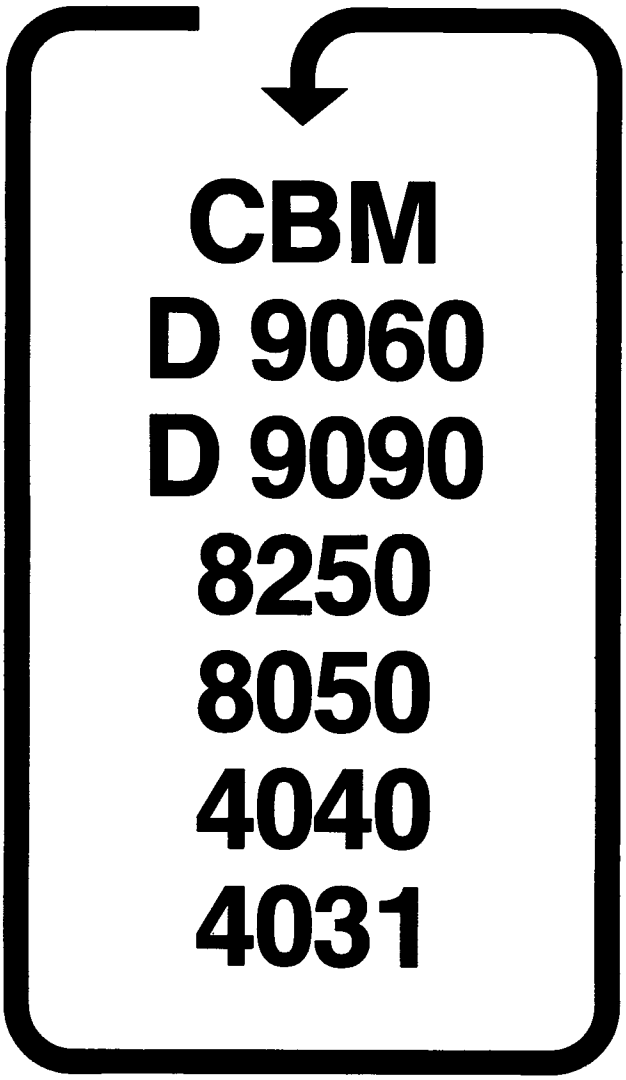

commodore - computer

Bedienungshandbuch

Inhaltsverzeichnis

Teil 1: Die CBM Disk-Einheiten

Einführung

Das Handbuch enthält Informationen über folgende Commodore Disk-Laufwerke:
- D9060/D9090 5 1/4 Zoll Winchester Disk (Festplatte)
- 8250 Doppellaufwerk, beidseitiger Diskettenzugriff
- 8050 Doppellaufwerk, einseitiger Diskettenzugriff
- 4040 Doppellaufwerk, einseitiger Diskettenzugriff
- 2031 Einzellaufwerk, einseitiger Diskettenzugriff

Die Befehle gelten größtenteils für alle Modelle. Die Ausnahmen sind in den entsprechenden Kapiteln beschrieben.

Durch das Commodore Disk-System wird die Leistungsfähigkeit des Commodore Computer Systems stark erweitert. Sämtliche Commodore Disk-Einheiten sind sogenannte "intelligente" Peripheriegeräte und benötigen daher für ihren Betrieb keinen zusätzlichen Speicherbereich im Rechner. Das bedeutet, daß im Rechner immer genausoviel RAM-Speicher zur Verfügung steht, egal, ob ein Disk-Laufwerk angeschlossen ist oder nicht.

Beschreibung D9060/D9090

Die beiden in diesem Handbuch beschriebenen Festplatten-Modelle sind Einzellaufwerk-Speichereinheiten. Die Hauptkomponenten sind Schreib-/Lese-Steuerung, Laufwerk-Motor-Steuerung, Laufwerk-Mechanik, zwei oder drei Platten mit beidseitiger Beschichtung, vier bzw. sechs Schreib-/Leseköpfe und der Spur-Positionierungs-Mechanismus. Die Laufwerke werden mit dem Rechner über eine IEEE-Schnittstelle verbunden. Der Steckeranschluß hierbei befindet sich auf der Rückseite des Laufwerkes. Außerdem befinden sich dort der Netzanschluß, der Ein-/Aus-Schalter und eine Sicherung.

Beschreibung 8250

Das 8250 Doppellaufwerk benutzt 100 Spuren pro Inch und Zwei-Kopf-Laufwerke mit einer formatierten Kapazität von 1.066.496 Bytes pro Laufwerk. Jede 8250 Diskette hat 154 Spuren, 77 auf jeder Seite, und ist weitgehend schreib- und lese-kompatibel mit dem Laufwerk des Modells 8050 (siehe Seite 59). Die 8250 benutzt "Micropolis" oder "Tandon" Laufwerke.

Beschreibung 8050

Die 8050 Doppellaufwerkseinheit benutzt 100 Spuren pro Inch und Ein-Kopf-Laufwerke mit einer formatierten Kapazität von 533.248 Bytes pro Laufwerk. Jede 8050 Diskette hat 77 Spuren und ist eingeschränkt schreib-/lese-kompatibel mit dem Laufwerk des Modells 8250 (siehe Seite 59). Die Kompatibilität ist auf eine Diskettenseite begrenzt. Die 8050 benutzt "Micropolis" oder "Tandon" Laufwerke.

Beschreibung 4040

Das 4040 Doppellaufwerk benutzt 48 Spuren pro Inch und Ein-Kopf-Laufwerke mit einer formatierten Kapazität von 174.848 Bytes pro Laufwerk. Jede 4040 Diskette hat 35 Spuren und ist **nicht** schreib-/lese-kompatibel mit den Laufwerken der Modelle 8050 oder 8250. Disketten von 4040-Laufwerken sind schreib-/lese-kompatibel mit den Laufwerken der Modelle 2031 und VC-1540/1541.

Beschreibung 2031

Das 2031 Einzellaufwerk benutzt 48 Spuren pro Inch und ein Ein-Kopf-Laufwerk mit einer formatierten Kapazität von 174.848 Bytes. Jede 2031-Diskette hat 35 Spuren und ist schreib-/lese-kompatibel mit den Laufwerken der Modelle 4040 und VC 1540/1541.

Anschluß der Disk-Einheit an den Rechner

Es gibt zwei Typen von Verbindungskabeln, um die Disk-Einheit mit dem System zu verbinden. Diese Kabel können vom Commodore-Händler geliefert werden.

S:P-Kabel: Art.Nr.: 602501
Dieses Kabel wird benutzt, um die Disk-Einheit direkt an den Rechner anzuschließen (außer SK-Typen).

P:P-Kabel: Art.Nr.: 602502
Dieses Kabel wird benutzt, um die Disk-Einheit an ein anderes Peripheriegerät (z.B. Drucker) oder an einen Rechner 8032 SK oder 8096 SK anzuschließen.

Die Disk-Einheit sollte das **erste** am Rechner angeschlossene Gerät sein!

Anleitung für den Anschluß einer Disk-Einheit an den Rechner:

1. Der Rechner und alle Peripheriegeräte müssen ausgeschaltet werden.

2. Die Disk-Einheit sollte so dicht wie möglich zum Rechner aufgestellt werden.

3. Ein S:P- oder P:P-Kabel stellt die Verbindung zwischen der IEEE-448-Schnittstelle am Rechner und der Disk-Einheit her.

4. Das Netzkabel der Disk-Einheit sollte bei ausgeschaltetem Netzschalter angeschlossen werden.

5. Nach Einschalten des Rechners sollte folgende Meldung erscheinen:
   *** Commodore Basic ***
   31743 Bytes Free (abhängig von der Speichergröße des Rechners)
   ready

6. Nach Einschalten der Disk-Einheit sollten alle Anzeigeleuchten zweimal aufleuchten. Die zweifarbige Netz/Fehler Leuchte sollte grün bleiben, was anzeigt, daß das Gerät eingeschaltet ist.

Wenn die Laufwerksleuchten anbleiben, dauernd flackern oder das Netz/Fehler Licht länger als fünf Sekunden rot bleibt, ist das Gerät auszuschalten. Nach einer kleinen Wartezeit ist der Vorgang zu wiederholen. Wenn der gleiche Effekt wieder auftritt, sind alle anderen Geräte vom IEEE-Bus zu entfernen. Damit ist sichergestellt, daß kein anderes Gerät die Disk-Einheit beeinflußt. Besteht das Problem noch immer, sollte mit dem Commodore-Händler Rücksprache genommen werden.

Nach dem Einschalten einer D9060 oder D9090 Festplatteneinheit muß eine Wartezeit von mindestens einer Minute eingehalten werden, bevor die erste Disk-Operation ausgeführt wird. Diese Zeit wird benötigt, um die Umdrehungsgeschwindigkeit zu stabilisieren. Sämtliche vorher abgesandten Befehle rufen die Fehlermeldung "DRIVE NOT READY" hervor, und das Laufwerk reagiert auf weitere Befehle nicht, bis der Initialisierungsbefehl ausgeführt ist.

Handhabung der Disketten

Die Disketten-Speichereinheiten haben entweder Micropolis, Tandon oder Shugart Laufwerke. Allgemein sollte darauf geachtet werden, daß die Disketten zentriert (nicht seitlich verschoben) in der Umhüllung liegen, bevor diese in das Laufwerk eingesteckt werden. Abhängig von den Laufwerk-Typen ist folgendes zu beachten:

1. **Micropolis - 8250:** Die in der 8250 benutzten Micropolis-Laufwerke haben Schalter zur Erkennung des Einsetzens der Disketten. Nach Schließen der Tür startet der Motor jedoch nicht. Um ein gutes Einsetzen der Disketten zu gewährleisten, sollte die Tür vor dem Einrasten ein- oder zweimal leicht heruntergedrückt werden.

2. **Micropolis - 8050:** Die in der 8050 benutzten Micropolis-Laufwerke starten den Laufwerk-Motor, sobald die Diskette arretiert ist. Die Tür sollte nach dem Einsetzen der Diskette fest, aber nicht gewaltsam heruntergedrückt werden.

3. **Tandon - 8050:** Es gibt zwei Versionen der Tandon-Laufwerke. Die beiden Versionen sind äußerlich gleich. Die spätere Version hat jedoch einen Schalter, der zum besseren Einsetzen der Diskette den Laufwerk-Motor startet. Bei der früheren Version wird empfohlen, die Tür vor dem Einrasten ein- oder zweimal leicht herunterzudrücken, was bei der späteren Version nicht notwendig ist.

4. **Shugart - 4040 / 4031:** Die Disketten werden durch einfaches Schließen der Laufwerk-Tür eingesetzt. Diese Laufwerke haben keine Erkennung für das Einsetzen der Diskette. Aus diesem Grund ist das Initialisierungskommando noch vor jedem anderen Disk-Kommando auszuführen.

Disketten sind empfindlich, stellen aber bei richtigem Umgang ein sehr zuverlässiges Speichermedium dar. Sie sollten stets schonend behandelt und niemals gewaltsam in das Laufwerk eigesetzt werden. Außerhalb des Laufwerks sollten sie immer in den Hüllen aufbewahrt werden. Disketten sollten nicht in der Nähe von magnetischen Feldern (z.B. Motoren oder Transformatoren) oder starker Hitze aufbewahrt werden.
Es sollten regelmäßig Sicherungskopien von den Datenbeständen erstellt werden, die an einem sicheren Ort aufbewahrt werden.

Sämtliche soft-sektorisierten Disketten arbeiten zuverlässig mit Commodore Disk-Einheiten. Für die 8050 und die 8250 Disk-Einheiten wird jedoch die Verwendung von "double-density"-Disketten (doppelte Aufzeichnungsdichte) empfohlen. Disketten mit Verstärkungsringen erhöhen die Zuverlässigkeit (besseres Zentrieren beim Einsetzen).

Commodore Disk-System Spezifikationen

Modell	D9090	D9060	8250	8050	4040	2031
Laufwerke pro Einheit	1	1	2	2	2	1
Köpfe pro Laufwerk	6	4	2	1	1	1
Formatierte Speicher-Kapazität pro Einheit	7,47 MB	4,98 MB	2,12 MB	1,05 MB	340 KB	170 KB
Sequent. Datei (max.)	7,41 MB	4,94 MB	1,05 MB	521 KB	168 KB	168 KB
Relative Datei (max.)	7,35 MB	4,90 MB	1,04 MB	183 KB	167 KB	167 KB
Speicher Disk-System	4 KB	4 KB	4 KB	4 KB	4 KB	2 KB
Disk-Formate:						
Zylinder (Spuren)	153	153	77	77	35	35
Sektoren pro Zylinder	128	192	-	-	-	-
Sektoren pro Spur	32	32	23-29	23-29	17-21	17-21
Bytes pro Sektor	256	256	256	256	256	256
Freie Blöcke (Sektoren)	29162	19442	8266	4104	1328	664
Übertragungsgeschwindigkeiten: (Bytes/Sekunde)						
Intern zur Einheit	5 MB	5 MB	40 KB	40 KB	40 KB	40 KB
IEEE-488 Bus	1,2 KB	1,2 KB	1,2 KB	1,2 KB	1,2 KB	1,2 KB
Zugriffszeit: (Millisekunden)						
Spur-zu-Spur	3	3	6	*	30	30
Durchschnitt Spur	153	153	125	**	360	360
Kopfpositionierung	15	15	-	-	-	-
Durchschnittl. Zugiff	8,34	8,34	100	100	100	100
Umdrehungen pro Min.	3600	3600	300	300	300	300
* Spur-zu-Spur:	Micropolis 8050 = 30ms			Tandon 8050 = 5ms		
** Durchschnitt Spur:	Micropolis 8050 = 750ms			Tandon 8050 = 125ms		
Abmessungen:						
Höhe (cm)	14,6	14,6	17,8	17,8	17,8	14,0
Breite (cm)	21,0	21,0	38,1	38,1	38,1	20,3
Tiefe (cm)	38,7	38,7	34,9	34,9	34,9	36,2
Gewicht (kg)	9,45	9,45	12,6	12,6	12,6	9,0
Elektrische Daten:						
Leistungsaufnahme	200 W	200 W	60 W	50 W	50 W	40 W
Spannung (alle Modelle)	220 V Wechselspannung, 50 Hz					

Teil 2: Anwendung der CBM Disk-Einheiten

Einleitung

Commodore Disk-Einheiten erweitern die Rechnerleistung des Systems mit zusätzlichem Speicher. Man hat die Möglichkeit, Dateien abzuspeichern und zu verwalten. Die Disk-Einheiten werden direkt angesteuert mit:

BASIC Befehlen, über Tastatur eingegeben
BASIC Anweisungen, innerhalb von Programmen
Speziellen Disk-Befehlen.

In diesem Handbuch wird die Anwendung, Funktion und das Format der Befehle, die dem Anwender Disk-bezogene Arbeiten erlauben, erklärt. Für BASIC 4.0 Anwender sind jene BASIC-Befehle aufgeführt, die zu dem jeweiligen Disk-Befehl gehören. Eine Gegenüberstellung der BASIC 3.0 und BASIC 4.0-Befehle befindet sich auf den Seiten 79 und 80.

Dateien

Eine Datei ist eine organisierte Zusammenstellung von Informationen, gespeichert auf einem Datenträger, wie z.B. Magnetband oder Diskette. Für einen Computer ist es notwendig, daß man Dateien mit Namen für deren Identifikation versieht.
Eine Datei kann Programm-Instruktionen (Programmdatei) oder Daten (z.B. Namenslisten oder Materialbestand) beinhalten, auf die ein Programm zugreifen kann, um die Information zu erhalten.

Commodore Disk-Operations-System (DOS)
(Disk-Betriebssystem)

Das DOS ist verantwortlich für die Handhabung des Informationsaustauschs zwischen der Disk-Steuerung und dem Rechner. Das DOS führt viele Funktionen aus, die dem Benutzer zwar unsichtbar bleiben, für die Disk-Operationen aber unabdingbar sind. Zum Beispiel überwacht das DOS die Ein-/Ausgaben (E/A) der Disk in der Weise, daß Kanäle entsprechend zugewiesen werden und keine längeren Wartezeiten entstehen. Das DOS benutzt die Kanalstruktur zum Untersuchen des Inhaltsverzeichnisses, sowie zum Kopieren und Löschen von Dateien.

Die Block-Verfügbarkeits-Karte (engl.: Block Availability Map = BAM)

Die BAM ist eine Auflistung der verfügbaren und zugewiesenen Blöcke (=Sektoren) auf der Diskette. Beim Formatieren einer Diskette wird die BAM erzeugt, welche dann beim Initialisieren in den DOS-Speicher geladen wird. Die BAM steht an verschiedenen Stellen auf der Diskette, abhängig vom Modell der Disk-Einheit.
Beim Abspeichern von Daten auf der Diskette wird das DOS entsprechend der Eintragungen in der BAM entscheiden, ob Platz auf der Diskette verfügbar ist. Bei sequentiellen oder Programm-Dateien überprüft das DOS den Leerbereich, bevor jeder Block der Datei geschrieben wird. Wenn ein freier Block gefunden wurde, wird die BAM aktualisiert. Sollten keine freien Blöcke verfügbar sein, wird eine Fehlermeldung erzeugt.
Entsprechend der geänderten Eintragungen der BAM im DOS-Speicher wird die BAM auf der Diskette periodisch aktualisiert. Dies geschieht, wenn bei einem Programm ein "save"- bzw. "dsave"-Befehl oder bei Dateien ein "close"- bzw. "dclose"-Befehl ausgeführt wird. Die BAM auf der Diskette kann aus mehreren Blöcken bestehen. Im DOS-Speicher befindet sich nur ein Block der BAM.

Kommunikation mit dem DOS

Die Ein-/Ausgabeprogrammierung kann umfangreich werden, wenn Daten zu oder von Kassetten-Einheiten, Druckern, Disk-Einheiten und anderen Peripheriegeräten übertragen werden. Für diese umfangreicheren Operationen muß ein "Kanal" zwischen dem Programm und dem ausgewählten Gerät unter Benutzung der CBM BASIC "open"- bzw. "dopen"-Anweisung eröffnet werden. Nach Ausführung der gewünschten Operationen muß der Kanal mit der "close"- bzw. "dclose"-Anweisung wieder geschlossen werden.

Jedes der am Rechner angeschlossenen Geräte hat seine eigene physikalische Gerätenummer (8 bis 15 für Disk-Einheiten), auf die es antwortet, wenn es vom Rechner angesprochen wird. Die Gerätenummer wird als Parameter beim Eröffnen eines Kanals angegeben, um das entsprechende Gerät zu identifizieren.
CBM Disk-Geräte sind fabrikmäßig mit der Gerätenummer 8 ausgerüstet. Diese kann jedoch über DOS-Befehle geändert werden. Das DOS erkennt die angegebene Gerätenummer durch die Disk-Behandlungs-Befehle und einigen Dateiverwaltungsbefehlen. Ist keine Gerätenummer angegeben, wird das DOS die Gerätenummer 8 annehmen.

Der Programmierer vergibt die Kanalnummern 0-255 als logische Dateinummern. Diese bringen Anweisungen wie "open" bzw. "dopen", "close" bzw. "dclose", "input", "get" und "print" untereinander in Beziehung und stellen die logische Verknüpfung zwischen Rechner und Peripheriegerät dar.

Zusätzlich zu den logischen Datei-Nummern und den Gerätenummern reagieren Commodore Disk-Einheiten auf verschiedene "sekundäre Adressen". Diese kann man sich als "Befehle" vom Rechner vorstellen, die der Disk-Einheit mitteilen, welche Operation auszuführen ist.
Sekundär-Adresse 0 wird benutzt (mit "dload"), um eine Programm-Datei in den Arbeitsspeicher des Rechners zu laden, Sekundär-Adresse 1 (mit "dsave"), um Programme vom Speicher in eine Disketten Programm-Datei zu schreiben. Die Sekundär-Adressen 2 - 14 werden benutzt, um auf Dateien zuzugreifen. Sekundär-Adresse 15 ist eine spezielle "Befehl/Status"-Adresse und wird benutzt, um viele der in diesem Handbuch beschriebenen Disk-Operationen auszuführen und um Status-Informationen über Disk-Operationen zu erhalten. Der mit Sekundär-Adresse 15 eröffnete Kanal wird auch als Befehls- oder Fehlerkanal bezeichnet.

Übersicht der DOS-Versionen

DOS 2.1 arbeitet mit der 4040 Doppellaufwerk-Einheit. Die 2040 Disk-Einheiten können durch Austausch der ROM-Chips auf DOS 2.1 umgerüstet werden. Die Zuverlässigkeit des Aufzeichnungsformates wurde im DOS 2.1 gegenüber DOS 1.0 dadurch verbessert, daß von Spur 18 bis 24 jeweils ein Block (Sektor) weggelassen wurde. Als Ergebnis faßt das Inhaltsverzeichnis 144 Datei-Einträge und 664 Blöcke für Anwenderdaten.

Die Struktur der relativen Datei wurde im DOS 2.1 ergänzt, um die Voraussetzung für freien Dateizugriff zu schaffen. Die Block Schreib-/Lese-Befehle des DOS 1.0 werden zwar akzeptiert, aber es sollten die entsprechenden "U1" und "U2" Dienstbefehle benutzt werden, um die Kompatibilität zu künftigen CBM Disk-Produkten sicherzustellen.

Allgemein sollte Software, die nicht von physikalischen Geräteeigenschaften abhängt, aufwärts kompatibel sein. Programme, die Block Schreib-/Lese-Befehle benutzen, sind sehr empfindlich gegenüber einem DOS-Austausch.

DOS 2.5 wird in allen 8050 Doppellaufwerk-Einheiten benutzt. Das DOS 2.5 beinhaltet alle Merkmale des DOS 2.1 und ist an zusätzliche Leistungen angepaßt. DOS 2.5 enthält Erweiterungen, wie die Überwachung des Disketten-Einsetzens und bessere Fehlerbehandlungstechniken. Das Inhaltsverzeichnis erlaubt 224 Datei-Einträge und enthält 2052 Blöcke für Anwenderdaten.

DOS 2.6 wird in den 4031 Einzellaufwerk-Einheiten benutzt. DOS 2.6 ist voll kompatibel mit DOS 2.1 (benutzt in der 4040) mit der Einschränkung, daß Doppellaufwerk-Befehle nicht arbeiten.

DOS 2.7 wird in den doppelseitigen 8250 Doppellaufwerk-Einheiten benutzt. DOS 2.5 Disk-Befehle und die 8050 Disk-Einheit sind aufwärts kompatibel zu DOS 2.7 und der 8250. Mit gewissen Einschränkungen sind Disketten, die auf einer der beiden Disk-Einheiten erstellt wurden, schreib-/lesekompatibel. Ein wichtiges Merkmal der 8250 ist die vergrößerte Kapazität für relative Dateien, die erlaubt, daß diese die gesamte 8250-Diskette belegen können. Es steht eine Kapazität von 1 Million Bytes zur Verfügung.

DOS 3.0 wird in den D9060 und D9090 Festplatten-Einheiten benutzt. Besonderheiten von DOS 3.0 sind das erweiterbare Inhaltsverzeichnis, das eine uneingeschränkte Anzahl von Datei-Einträgen erlaubt, außerdem die interne Verwaltung von Ersatzsektoren für defekte Sektoren und die selbständige Platzierung der BAM.

Allgemeine Operationen des DOS

Der DOS Datei-Schnittstellen-Controller ist verantwortlich für den Datenverkehr zwischen dem IEEE-Bus und dem Disk-Controller.

Das Datei-System ist durch Kanäle organisiert, die durch die BASIC "open"- bzw. "dopen"-Anweisung eröffnet werden. Beim Ausführen dieser Anweisungen weist das DOS jedem Kanal einen Arbeitsbereich und ein bis drei Disk-E/A-Puffer zu. Ist kein Arbeitsbereich oder kein Puffer frei, wird ein NO CHANNEL-Fehler erzeugt. Das DOS benutzt auch die Kanal-Struktur, um das Inhaltsverzeichnis zu untersuchen oder um Dateien zu löschen oder zu kopieren.

Der Speicher zwischen dem Disk,Controller und dem Datei-Schnittstellen-Controller wird für 256-Byte Pufferbereiche benutzt. Drei der 16 Puffer werden vom DOS für die BAM, Befehlskanal E/A und die "Job Queue" des Disk-Controllers benutzt.

Die "Job Queue" ist die ständige Verbindung zwischen den beiden Controllern. "Jobs" werden auf der Dateiseite dadurch eingeleitet, daß der Disk-Controller mit den Sektor-Informationen und Informationen über den Typ der Operation versorgt wird. Der Disk-Controller sucht nach dem optimalen "Job" und versucht die Ausführung. Anstelle des "Job"-Befehls wird der Fehlerzustand zurückgeschickt. Sofern der "Job" nicht erfolgreich abläuft, führt die Dateiseite eine bestimmte Anzahl von Wiederholungen durch, bevor eine Fehlermeldung erzeugt wird.

Die mit der "open"-Anweisung angegebene Sekundär-Adresse wird vom DOS als Kanalnummer benutzt. Die Nummer, die der Anwender einem Kanal zuweist, ist nur eine Referenznummer, die benutzt wird, um auf die Arbeitsbereiche zuzugreifen und hat keinen Bezug zur Organisation der DOS-Kanäle.

Die "dload"- und "dsave"-Anweisungen übertragen die Sekundär-Adressen 0 beziehungsweise 1. Das DOS interpretiert diese Sekundär-Adressen automatisch als "dload" und "dsave"-Funktionen. Wenn diese Funktionen nicht gewünscht werden, dürfen beim Eröffnen von Dateien die Sekundär-Adressen 0 und 1 nicht benutzt werden. Die verbleibenden Nummern 2 bis 14 können als Sekundär-Adressen benutzt werden.
Mit dem Eröffnen einer Datei werden im DOS für sequentielle Dateien 2, und für relative Dateien 3 der insgesamt 10 für den Anwender verfügbaren Puffer belegt.

Dateinamen

Commodore Disk-Einheiten können auch mit Abkürzungen von Dateinamen arbeiten. Hierbei werden die Zeichen "?" und "*" zur Ergänzung der Namensabkürzungen bzw. Namensteile verwendet. Das Zeichen "*" wird am Ende einer Zeichenfolge angehängt, um zu definieren, daß der Rest des Namens bedeutungslos ist.
Zum Beispiel könnte "FIL*" verweisen auf folgende Namen:
FIL,
FILE1,
FILEDATA,
FILLER,
oder andere Namen die mit den Buchstaben "FIL" beginnen.

Das Zeichen "?" kann irgendwo innerhalb einer Zeichenfolge hingesetzt werden, um zu definieren, daß das an dieser Position stehende Zeichen ohne Bedeutung ist.
Zum Beispiel könnte "?????.SRC" auf folgende Namen verweisen:
TSTER.SRC,
DIAGN.SRC,
PROGR.SRC,
aber **nicht** auf "SRC.FILES".

Sowohl die Zeichen als auch die Positionen der Zeichen sind von Bedeutung.

Die Zeichen "?" und "*" können auch sinnvoll kombiniert werden. Das Zeichen "*" sollte jedoch immer als letztes in einer Zeichenfolge erscheinen, egal ob "?" benutzt wurde oder nicht.
Zum Beispiel ist "J*?????" nicht sinnvoll, weil die "?" in einem Bereich stehen, der (bedingt durch "*") bedeutungslos ist.

Die Namensangabe "P???FIL*" kann z.B. auf folgende Namen verweisen:
PET FILE,
PRG-FILE-32,
POKEFILES$$,
oder andere Namen, die mit "P" beginnen und in den Positionen 5-7 die Zeichenfolge "FIL" haben.

Mit dem Befehl **dload "*"** wird die erste Programm-Datei im Disk-Inhaltsverzeichnis von Laufwerk 0 geladen.

Mit dem Befehl **dopen** mit einer Abkürzung als Namensangabe wird die erste erreichte Datei, die der Definition der Abkürzung entspricht, eröffnet. Der Befehl **dopen** darf nicht mit einer Namensabkürzung versehen werden, wenn eine neue Datei eröffnet werden soll.

Beim **scratch**-Befehl sollten Abkürzungen vorsichtig angewendet werden, da möglicherweise (unbeabsichtigt) auf mehrere Dateien zugegriffen wird.

Die Befehle **rename**, **dsave** oder **copy** sollten nicht mit abgekürzten Namen benutzt werden.

Disk-Bedienungsbefehle

Die folgenden Disk-Befehle ermöglichen die Dateibehandlung und Disk-Bedienung.

Die Befehle auf **Disk-Ebene** sorgen für
a) das Formatieren von Disketten, um diese gebrauchsbereit zu machen,
b) das Anzeigen des Verzeichnisses der auf der Diskette enthaltenen Dateinamen (Directory),
c) die Initialisierung eines Gerätes, um sicherzustellen, daß das DOS erkennt, welche Blöcke auf der Diskette belegt sind,
d) das Wiederherstellen einer gültigen BAM im Falle eines Software-Fehlers.

Die Befehle auf **Datei-Ebene** sorgen für
a) das Kopieren von Dateien von einem Laufwerk zum anderen,
b) das Anhängen einer Datei an eine andere,
c) das Wechseln der Dateinamen,
d) das Löschen nicht mehr erwünschter Dateinamen aus dem Inhaltsverzeichnis

	Funktion	**Befehl**
Disk-Ebene:	Formatieren einer Diskette	header
	Lesen von BAM in DOS-Speicher	initialize
	Lesen von Disk-Inhaltsverzeichnis	directory
	Wiederherstellung von BAM	collect
Datei-Ebene:	Kopieren von Dateien	copy
	Kopieren mit Anhängen	concat
	Umbenennen einer Datei	rename
	Löschen einer Datei	scratch

Die Laufwerk-Nummern in den folgenden Beispielen der DOS-Befehle berücksichtigen Disk-Einheiten mit zwei Laufwerken. Bei Benutzung der 4031 Einzel-Disk oder der D9060/9090 Festplatten-Laufwerke müssen alle Verweise auf Laufwerknummern "0" sein. Ein Verweis auf Laufwerk 1 hat einen Fehler-Zustand zur Folge.

Variable in Befehlsparametern

Jeder Disk-Befehl ist verbunden mit einem oder mehreren wahlweisen Parametern, die benutzt werden können, um Dateinamen, Laufwerknummern, etc. anzugeben. Wenn benötigt, können die Befehlsparameter in zwei Formen erscheinen. Parameter können direkt angegeben werden, wie:

```
dopen#1,"Bestandsdatei",d1
```

oder mit eingeklammerten BASIC Variable-Namen, wie:

```
a$ = "Bestandsdatei" : dn = 1
dopen#1,(a$),d(dn)
```

Die beiden "dopen"-Befehle bewirken dasselbe.

Bei der Eingabe von Disk-Befehlen über die Tastatur im Direkt-Modus müssen die Parameter direkt angegeben werden. In Programmen können Parameter direkt oder als Variable angegeben werden oder beide Formen können im gleichen Befehl benutzt werden.

Befehlsabkürzungen

Gleichgültig, ob im Direkt-Modus oder in einem Programm, können DOS-Befehle entweder in ihrer vollen Schreibweise oder in abgekürzter Form auftreten.
Für jeden Befehl müssen genügend Zeichen eingegeben werden, um ihn von den anderen DOS-Befehlen zu unterscheiden. Alle, außer dem letzten Zeichen der Abkürzung, werden als Kleinbuchstaben eingegeben, das letzte Zeichen muß ein Großbuchstabe sein.
Zum Beispiel sind "catalog" und "cA", sowie "print#1" und "pR1" für das DOS identisch. Die Abkürzung von Befehlen bringt keine Ersparnis im Speicher, ist aber für den Benutzer sehr hilfreich. Beim Listen eines Programms, welches Befehlsabkürzungen enthält, erscheinen die Befehle in ihrer vollen Länge.

Teil 3: BASIC-Befehle für die Disk-Bedienung

Erklärung der Zeichen bei der Angabe der Befehlsformate:

< **>** ... Entsprechend der Angaben **muß** hier ein Wert eingesetzt werden.
[**]** ... Diese Angabe kann bei Bedarf eingesetzt werden. Bei Weglassen wird vom Interpreter ein Wert zugewiesen (sog. Default-Wert).

append#

Format: append# <log. Datei-Nummer>,"<Name>" [,d<x>] [onu<y>]

Kurzform: aP#

Zweck: Verlängern einer sequentiellen Datei

Beschreibung: Bei sequentiellen Dateien wirkt "append#" wie "dopen". Der Datensatz-Zeiger wird hierbei jedoch nicht an den Anfang, sondern an das Ende der Datei gestellt.
Ohne Angabe gilt Laufwerk-Nummer = 0 und Einheit-Nummer = 8.

Beispiel: append#1,"MASTER"

Die Datei "MASTER" auf Laufwerk 0, Einheit 8 mit der logischen Datei-Nummer 1 wird zum Verlängern wiedereröffnet.

backup

Format: backup d<x1> to d<x2> [onu<y>]

Kurzform: bA

Zweck: Kopieren des gesamten Inhalts einer CBM-Diskette auf eine zweite Diskette.

Beschreibung: Der "backup"-Befehl ist nur bei Disk-Einheiten mit zwei Laufwerken anwendbar (nicht bei Festplatten-Einheiten!).
Die Diskette, auf die geschrieben wird, muß nicht formatiert sein, da alle auf der Ausgangsdiskette vorhandenen Informationen (incl. Formatierung) übertragen werden.
Ohne Angabe gilt Einheit-Nummer = 8.

Beispiel: **backup d0 to d1**

Inhalt der Diskette von Laufwerk 0 wird auf die Diskette in Laufwerk 1 kopiert.

catalog

Format: catalog [d<x>] [onu<y>]

Kurzform: cA

Zweck: Lesen und Anzeigen des Disk-Inhaltsverzeichnisses

Beschreibung: Mit diesem Befehl bekommt man auf dem Bildschirm eine Auflistung der auf der Diskette gespeicherten Dateinamen. Der Speicherinhalt des Rechners wird nicht zerstört. Die Bildschirmanzeige beinhaltet folgendes:

- Disk-Name, Disk-ID und DOS-Version
- Datei-Namen und Datei-Typ
- Datei-Größe (Anzahl der belegten Blöcke)
- Anzahl der verfügbaren (freien) Blöcke auf der Diskette

Wenn man die Laufwerk-Nummer (d1) nicht angibt, wird das Inhaltsverzeichnis beider Laufwerke angezeigt.
Ist die Disk-Einheit-Nummer nicht "8", so muß diese auch angegeben werden (z.B.: onu12).

Beispiel: catalog d1

Das Inhaltsverzeichnis von Laufwerk 1 wird angezeigt.

Bemerkung: Zum Auflisten des Inhaltsverzeichnisses auf einem Commodore Drucker kann die folgende Befehlsfolge benutzt werden:

```
open1,4     Log. Kanal zu Gerät 4 (Drucker) wird eröffnet
cmd1        Schaltet Bildschirmausgabe zu Gerät 4
catalog     Druckt das Inhaltsverzeichnis
print#1     Bringt die Ausgabe wieder zum Bildschirm
close1      Schließt den log. Kanal zum Drucker
```

close

Format: close ‹log. Datei-Nummer›

Kurzform: cl0

Zweck: Schließen eines logischen E/A-Kanals

Beschreibung: Eine mit "open" bzw. "dopen" eröffnete logische Datei wird abgeschlossen. Die logische Datei-Nummer, auch als logische E/A-Kanal-Nummer bezeichnet, ist nun wieder verfügbar (0-255).
Der "close"-Befehl bewirkt bei einer sequentiellen Datei, daß die noch im Ausgabepuffer stehenden Daten abgespeichert werden.

Beispiel: close7

Die Datei mit der logischen Dateinummer 7 wird abgeschlossen.

collect

Format: collect [d<x>] [onu<y>]

Kurzform: coL

Zweck: Gibt den Bereich von nicht ordnungsgemäß abgeschlossenen Dateien frei und löscht deren Eintragungen im Inhaltsverzeichnis.
Sämtliche Datenblöcke, die keiner Datei zugewiesen sind, werden freigegeben.
Die BAM wird neu aufgebaut und auf Diskette geschrieben.

Beschreibung: Eine Datei gilt dann als nicht ordnungsgemäß abgeschlossen, wenn diese mit "open" bzw. "dopen" eröffnet, aber nicht mit "close" bzw. "dclose" abgeschlossen wurde.
Der "collect"-Befehl läuft durch jeden Datenblock aller Dateien auf der Diskette. Nach erfolgreichem Durchlauf wird die neue BAM im Disk-Speicher erzeugt und auf Diskette geschrieben.
Sämtliche Blöcke, die keinem Dateinamen zuzuordnen sind, werden wieder zur Benutzung freigegeben. Dieser Effekt tritt bei allen Blöcken auf, die im Direktzugriff ("block-allocate") als belegt gekennzeichnet wurden.
Wenn man die Laufwerknummer (d0) nicht angibt, wird der Befehl bei beiden Laufwerken (sofern vorhanden) ausgeführt.
Ist die Disk-Einheit-Nummer nicht "8", so muß diese auch angegeben werden (z.B.: onu10).

Beispiel: collect d0

Der Befehl wird in Laufwerk 0, Disk-Einheit 8 ausgeführt.

concat

Format: concat [d<x1>] ,"<Dateiname 1>"
to [d<x2>] ,"<Dateiname 2>" [onu<y>]

Kurzform: conC

Zweck: Zusammenfügen von zwei sequentiellen Dateien.

Beschreibung: Der "concat"-Befehl kopiert eine sequentielle Datei hinter das Ende einer anderen sequentiellen Datei. Die Ursprungsdatei (Dateiname 1) bleibt unverändert, die Bestimmungsdatei (Dateiname 2) enthält danach die Daten beider Dateien.
Ohne Angabe der Ursprungs-Laufwerk-Nummer (x1) wird die Datei auf beiden Laufwerken gesucht.
Ohne Angabe gilt Bestimmungslaufwerk (x2) = 0,
Einheit-Nummer = 8.

Beispiel: concat d0,"Namen" to d1,"Freunde"

Die Datei "Namen" in Laufwerk 0 wird an die Datei "Freunde" in Laufwerk 1 angehängt. Die Datei "Namen" wird nicht verändert, die Datei "Freunde" erhält die Daten beider Dateien.

copy

Format: copy [d<x1>][,"<Dateiname 1>"]
to [d<x2>][,"<Dateiname 2>"][onu<y>]

Kurzform: coP

Zweck: Kopieren von Dateien innerhalb eines Laufwerks oder zu einem anderen Laufwerk.

Beschreibung: Die Ursprungsdatei wird unter dem als Zieldatei angegebenen Namen entweder zum gleichen oder zum anderen Laufwerk (entsprechend der angegebenen Laufwerk-Nummer) kopiert. Die Ursprungsdatei bleibt unverändert.
Ohne Angabe eines Dateinamens wird der gesamte Disketteninhalt kopiert.
Ohne Angabe der Ursprungslaufwerk-Nummer wird auf beiden Laufwerken nach der Datei gesucht.
Ohne Angabe gilt Bestimmungslaufwerk (x2) = 0,
Einheit-Nummer = 8.

Beispiel: copy d0,"Bestand" to d1,"Lagerbestand" onu9

In der Disk-Einheit 9 wird die Datei "Bestand" in Laufwerk 0 zum Laufwerk 1 unter dem Namen "Lagerbestand" kopiert.

Beispiel: copy d0,"Bestand" to d1,"*"

In der Disk-Einheit 8 wird die Datei "Bestand" in Laufwerk 0 zum Laufwerk 1 unter dem gleichen Namen kopiert.

Beispiel: copy d0 to d1

In der Disk-Einheit 8 werden alle Dateien von Laufwerk 0 nach Laufwerk 1 kopiert.

dclose

Format: dclose [#<log. Datei-Nummer>] [onu<y>]

Kurzform: dcL

Zweck: Abschließen von Disk-Dateien

Beschreibung: Der "dclose"-Befehl schließt Dateien ab, die mit "open" bzw. "dopen" eröffnet wurden, aktualisiert die BAM und die Eintragung der belegten Blöcke im Inhaltsverzeichnis.
Die BAM wird neu aktualisiert auf Diskette geschrieben.
Ohne Angabe der logischen Datei-Nummer werden alle gegenwärtig eröffneten Disk-Dateien abgeschlossen.
Ohne Angabe ist die Einheit-Nummer = 8.

Beispiel: dclose#25

Die logische Datei-Nummer 25 wird abgeschlossen.

Bemerkung: Nach einer Bearbeitung sollte grundsätzlich jede Datei abgeschlossen werden.
Maximal zehn eröffnete Dateien im Rechner und fünf pro Disk-Einheit sind erlaubt. Es ist daher ratsam, jede Datei so schnell wie möglich abzuschließen, um stets die maximale Anzahl eröffneter Dateien zur Verfügung zu haben.

directory

Format: directory [d<x>] [onu<y>]

Kurzform: diR

Zweck: Anzeigen des Inhaltsverzeichnisses einer Diskette.

Beschreibung: Siehe Beschreibung des "catalog"-Befehls.
Die Befehle "catalog" und "directory" haben die gleiche Funktion.

Beispiel: directory d1

Das Inhaltsverzeichnis von Laufwerk 1, Gerät 8 wird angezeigt.

dload

Format: dload [d<x>] [onu<y>] ,"<Dateiname>"

Kurzform: dL

Zweck: Laden einer Programm-Datei in den Arbeitsspeicher des Rechners.

Beschreibung: Der "dload"-Befehl überträgt Programm-Dateien von der Disk zum Speicher des Rechners und schließt alle eröffneten Dateien ab. Aus diesem Grund muß der Benutzer einen neuen "dopen"-Befehl geben, um die Kommunikation mit den Disk Befehls- und Fehlerkanälen fortzusetzen.
Ohne Angabe gilt Laufwerk = 0 und Disk-Einheit = 8.
Der Dateiname darf nach dem letzten signifikanten (zur Unterscheidung notwendigen) Zeichen mit einem "*" abgekürzt werden.

Beispiel: dload d0,"Kunden"

Das Programm "Kunden" wird von Laufwerk 0, Disk-Einheit 8 in den Speicher des Rechners geladen.

Bemerkung: Bei Rechnern mit BASIC 4.0 bewirkt das Betätigen der Tasten "SHIFT" und "RUN/STOP" (gleichzeitig) das Laden und Starten des ersten Programms im Inhaltsverzeichnis von Laufwerk 0, Disk-Einheit 8.
Bei deutscher Tastatur sind die Tasten "CTRL", "SHIFT" und "RUN/STOP" zu betätigen.

dopen

Format: 1. Für sequentielle Dateien:

dopen# <log. Datei-Nummer> ,"<Datei-Name>" [,w] (oder: ,r)
[,d<x>] [onu<y>]

2. Für relative Dateien:

dopen# <log. Datei-Nummer> ,"<Datei-Name>" ,l Satzlänge
[,d<x>] [onu<y>]

Kurzform: dO#

Zweck: Eröffnen einer sequentiellen oder relativen Datei zum Schreiben oder Lesen.

Beschreibung: Der "dopen"-Befehl stellt die Verbindung zwischen einer logischen Datei-Nummer und relativen oder sequentiellen Dateien auf der Disk her und reserviert Pufferraum in der Disk-Einheit für die Bearbeitung der eröffneten Dateien.
Der "dopen"-Befehl, bei einer bereits eröffneten Datei angewendet, bewirkt einen Fehler und das Abschließen der Datei. Anschließend läßt sich diese wieder eröffnen.
Ohne Angabe gilt Laufwerk = 0, Disk-Einheit = 8.
Entsprechend dem Dateityp ist das Befehlsformat etwas unterschiedlich:

Für **sequentielle** Dateien gilt ",w" = "schreiben", ",r" oder Weglassen dieses Parameters = "lesen".
Beim Eröffnen einer nicht vorhandenen Datei zum Lesen erscheint die Fehlermeldung FILE NOT FOUND.

Für **relative** Dateien ist die Angabe der Satzlänge beim Anlegen erforderlich (1 max.254).

Beispiel: **1. Sequentielle Datei**

dopen#1,"Konto",dO,w

Die sequentielle Datei "Konto" wird unter der logischen Datei-Nummer 1 auf der Diskette in Laufwerk 0 zum Schreiben eröffnet. Der Datei-Name "Konto" darf noch nicht auf der Diskette vorhanden sein, da sonst die Meldung FILE EXISTS erscheint. Soll eine bereits bestehende Datei überschrieben werden, so ist vor den eigentlichen Namen ein "§" zu setzen (z.B. "§Konto").

Beispiel: **2. Relative Datei**

dopen#5,"Lager",d1,l128

Die relative Datei "Lager" wird unter der logischen Datei-Nummer 5 auf Laufwerk 1 mit einer Satzlänge von 128 Zeichen eröffnet. Besteht diese Datei noch nicht, wird sie nun erzeugt. Besteht die Datei bereits, wird sie sowohl zum Lesen als auch zum Schreiben eröffnet.

Bemerkung: Ist die logische Datei-Nummer kleiner als 128, erfolgt nach einem "print#"-Befehl ein "Carriage-Return" chr$(13) . Ist sie größer als 128, erfolgt zusätzlich ein "Line-Feed" chr$(10) . "Carriage-Return" kann durch ein ";" unterdrückt werden.

dsave

Format: dsave [d<x>] [onu<y>] ,"<Datei-Name>"

Kurzform: dS

Zweck: Speichern eines Programms auf Diskette.

Beschreibung: Der "dsave"-Befehl überträgt ein Programm aus dem Speicher des Rechners in eine Datei auf der Diskette. Das DOS kennzeichnet diese Datei als Programm-Datei-Typ.
Der Programmname darf bis zu 16 Zeichen lang sein (einschließlich Leerzeichen).
Ohne Angabe gilt Laufwerk = 0, Disk-Einheit = 8.

Beispiel: dsave d1,"Lagerverwaltung"

Das im Speicher stehende Programm wird unter dem Namen "Lagerverwaltung" auf die Diskette in Laufwerk 1 gespeichert.

Bemerkung: Ein "§" vor dem eigentlichen Programmnamen ermöglicht das direkte Überschreiben einer gleichnamigen Datei (z.B. "§Lagerverwaltung").
Ansonsten darf der gewählte Name nicht bereits auf der Diskette vorhanden sein.

get#

Format: get#<log. Datei-Nummer>,<Variable>

Kurzform: gE#

Zweck: Lesen eines Bytes aus einer Datei (bzw. Puffer) in eine Variable

Beschreibung: Der "get#"-Befehl ist nur gültig, wenn er auf einen Puffer oder eine Datei verweist, die zum Lesen eröffnet wurde.
Ist das gelesene Byte binär Null, so wird einer numerischen Variable der Wert 0, einer Stringvariable ein leerer String zugeordnet.

Beispiel: get#15,a

Ein Byte aus der logischen Datei 15 wird in die numerische Variable "a" gelesen.

Bemerkung: Für nachfolgende Operationen, wie z.B. Vergleiche, muß einer Stringvariablen mit einem leeren String (hervorgerufen durch Lesen von binär Null) ein Wert zugewiesen werden. Dies kann geschehen wie folgt:

```
get#7,b$ : if b$="" then b$=chr$(0)
```

header

Format: header" Disketten-Name " [,d<x>] [onu<y>] [,i ID-Nummer]

Kurzform: hE

Zweck: Formatieren einer neuen Diskette oder Löschen einer beschriebenen Diskette.

Beschreibung: Der "header"-Befehl formatiert eine neue Diskette, d.h. er vollzieht die Sektoraufteilung entsprechend den Anforderungen der Disk-Einheit. Spur- und Sektoradressen werden auf die Diskette geschrieben, sämtliche Blöcke werden mit binären Nullen gefüllt, die BAM wird erzeugt und das Format des Inhaltsverzeichnisses wird angelegt.
Da durch die "header"-Operation bei einer bereits benutzten Diskette alle Daten gelöscht werden, erscheint vor der Ausführung (im Direkt-Modus) die Frage "ARE YOU SURE?" (sind Sie sicher?). Erst nach Beantworten mit "y" wird die "header"-Operation ausgeführt.
Die ID-Nummer muß 2-stellig angegeben werden.
Ohne Angabe des Parameters ",i ID-Nummer " wird der Diskette nur der angegebene Name zugewiesen und das Inhaltsverzeichnis und BAM werden gelöscht.
Der Name darf bis zu 16 Stellen lang sein.
Ohne Angabe gilt Laufwerk = 0 und Disk-Einheit = 8.

Beispiel: header"Inventur 1982",d0 onu9,i02

Die Diskette in Laufwerk 0, Disk-Einheit 9 wird formatiert und bekommt den Namen "Inventur 1982" und die ID "02"

initialisieren

Format: print# log. Kanal-Nummer des Fehlerkanals ,"i<x>"

Kurzform: oP log. Fehlerkanal-Nr. , y ,15,"i x "

Zweck: Übernimmt die BAM von der Diskette oder Platte des angesprochenen Laufwerks in den Arbeitsspeicher der Disk-Einheit.

Beschreibung: Nach dem Einsetzen einer Diskette muß das Laufwerk initialisiert werden, d.h. die BAM wird in den Speicher der Disk-Einheit geladen (siehe auch Bemerkung).
Steht die aktuelle BAM nicht im Speicher, so geht das DOS von einer anderen (nicht zutreffenden) Diskettenbelegung aus, was die Fehlermeldung "DISK ID MISMATCH" oder Datenverlust hervorrufen kann.
Das Weglassen der Laufwerk-Nummer bewirkt bei Doppel-Laufwerken die Initialisierung beider Laufwerke.

Beispiel:

```
open 22,8,15
print#22,"i0"
close 22
```

oder:

```
open 22,8,15,"i0"
close 22
```

Laufwerk 0 der Disk-Einheit 8 wird initialisiert.

Bemerkung: Die 4040 und 4031 Disk-Einheiten prüfen die Disketten-ID bei jeder Disk-Adressierung. Wird eine neue ID erkannt, wird automatisch eine Initialisierung durchgeführt. Hat eine Diskette die gleiche ID wie die zuvor benutzte, so wird der Diskettentausch nicht erkannt. Wird dann durch das Programm keine Initialisierung durchgeführt, so kann dies zu Datenverlust führen.
Die Disk-Einheiten 8050 und 8250 erkennen das Einsetzen bzw. Herausnehmen einer Diskette. Die Initialisierung läuft dann automatisch, wenn die Laufwerk-Tür geschlossen wird (bei 8050 Micropolis) oder bei der ersten Laufwerk-Adressierung (bei 8050 und 8250 Tandon).

input#

Format: input# <log. Datei-Nummer>,<Variable>

Kurzform: iN

Zweck: Lesen von Daten aus einer relativen oder sequentiellen Datei mit Zuordnung zu einer Variablen.

Beschreibung: Der "input#"-Befehl ist nur gültig, wenn auf eine Datei verwiesen wird, die zum Lesen eröffnet wurde.
Der Variablen-Typ muß den zu lesenden Daten entsprechen.
Die maximale Stringlänge beim "input#" beträgt 80 Zeichen.
Es gelten grundsätzlich die gleichen Regeln wie beim "input"-Befehl vom Bildschirm.
Führende Leerzeichen, "Carriage-Return" und "Line-Feed" werden ignoriert. Das erste Zeichen, das keines der eben erwähnten ist, gilt als Beginn der Daten.
Die Anzahl der Variablen sollte der Anzahl der beim "print#"-Befehl benutzten entsprechen. Wurden hier jedoch Variablen benutzt, die länger als 8o Zeichen sind, so muß anstelle des "input"-Befehls mit dem "get"-Befehl gearbeitet werden.
Während des Lesevorgangs wird der Statusvariablen "st" der Wert 0 zugeordnet. Nachdem das letzte Zeichen der Datei gelesen wurde, erhält diese den Wert 64. Wird versucht, weiterzulesen, ist der Wert von st = 2.

Beispiel: input#5,a$

Die Daten aus der logischen Datei Nummer 5 werden in die Variable a$ gelesen. Die Eingabe wird durch beendet durch:
ein Maximum von 80 Zeichen
ein Carriage-Return chr$(13)
ein Komma chr$(44)

Beispiel: input#5,a$,b$,c$

Die Daten mußten hier beim Schreiben durch Trennzeichen separiert werden, um sie getrennt zurück zu erhalten.
Keiner der einzelnen Strings darf länger als 80 Zeichen sein.

Bemerkung: Besondere Vorsicht ist beim Lesen aus relativen Dateien geboten! Wenn der Satzzeiger auf einen nicht vorhandenen Satz gesetzt ist, kann ein Leseversuch die Datei zerstören. Vor dem "input#" ist deshalb immer die Disk-Statusvariable "ds" oder "ds$" abzufragen. Kein "input" bei ds$ = 50 RECORD NOT PRESENT!

open

Format: open<log. Datei-Nummer>,<Einheit-Nummer>,<Sekundär-Adresse>,
"<Laufwerk>:<Datei-Name>, <Datei-Typ> , <Option> "

open<log. Datei-Nummer>,<Einheit-Nummer>,<Sekundär-Adresse>,
"<Befehls-String>"

Kurzform: oP

Zweck: Eröffnen eines E/A-Kanals über den IEEC-Bus oder ein internes Gerät.

Beschreibung: Die logische Datei-Nummer muß einen Wert zwischen 1 und 255 haben.
Logische Datei-Nummern größer als 128 senden mit jedem "print#"-Befehl "Carriage-Return" und "Line-Feed". Bei Nummern kleiner als 128 wird nur "Carriage-Return" gesendet. Dieses kann mit einem Semikolon hinter der Anweisung unterdrückt werden.

Für Datei-Typ gilt: s = sequentielle Datei
p = Programm-Datei
u = User-Datei
l Satzlänge = relative Datei

Option: r = read (lesen)
w = write (schreiben)
keine Angabe = read (nicht bei rel. Dateien)

Beispiel:
```
10 open7,8,2,"1:Datei,s,w"
20 for i=1 to 10
30 i$ = chr$(i)
40 print#7,i$;chr$(13);
50 next i
60 close7
```

Bemerkung: Beim Eröffnen einer Datei mit Schreibzugriff ist vor der Laufwerk-Nummer "§" bzw. das "At-Sign" anzugeben (nicht bei rel. Dateien):

```
10 open7,8,2,"§1:Datei,s,w"
```

Beispiel:
```
open 1,8,15,"i0" : close1
```

Laufwerk 0 in Disk-Einheit 8 wird initialisiert.

Beispiel:
```
open 1,8,10,"#"
```

Ein freier Puffer wird eröffnet.

Beispiel:
```
open 1,8,5,"#9"
```

Puffer 9 wird eröffnet.

print#

Format: print#<log. Datei-Nummer>,<Variable>

Kurzform: pR

Zweck: Übertragung von Daten zu einer vorher eröffneten Datei.

Beschreibung: Der "print#"-Befehl bezieht sich auf die mit "dopen" angegebene logische Datei-Nummer.
Ist die logische Datei-Nummer größer als 128, so wird zusätzlich zum "Carriage-Return" ein "Line-Feed" übertragen (BASIC 4.0).
Ein ";" hinter der Variablen unterdrückt "Carriage-Return".

Beispiel: print#8,y$

Der Inhalt von y$ wird zur Datei Nummer 8 geschrieben.

Beispiel: print#5,a$;b$;c$

Die Variablen a$, b$ und c$ werden als zusammenhängender Datenstring gespeichert (Variable-Namen mit Semikolons getrennt).

Beispiel: print#11,achr(13)bchr(13)c$

Die Variablen a$, b$ und c$ werden durch "Carriage-Return" getrennt. Mit "input" oder "get" kann man sie separat wieder einlesen.

record#

Format: record# <log. Datei-Nummer>,<Satz-Nummer> [,<Byte>]

Kurzform: reC#

Zweck: Positionieren des Datensatz-Zeigers einer relativen Datei vor nachfolgenden Befehlen wie "get#", "input#" oder "print#".

Beschreibung: Die Satz-Nummer darf den Wert 65535, die Byte-Nummer den Wert 255 nicht überschreiten.
Wird der Datensatz-Zeiger hinter den letzten vorhandenen Satz positioniert, erhält "ds$" den Wert 50 RECORD NOT PRESENT.
Durch einen nachfolgenden "print"-Befehl werden bis zur geforderten Satznummer die fehlenden Sätze erzeugt. Die Datei wird dadurch entsprechend vergrößert.
Ohne Angabe der Byte-Nummer wird der Datensatz-Zeiger zum ersten Byte des angegebenen Datensatzes positioniert.

Beispiel:
```
record#15,12,8
```

Der Datensatz-Zeiger wird zu Byte 8 im Satz 12 der logischen Datei-Nummer 15 positioniert.

Beispiel:
```
record#1,25
input#1,a$
```

Der nächste Datensatz wird als String gelesen und der Variablen a$ zugewiesen.

rename

Format: rename [d<x>] [onu<y>] ,"<alter Datei-Name>"
to "<neuer Datei-Name>"

Kurzform: reN

Zweck: Ändern eines Disk-Datei-Namens

Beschreibung: Die Datei, deren Name geändert werden soll, muß vorher mit "close" oder "dclose" abgeschlossen werden. Der neue Name darf noch nicht auf der Diskette vorhanden sein. Fehlermeldung: FILE EXISTS.
Ohne Angabe gilt Laufwerk = 0 und Disk-Einheit = 8.

Beispiel: rename d1,"Möbel" to "Einrichtung"

In Laufwerk 1, Disk Einheit 8 wird die Datei "Möbel" in "Einrichtung" umbenannt.

scratch

Format: scratch [d<x>] [onu<y>] ,"<File-Name>"

Kurzform: sC

Zweck: Löschen von Dateien im Inhaltsverzeichnis

Beschreibung: Der "scratch"-Befehl entfernt nicht mehr erwünschte Dateien aus dem Inhaltsverzeichnis. Die Daten werden **nicht** gelöscht, sondern der Datei-Typ wird auf "0" gesetzt. Die benutzten Blöcke werden in der BAM als verfügbar markiert, was ein späteres Überschreiben ermöglicht.
Vor der Ausführung (im Direkt-Modus) fragt das System: ARE YOU SURE? (Sind Sie sicher?). Nach Betätigen von "y" wird der "scratch"-Befehl ausgeführt.
Ohne Angabe der Laufwerk-Nummer wird in beiden Laufwerken nach der Datei gesucht.
Ohne Angabe gilt Disk-Einheit = 8.

Beispiel: scratch d1,"Datei-X"

Die Datei "Datei-X" in Laufwerk 1 wird aus dem Inhaltsverzeichnis gelöscht und die belegten Blöcke werden freigegeben.

verify

Format: verify <Einheit-Nummer> ,"d<x>:<Datei-Name>"

Kurzform: vE

Zweck: Vergleichen des Programms im Speicher mit der entsprechenden Programm-Datei auf der Diskette.

Beschreibung: Beim Feststellen einer Ungleichheit kommt die Meldung: VERIFY ERROR.

Beispiel: verify 8,"dO:Kontoführung"

Die Programm-Datei "Kontoführung" in der Disk-Einheit 8, Laufwerk 0 wird mit dem derzeit im Rechner-Speicher stehenden Programm verglichen.

Teil 4: Disk-Dienstprogramme

Die Disk-Dienstprogramme bestehen aus folgenden Befehlen:

Befehl:	Kurzform:	Format:	Seite:
block-read	b-r	"b-r:"<Kanal>;<Laufwerk>;<Spur>;<Sektor>	40
block-write	b-w	"b-w:"<Kanal>;<Laufwerk>;<Spur>;<Sektor>	41
block-execute	b-e	"b-e:"<Kanal>;<Laufwerk>;<Spur>;<Sektor>	42
buffer-pointer	b-p	"b-p:"<Kanal>;<Zeiger-Position>	43
block-allocate	b-a	"b-a:"<Laufwerk>;<Spur>;<Sektor>	44
block-free	b-f	"b-f:"<Laufwerk>;<Spur>;<Sektor>	45
*(memory-write)	m-w	"m-w"<Adr.-Byte rechts><Adr.-Byte links> Anzahl Bytes Daten-Bytes	46
*(memory-read)	m-r	"m-r"<Adr.-Byte rechts><Adr.-Byte links>	47
*(memory-execute)	m-e	"m-e"<Adr.-Byte rechts><Adr.-Byte links>	48
user<x>	u<x>	"u<x>:"<Kanal>;<Laufwerk>;<Spur>;<Sektor>	49-51

Bemerkung:

1. **Die aufgeführten Befehle können nur in Verbindung mit dem "print#"-Befehl über den Befehlskanal (Sekundär-Adresse 15) angewandt werden.** Das vollständige Format sieht dann beispielseise für den "block-read" aus wie folgt:

   print#<log.Datei-Nummer>,"b-r"<Laufwerk>;<Spur>;<Sektor>

2. Bei den "memory"-Befehlen ist zu beachten, daß die Werte über die "chr$(<xx>)"-Anweisung angegeben werden müssen.
   Von den beiden Adress-Bytes ist als erstes das rechte der Hexadezimal-Adresse, umgerechnet in den Dezimalwert, anzugeben.

   Beispiel: Adresse = 120D (hexadezimal)
   rechtes (low) Byte = 0D (hex.) = 13 (dezimal)
   linkes (high) Byte = 12 (hex.) = 18 (dezimal)
   Die Adressangabe sieht daher folgendermaßen aus:
   "m-w"chr$(13)chr$(18).....

3. Außer den gezeigten Formaten gibt es noch andere Darstellungsarten, die jedoch keine funktionellen Unterschiede bewirken. So kann z.B. ein "block-read" folgendermaßen dargestellt werden:

   "block-read:"2,1,4,0
   "b-r"2,1,4,0
   "b-r:"2,1,4,0
   "b-read:"2;1;4;0

* Bei den "memory"-Befehlen ist im Befehlsformat nur die Kurzform gültig !

block-read

Format: print#<log.Datei-Nummer>,"block-read:"<Kanal>,
<Laufwerk>,<Spur >,<Sektor>

Kurzform: "b-r"

Zweck: Lesen eines Blocks von der Diskette

Beschreibung: Dieser Dienstleistungs-Befehl ermöglicht direkten Zugriff zu einem Block auf der Diskette. In Verbindung mit anderen Block-Befehlen läßt sich über BASIC ein Direktzugriff-Datei-System aufbauen. Wird das letzte Datenzeichen mit "get#" oder "input#" gelesen, wird das Statuswort (ST) im Rechner auf 64 gesetzt.

Beispiel:

```
10 open 12,8,15,"i0"      : rem Befehlskanal eröffnen
20 open 5,8,2,"#"         : rem Datenpuffer eröffnen
30 print#12,"b-r"2,0,1,1 : rem Block lesen
40 get#5,a$               : rem Byte 1 wird a$ zugewiesen
.. ...                    ...
.. ...                    ...
70 dclose                 : rem Kanäle werden abgeschlossen
```

Bemerkung: Das erste Byte (Byte 0) dient zur Angabe der Datenlänge im Block. Der Pufferzeiger steht auf der 2. Position (Byte 1). Die Blockdaten lassen sich bis zu der in Byte 0 festgelegten Datenlänge lesen.

block-write

Format: print#<log.Datei-Nummer>,"block-write:"<Kanal>
<Laufwerk>,<Spur>,<Sektor>

Kurzform: "b-w"

Zweck: Schreiben eines Blocks auf Diskette

Beschreibung: Mit dem "block-write"-Befehl wird der augenblickliche Wert des Puffer-Zeigers auf die Anfangsposition 0 des neuen Puffers geschrieben. Der Puffer wird dann auf den angegebenen Block geschrieben, so daß auf Block-Position 0 die Datenlänge angegeben ist. Der Pufferzeiger steht dann auf Position 1.

Beispiel:

```
10 open 14,8,15,"i0"
20 open 6,8,2,"#"
30 print#6,a$
40 print#14,"b-w"2,0,1,1
50 close6 : close14
```

Inhalt von a$ wird in Laufwerk 0 auf Spur 1, Sektor 1 geschrieben.

block-execute

Format: print#<log.Datei-Nummer>,"block-execute:"<Kanal>,
<Laufwerk>,<Spur>,<Sektor>

Kurzform: "b-e"

Zweck: Programm-Routinen (Maschinencode) werden von Diskette in den Disk-Laufwerk-Speicher geladen und ausgeführt.

Beschreibung: Nachdem der Block in den angegebenen Puffer geladen wurde, beginnt der Datei-Schnittstellen-Controller mit der Ausführung des Inhalts. Die Ausführung muß mit einer RTS (Return From Subroutine) Instruktion beendet werden.

Beispiel:

```
10 open 9,8,15,"i0"
20 open 5,8,6,"#"
30 print#9,"b-e"6,0,1,10
40 dclose
```

Der Block in Laufwerk 0, Spur 1, Sektor 10 wird in den zu Kanal 6 gehörenden Puffer gelesen. Der Inhalt wird beginnend mit Position 0 im Puffer ausgeführt.

Buffer-pointer

Format: print#<log.Datei-Nummer>,"buffer-pointer:"<Kanal>,
<Zeigerposition>

Kurzform: "b-p"

Zweck: Positionieren des Puffer-Zeigers

Beschreibung: Der Pufferzeiger in dem zum angegebenen Kanal gehörenden Puffer wird auf eine neue Position gesetzt. Dies ist nützlich, wenn auf einzelne Felder oder einen Satz innerhalb eines Blocks zugegriffen oder ein Block in zwei Sätze geteilt werden soll.

Beispiel:

```
10 open 4,8,15,"i0"
20 open 5,8,3,"#"
30 print#4,"b-p"3,58
40 dclose
```

Der Puffer-Zeiger in dem zu Kanal 3 gehörenden Puffer wird auf Position 58 gebracht.

block-allocate

Format: print#<log.Datei-Nummer>,"block-allocate:"
<Laufwerk>,<Spur>,<Sektor>

Kurzform: "b-a"

Zweck: Ein Block wird in die BAM als benutzt eingetragen.

Beschreibung: Der "block-allocate"-Befehl fordert das DOS auf, einen Block als "in Gebrauch" zu kennzeichnen und die BAM zu aktualisieren. Beim Schreiben von Dateien auf die Diskette wird der so gekennzeichnete Block übersprungen. Die BAM wird beim Abschließen einer Ausgabedatei oder eines Befehlskanals auf die Diskette geschrieben.
Ist der im "block-allocate"-Befehl spezifizierte Block bereits als "in Gebrauch" gekennzeichnet, bringt der Fehler-Kanal einen NO BLOCK-Fehler (Fehler Code 65) und zeigt den nächsten verfügbaren Block an. Es werden nur Blöcke angezeigt, die eine größere Nummer (höherer Sektor oder Spur) haben als der spezifizierte.

Beispiel:

```
10 open 7,8,15,"i0"
20 print#7,"b-a"0,12,19
30 dclose
```

Das Kennzeichnen von Block (Sektor) 19 auf Spur 10 im Laufwerk 0 wird gefordert. Ist dieser Block bereits in Gebrauch, so wird mit der Fehlermeldung der nächste verfügbare Block angezeigt.

```
25 input#7,fn,fm$,sp,se
```

Der nächste verfügbare Sektor, beziehungsweise die nächste Spur, wird in "se" / "sp" gelesen. In "fn" steht die Fehlernummer, in "fm$" die Fehlermeldung.

block-free

Format: print#<log.Datei-Nummer>,"block-free:"<Laufwerk>,
<Spur>,<Sektor>

Kurzform: "b-f"

Zweck: Freigeben von Blöcken in der BAM

Beschreibung: Der "block-free"-Befehl arbeitet umgekehrt wie der "block-allocate"-Befehl. Blöcke die als "in Gebrauch" gekennzeichnet sind, werden wieder als verfügbar gekennzeichnet und somit zum Überschreiben freigegeben.

Beispiel:
```
10 open 10,8,15,"i0"
20 print#10,"b-f"1,14,12
30 dclose 10
```

Block auf Spur 14, Sektor 12 in Laufwerk 1 wird freigegeben.

memory-write

Format: print#<log.Datei-Nummer>,"m-w"<Adr.-Byte rechts>
<Adr.-Byte links><Anzahl Daten-Bytes><Daten-Bytes>

Kurzform: **"m-w"** (einzige zulässige Schreibweise!)

Zweck: Direktzugriff zum Speicher.

Beschreibung: Spezielle Programm-Routinen können in den Speicher des Disk-Laufwerkes geschrieben und dann über den "memory-execute"-Befehl oder einen der "user"-Befehle ausgeführt werden. Mit jedem "m-w"-Befehl können bis zu 34 Bytes abgelegt werden. Der hexadezimale Wert der DOS-Speicher-Adresse muß in das rechte und linke Byte zerlegt werden. Dann wird zuerst das **rechte** Byte in die entsprechende Dezimalzahl umgewandelt und über die "chr$"-Funktion angegeben. Dasselbe geschieht danach mit dem linken Byte.

Beispiel:

```
10 open 13,8,15,"i0"
20 print#13,"m-w"chr$(0)chr$(18)chr$(4)chr$(32)chr$(0)
     chr$(17)chr$(96)
30 close 13
```

Vier Bytes werden in Puffer 2 (Adresse 1200 hex.) geschrieben.

Beispiel: Eine andere Anwendung des "m-w"-Befehls ist die zeitweise Änderung der physikalischen Geräte-Nummer der Disk. Normalerweise haben alle Disk-Einheiten die Nummer 8. Werden zwei oder mehr Disk-Einheiten an einen Rechner angeschlossen, muß jede Disk-Einheit eine eigene Geräte-Nummer bekommen.
Das folgende Programmfragment dient zum Ändern der Geräte-Nummer für die Geräte: 4040, 8050, 8250, D9060, D9090.

```
10 open10, alte Geräte-Nummer ,15
20   print#10,"m-w"chr$(12)chr$(0)chr$(2)chr$( neue   Geräte-
     Nummer  + 32)chr$( neue Geräte-Nummer  + 64)
30 close10
```

Für die 2031 Disk-Einheiten gilt:

```
20 print#10,"m-w"chr$(119)chr$(0)chr$(2)chr$( neue Einheit-
     Nummer + 32 )chr$( neue Einheit-Nummer + 64 )
```

Folgende Reihenfolge sollte eingehalten werden:
1. Nur die erste Disk-Einheit einschalten
2. Ablauf des Programms
3. Einschalten der nächsten Disk-Einheit.

Andere Möglichkeit:
Die von der Änderung nicht betroffenen Geräte sind während des Programmablaufs vom IEEC-Bus zu trennen.

Nach dem Aus- und wieder Einschalten ist die alte Geräte-Nummer wieder vorhanden.

memory-read

Format: print#<log.Datei-Nummer>,"m-r"<Adr.-Byte rechts>
<Adr.-Byte links>

Kurzform: "m-r" (einzig zulässige Schreibweise !)

Zweck: Lesen eines Bytes im DOS-Speicher

Beschreibung: Das auf der angegebenen Adresse stehende Byte im DOS-Speicher wird gelesen. Es können auch Variablen des DOS oder die Pufferinhalte gelesen werden. Der "m-r"-Befehl verändert den Inhalt des Fehlerkanals, weil dieser zur Datenübertragung zum Rechner benutzt wird. Das nächste "get#" vom Fehlerkanal (Sekundär-Adresse 15) überträgt das Byte. Weiteres Lesen mit "get#" liefert die folgenden Bytes.
Ein "input#" vom Fehlerkanal sollte nach einem "m-r"-Befehl nicht ausgeführt werden, bis ein anderer (nicht "memory"-) DOS-Befehl ausgeführt wurde.

Beispiel:
```
10 open 5,8,15,"i0"
20 print#5,"m-r"chr$(128)chr$(0)
30 get#15,a$
40 dclose#5
```

Das Byte von Adresse 0080 hex. wird nach a$ gelesen.

memory-execute

Format: print#<log.Datei-Nummer>,"m-e"<Adr.-Byte rechts>,
<Adr.-Byte links>

Kurzform: "m-e" (einzig zulässige Schreibweise !)

Zweck: Programm-Subroutinen im DOS-Speicher werden ausgeführt.

Beschreibung: Mit dem "m-e"-Befehl werden Subroutinen, die im DOS-Speicher stehen, ausgeführt. Um zum DOS zurückzukehren, muß die Subroutine mit einer RTS-Instruktion abgeschlossen werden.

Beispiel:

```
10 open 7,8,15,"i0"
20 print#7,"m-e"chr$(128)chr$(49)
30 dclose 7
```

Bei Adresse 3180 hex. wird mit der Ausführung der Programm-Routine begonnen.

"user"-Befehle

Die "user"-Befehle liefern über eine Sprungtabelle (mittels des "user"-Zeigers) eine Verbindung zum 6502 Maschinen-Code. Das zweite Zeichen in diesen Befehlen dient als Tabellenindex. Die ASCII-Zeichen "0" bis "9" oder "a" bis "o" können benutzt werden. Null setzt den "user"-Zeiger zu einer Standard-Sprungtabelle, die Verbindungen zu Spezialroutinen enthält.

Die speziellen "user"-Befehle u1 (oder ua) und u2 (oder ub) können in allen DOS-Versionen als Ersatz für "block-read" und "block-write" benutzt werden. Wegen Fehlern im DOS 2.1 arbeiten die "b-r"- und "b-w"-Befehle bei 4040 Disk-Einheiten nicht korrekt. Diese Befehle müssen hier durch "u1" und "u2" ersetzt werden.

Standard "user"-Sprungtabelle

Standard-Bezeichnung	Alternativ-Bezeichnung	Funktion
u1	ua	Ersatz für "block-read"
u2	ub	Ersatz für "block-write"
u3	uc	Sprung nach 1300 hex.
u4	ud	Sprung nach 1303 hex.
u5	ue	Sprung nach 1306 hex.
u6	uf	Sprung nach 1309 hex.
u7	ug	Sprung nach 130C hex.
u8	uh	Sprung nach 130F hex.
u9	ui	Sprung nach 10F0 hex. (NMI)
u:	uj	"Power-Up"-Vector

Die "u3" bis "u9"-Befehle sind anwender-definiert. Die Sprungadressen liegen im Pufferbereich des Disk-RAM. Vom Anwender geschriebene Routinen können dort mittels "m-w"- oder "b-r"-Befehlen abgespeichert werden.
Die Sprungadresse 10F0 hex. ist die Lokation des "NMI Interrupt Handler".
Die "u:" oder "uj"-Befehle bewirken, daß die Disk-Einheit die Netzeinschalt-Sequenz ausführt und setzt die Einheit-Nummer zurück. Die Laufwerke müssen initialisiert werden, bevor weitere Befehle erfolgen können.

u1

Format: print#<log.Datei-Nummer>,"u1:"<Kanal>,
<Laufwerk>,<Spur>,<Sektor>

Kurzform: "u1"

Zweck: Lesen eines Blocks auf der Diskette

Beschreibung: "u1" bringt den Zeichenzähler auf den Wert 255 und liest den gesamten Block in den Speicher. Dies erlaubt den Zugriff zu allen Bytes im Block, einschließlich Spur- und Sektor-Verbindungs-Zeiger. Der Puffer-Zeiger steht auf Byte 0.

Beispiel:

```
10 open 14,8,15,"i0"
20 open 3,8,2,"#"
30 print#14,"u1:"2,1,4,9
40 get#3,a$
.. ...
.. ...
.. ...
90 dclose
```

Der Block auf Spur 4, Sektor 9 in Laufwerk 1 wird gelesen. Das erste Byte wird "a$" zugewiesen.

u2

Format: print#<log.Datei-Nummer>,"u2:"<Kanal>,
<Laufwerk>,<Spur>,<Sektor>

Kurzform: "u2"

Zweck: Schreiben eines Blocks auf Diskette

Beschreibung: "u2" schreibt einen Puffer in einen Block auf der Diskette, ohne den Inhalt von Position 0 zu verändern, wie z.B. beim "b-w" Befehl. Dies ist nützlich, wenn ein Block gelesen (mit "b-r"), aktualisiert ("b-p" zum Feld und "print#") und zurückgeschrieben wird.

Beispiel:

```
10 open 11,8,15,"i0"
20 open 13,8,2,"#"
30 print#13,a$
40 print#11,"u2:"2,0,7,12
50 dclose
```

Inhalt von a$ und restlicher Pufferinhalt wird auf Spur 7, Sektor 12 in Laufwerk 0 geschrieben.

Teil 5: Dateiverwaltung

Relative Dateien

Der Direktzugriff ist eine Methode, die dem Programmierer erlaubt, einen Zeiger auf einen beliebigen Satz auf der Diskette, relativ zum Anfang der Datei, zu positionieren. Verglichen mit der sequentiellen Suchmethode, bei der die gesamte Datei nach der gewünschten Information durchsucht werden muß, bedeutet der Direktzugriff eine große Zeitersparnis.

Interne Verwaltung von relativen Dateien

Die drei Hauptkomponenten einer relativen Datei sind der Super-Seitensektor (nur bei DOS 2.7 und DOS 3.0), die Seitensektor-Verkettung der Blöcke und die Daten-Block-Verkettung. Alle sind durch Vorwärts-Zeiger, ähnlich denen der sequentiellen Dateien, miteinander verbunden.

Der Super-Seitensektor zeigt zum ersten Seitensektor in einer Gruppe von Seiten-Sektoren. Jeder Seitensektor zeigt zu einem anderen in der gleichen Gruppe und zeigt zu einer Daten-Block-Verkettung. Satzformate, in der Länge festgelegt, können sich von 1 bis 254 Bytes erstrecken. Die Anzahl der Sätze ist auf die Anzahl begrenzt (unter DOS 2.1 und 2.5), die in 720 Datenblöcken enthalten sein kann, weil jeder Seitensektor maximal 120 Datenblock-Zeiger enthalten kann und vom DOS nur 6 Seitensektoren angelegt werden. Die Anzahl der Sätze unter DOS 2.7 und DOS 3.0 ist auf die Kapazität der Diskette begrenzt, sollte in der Praxis jedoch 65.535 Bytes nicht übersteigen.

Die Seitensektoren enthalten keine Satzinformationen, sondern die Verweise zu den Datenblöcken. Die Position des Satzzeigers wird in Abhängigkeit vom Satzformat errechnet.

Der Seitensektor enthält außerdem eine Tabelle der Zeiger zu allen anderen Seitensektoren innerhalb der Datei. Um von einem Seitensektor zu einem anderen zu gelangen, wird vom DOS der Zeiger abgefragt und der entsprechende Seitensektor in den Speicher geladen. Unter Benutzung der im Seitensektor enthaltenen Informationen kann der Datenblock-Zeiger gesetzt werden, um den Datenblock, der den Satz enthält, zu lesen.

Die relativen Datei-Datenblock-Zeiger in den Seitensektoren erlauben dem DOS von einem Satz zu einem anderen, innerhalb von zwei Disk-Lesebefehlen, zu gelangen. Dies ist, verglichen mit sequentiellen Methoden, eine beträchtliche Zeitersparnis beim Suchen eines gewünschten Datenblocks.

Jeder Seitensektor enthält Zeiger für 1 bis 120 Datenblöcke. Es gibt 6 Seitensektoren für jede relative Datei unter DOS 2.1 (4040) und DOS 2.5 (8050). Dies liefert eine Dateikapazität von 182.880 Bytes (120 Zeiger / Seitensektor * 6 Seitensektoren * 254 Bytes / Datenblock).
Der Super-Seitensektor des DOS 2.7 (8250) und DOS 3.0 (Festplatten) hat die Möglichkeit, auf 127 Gruppen von Seitensektoren zu verweisen. Das ergibt eine Gesamtkapazität von 23.225.760 Bytes pro Datei (182.880 Bytes * 127 Gruppen von Seitensektoren).

Das Überspannen von Datenblöcken ist eine Schlüsseleinrichtung relativer Dateien, die hilft, die Anzahl der zum Finden und Wiedererhalten der Daten erforderlichen Disk-Schreib-/Lese-Operationen zu verringern. Bevor erklärt wird, wie diese Einrichtung die Zeitausnutzung sehr verbessert, bedarf es der Erklärung, wie E/A-Kanäle durch relative Dateien genutzt werden:

> Wird ein Kanal für eine bereits bestehende Datei eröffnet, wird das DOS zum ersten bereitgestellten Satz positioniert. Die Angabe der Satzlänge ist in der "dopen"-Anweisung nicht erforderlich, sofern die Datei bereits besteht. Das DOS überprüft die Übereinstimmung des Satzformats (sofern angegeben) mit dem im Inhaltsverzeichnis eingetragenen Format der bestehenden Datei. Stimmen die Formate nicht überein, wird eine Fehlermeldung erzeugt.

Relative Dateien erfordern drei Puffer, sequentielle Dateien dagegen benötigen nur zwei Puffer. Weil es im System zwölf Puffer gibt, von denen zwei zum Suchen im Inhaltsverzeichnis und für interne Funktionen benutzt werden, können gleichzeitig nur drei relative Dateien eröffnet werden. Die höchste Anzahl der benutzbaren Puffer ist zehn, was die Gesamtzahl der Kanäle, die eröffnet sein können, begrenzt.

Steht ein Satz im Grenzbereich zwischen zwei Datenblöcken, also fängt er im einen Block an und hört im anderen auf, liest das DOS sowohl den ersten Abschnitt als auch die folgenden Sätze im zweiten Datenblock. In der Praxis überspannen die Sätze der meisten relativen Dateien Datenblöcke. Die einzigen Ausnahmen bilden die Satzlängen 1, 2, 127 und 254. Dies sind ganzzahlige Anteile von 254 und ein Überspannen ist nicht erforderlich.

Diese Methode des Überspannens hat den Vorteil, daß kein zusätzlicher Speicher benötigt wird, abgesehen von dem für die Seitensektor-Blöcke in den relativen Dateien. Wird ein Satz mittels der "print#"-Anweisung geschrieben, wird der Datenblock nicht unmittelbar auf die Diskette geschrieben. Er wird erst dann geschrieben, wenn sich das DOS über den Datenblock, in dem sich dieser Satz befindet, hinausbegibt. Dies kann durch aufeinanderfolgendes Ausgeben zu sequentiellen Sätzen geschehen, oder durch Positionieren zu einem Satz außerhalb des Blocks.

Wegen der Überspannungs-Einrichtung ist es verboten, mehrere Kanäle für dieselbe relative Datei gleichzeitig zu eröffnen, wenn irgend ein Kanal in die Datei schreiben soll. Ein Aktualisieren wird im Pufferbereich eines Kanals durchgeführt, aber die Änderung gelangt erst auf die Diskette, wenn das DOS über den betreffenden Block hinausgeht. Das DOS gibt hier keine Einschränkung, und wenn eine Datei ausschließlich zum Lesen eröffnet ist, kann es vorteilhaft sein, mehrere Kanäle für dieselbe Datei eröffnet zu haben.

Das DOS beendet die Ausgabe zu einem Satz durch die Abfrage des EOI-Signals, das mit jeder "print#"-Anweisung erzeugt wird. Geht die "print#"-Anweisung über das maximale Satzformat hinaus, wird eine Fehlermeldung erzeugt. Jegliche Datenüberlänge wird abgeschnitten, um eine Anpassung entsprechend dem angegebenen Satzformat zu erreichen. Das DOS positioniert zum nächstfolgenden Satz.

Enthält die "print"-Anweisung weniger Zeichen als das Satzformat, werden die verbleibenden Positionen mit Nullen oder binär Null gefüllt. Beim Lesen eines solchen Satzes, wird das EOI-Signal vom DOS zum Rechner erzeugt, wenn das letzte "nicht-Null-Byte" übertragen ist. Soll binäre Information gespeichert werden, muß ein Satzbegrenzer, wie "Carriage Return", benutzt werden. Das Satzformat muß um ein Zeichen vergrößert werden, um das Begrenzungszeichen unterzubringen.

Erstellen einer relativen Datei

Wird eine relative Datei erstmals eröffnet, sollte sie vom Programmierer initialisiert werden, um später einen schnelleren Zugriff zu ermöglichen und um sicherzustellen, daß das DOS ausreichend Platz auf der Diskette für spätere Dateneingaben reserviert. Eine relative Datei kann dadurch initialisiert werden, daß sie eröffnet wird, der Dateizeiger zur höchsten in der Datei zu erwartenden Satznummer positioniert wird, dieser Satz ausgegeben wird und die Datei dann abgeschlossen wird.

Beispiel:

```
dopen#1,"DATEI1",d0,150
record#1,100
print#1,chr$(255)
dclose#1
```

In dem Beispiel erzeugt "dopen" eine Datei auf Laufwerk 0, Disk-Einheit 8 mit dem Dateinamen "DATEI1" und einer Satzlänge von 50.

Die "record#"-Anweisung positioniert den Dateizeiger zu Satz Nummer 100, der noch nicht existiert. Die Fehlermeldung 50 RECORD NOT PRESENT wird an dieser Stelle vorkommen, sollte aber eher als Warnung anstatt als Fehlerzustand gedeutet werden. Diese Meldung ist eigentlich als Warnung gedacht, wenn das erste Mal auf einen neuen Satz zugegriffen wird. Sie besagt, daß keine "input" oder "get"-Operation vorgenommen werden sollte.

Die "print#"-Anweisung bewirkt, daß Satz Nummer 100 geschrieben wird. Während dieser Schreiboperation entdeckt das DOS, daß die Sätze 1 bis 99 noch nicht existieren und initialisiert diese dadurch, daß chr$(255) in die erste Stelle eines jeden Satzes geschrieben wird. Während diesem Prozeß werden auch alle notwendigen Seitensektoren und Daten-Block-Zeiger hergestellt.

Während das DOS neue Datenblöcke für relative Dateien erzeugt, wird die angeforderte Satzanzahl mit der Anzahl der verbleibenden Datenblöcke auf der Diskette verglichen. Stehen zuwenig Datenblöcke zur Verfügung, wird die Fehlermeldung 52 FILE TOO LARGE erzeugt.

Die "dclose"-Anweisung schließt die Datei ab und bewirkt, daß ein entsprechender Bereich in der BAM zugewiesen wird. Außerdem aktualisiert Sie den Blockzähler im Eintrag des Inhaltsverzeichnisses.

Nachdem die Datei initialisiert wurde, können Daten in die Datei geschrieben werden. Das Initialsieren einer Datei in dieser Art und Weise braucht nur einmal durchgeführt werden.

Erweitern einer relativen Datei

Für das Erweitern einer bestehenden Datei wird das gleiche Verfahren wie beim Erstellen einer Dtaei angewendet. Die Satznummer wird entsprechend der größeren Anzahl der Sätze geändert.

Bei der "dopen"-Anweisung kann die Angabe der Satzlänge entfallen. Falls vorhanden, muß sie mit der Länge der vorhandenen Sätze übereinstimmen, da sonst die Fehlermeldung 50 RECORD NOT PRESENT erzeugt wird.

Wenn eine Datei auf diese Weise erweitert wurde, sind die notwendigen Seitensektoren auch hergestellt. Die Seitensektoren bedürfen keiner Beachtung des Anwenders, da diese automatisch vom DOS erzeugt werden.

Zugriff auf eine relative Datei

Für eine sinnvolle Anwendung von relativen Dateien muß man darauf zum Lesen und Schreiben zugreifen können. Vor beiden Operationen wird mittels der "record#"-Anweisung der DOS Datei-Zeiger gesetzt und somit auf den gewünschten Satz positioniert.
Der Parameter für die Satznummer kann eine Konstante oder eine BASIC-Variable (in Klammern) sein:

```
dopen#1,"DATEI1",d0,l50
record#1,25
   oder: rn=25 : record#1,(rn)
print#1,"Frankfurt/M."
dclose#1
```

Der nun entstandene Satz sieht aus wie folgt:

```
         1         2         3         4         5
12345678901234567890123456789012345678901234567890
--------------------------------------------------------------------------------
Frankfurt/M.*
--------------------------------------------------------------------------------
```

* steht für Carriage Return chr$(13)

Das folgende Programm zeigt eine Zuatzeinrichtung der "record#"-Anweisung, die es erlaubt, auf individuelle Bytes innerhalb eines Satzes zum Schreiben oder Lesen zuzugreifen:

```
dopen#1,"DATEI1",d0,l50
record#1,25              (setzt Dateizeiger auf Satz 25)
print#1,"Frankfurt/M."
record#1,25,20           (setzt Zeichenzeiger auf Pos. 20)
print#1,"Hessen"
record#1,25,30           (setzt Zeichenzeiger auf Pos. 30)
print#1,"6000"
dclose#1
```

Der Satz 25 sieht nun folgendermaßen aus:

```
         1         2         3         4         5
12345678901234567890123456789012345678901234567890
--------------------------------------------------------------------------------
Frankfurt/M.*      Hessen*   6000*
--------------------------------------------------------------------------------
```

* steht für Carriage Return chr$(13)

Anmerkung:

Es ist wichtig, daß die Felder hintereinander beschrieben werden, weil das Schreiben eines Bytes am Beginn eines Satzes den Rest des Satzes im DOS-Speicher zerstört. Das bedeutet, daß zuerst auf Byte-Position 1 und dann auf Position 20 geschrieben werden muß und **nicht** zuerst auf Position 20 und dann auf 1!

Weil "Carriage Return" von der BASIC "input#"-Anweisung als Endezeichen erkannt wird, können die Daten aus dem vorigen Beispiel wie folgt eingelesen werden:

```
dopen#1,"DATEI1",d0
record#1,25
input#1,a$                (liest "Frankfurt/M." in a$)
record#1,25,20
input#1,b$                (liest "Hessen" in b$)
record#1,25,30
input#1,c$                (liest "6000" in c$)
dclose#1
```

Der "record#"-Befehl kann weggelassen werden, wenn sequentiell auf die Datei zugegriffen werden soll. Ein Beispiel hierfür ist das Schreiben einer großen Datei auf die Diskette. Angenommen, das Programm hat die Variable d$ bereits als Array dimensioniert, das 100 Elemente enthält. Diese Elemente sollen in den Sätzen Nummer 1 bis 100 von DATEI2 auf die Diskette geschrieben werden. Dies könnte mit folgenden Programmabschnitt ausgeführt werden:

```
dopen#1,"DATEI2",d0,130
for i=1 to 100
print#1,d$(i)
next i
dclose#1
```

Weil der Satzzeiger beim Eröffnen der Datei automatisch auf Satz 1 gesetzt wird, wird Satz 1 zuerst geschrieben. Wird kein "record"-Befehl ausgeführt, positioniert das DOS nach jedem "print#" automatisch auf den nächsten Satz. Aus diesem Grund werden die Elemente des Arrays d$ in die Sätze 1 bis 100 der Datei geschrieben.

Benutzung von 8050 Disketten in 8250 Laufwerken

Obgleich die 8050 und 8250 Disk-Einheiten eingeschränkt schreib-/lese-kompatibel sind, bewirkt der erste Zugriff zu einer 8050 Diskette, eingesetzt in ein 8250 Laufwerk, die Fehlermeldung 66 ILLEGAL TRACK OR SECTOR. Die Meldung tritt wegen der unterschiedlichen BAM-Inhalte der beiden Disk-Systeme auf und kann ignoriert werden. Die Fehlermeldung tritt meist nur einmal auf und alle weiteren Disk-Befehle werden korrekt ausgeführt.
Um die Benutzung zu erleichtern, sollten die Daten der 8050 Disketten auf 8250-formatierte Disketten mittels des "copy"-Befehls übertragen werden.
Relative Dateien lassen sich mit dem Programm "EXPAND.REL" übertragen.
Mit dem "backup"-Befehl kann nicht gearbeitet werden.

Die 8050 Disk-Einheit ist, mit einigen Ausnahmen, aufwärts kompatibel zur 8250. Die 8050 Disk-Einheit kann nicht auf die Rückseite (oben) einer 8250-formatierten Diskette zugreifen. Auf relative Dateien, die auf einer 8250 Disk-Einheit erstellt wurden, kann von einer 8050 nicht zugegriffen werden, sofern nicht die "Expanded Relative File"-Einrichtung vor dem Erstellen der Datei ausgeschaltet wurde, es sei denn, die komplette Datei befindet sich auf der Vorderseite (unten) der Diskette.

Verwaltung relativer Dateien auf der 8250 Disk-Einheit

Auf der 8050 Disk-Einheit sind relative Dateien auf 182.880 Bytes begrenzt. Bei der 8250 Disk-Einheit mit DOS 2.7 gibt es diese Begrenzung nicht mehr, und relative Dateien können die gesamte Diskette belegen. Normalerweise ist bei der 8250 die "Expanded Relative File"-Einrichtung (für erweiterte relative Dateien) freigegeben. Zum Lesen oder Schreiben 8050-formatierter relativer Dateien muß diese Einrichtung wie folgt gesperrt werden:

```
open 15,8,15
print#15,"m-w"chr$(164)chr$(67)chr$(1)chr$(255)
close 15
```

Hierdurch wird der Zugriff auf erweiterte relative Dateien gesperrt, bis die 8250 aus- und eingeschaltet wird, durch "user"-Befehl (u: oder uj) zurückgesetzt wird oder die "Expanded Relative File"-Einrichtung wieder wie folgt freigegeben wird:

```
open 15,8,15
print#15,"m-w"chr$(164)chr$(67)chr$(1)chr$(0)
close 15
```

Bestehende relative Dateien im 8050 Format können in das 8250 "Expanded Relative File"-Format mittels des "EXPAND.REL"-Programms, enthalten auf der TEST/DEMO-Diskette, umgewandelt werden. Zur Umwandlung muß das Programm mit der 8250 Disk-Einheit geladen ("dload") und gestartet werden. Entsprechende Instruktionen werden auf dem Bildschirm angezeigt. Auf die durch dieses Programm erweiterten relativen Dateien kann über die 8050 nicht mehr zugegriffen werden.

Teil 6: Disk Speicherformate

Block-Verteilung auf den Spuren

Disk-Einheit	Spur-Nummer	Anzahl Blöcke
2031	1 - 17	21
	18 - 24	19
	25 - 30	18
	31 - 35	17
4040	1 - 17	21
	18 - 24	19
	25 - 30	18
	31 - 35	17
8050	1 - 39	29
	40 - 53	27
	54 - 64	25
	65 - 77	23
8250	1 - 39	29
	40 - 53	27
	54 - 64	25
	65 - 77	23
	78-116	29
	117-130	27
	131-141	25
	142-154	23
D9060/D9090:	153 Spuren pro Aufzeichnungs-Oberfläche (4 auf der D9060 und 6 auf der D9090) mit 32 Sektoren pro Spur.	

BAM (Block Allocation Map) - Formate

2031 BAM-Format auf Spur 18, Sektor 00

Byte	Daten	Definition
0-1	18-00	Spur - Sektor des 1. Blocks vom Inhaltsverzeichnis
2	65	ASCII "a" identifiziert DOS 2.6-Format
3	00	Reserviert für künftige DOS-Benutzung
4-143		Bit-Karte der verfügbaren Blöcke, Spuren 1-35

4040 BAM-Format auf Spur 18, Sektor 00

Byte	Daten	Definition
0-1	18-00	Spur - Sektor des 1. Blocks vom Inhaltsverzeichnis
2	65	ASCII "a" identifziert DOS 2.1-Format
3	00	Reserviert für künftige DOS-Benutzung
4-143		Bit-Karte der verfügbaren Blöcke, Spuren 1-35

8050 BAM-Format (1.Block) auf Spur 38, Sektor 00

Byte	Daten	Definition
0-1	38-03	Spursektor vom 2. BAM-Block
2	67	ASCII "c" identifiziert DOS 2.5-Format
3	00	Reserviert für künftige DOS-Benutzung
4	01	Niedrigste Spur-Nummer eingetragen im 1. BAM-Block
5	51	Höchste Spur-Nummer (+1) eingetragen im 1. BAM-Block
6		Anzahl unbenutzter Blöcke auf Spur 1
7-10		Bit-Karte der verfügbaren Blöcke auf Spur 1
11-255		Bit-Karte der verfügbaren Blöcke, Spuren 2-50

8050 BAM-Format (2.Block) auf Spur 38, Sektor 03

Byte	Daten	Definition
0-1	39-01	Spursektor vom 1. Block des Inhaltsverzeichnisses
2	67	ASCII "c" identifiziert DOS 2.5-Format
3	00	Reserviert für künftige DOS-Benutzung
4	51	Niedrigste Spur-Nummer eingetragen im 2. BAM-Block
5	78	Höchste Spur-Nummer (+1) eingetragen im 2. BAM-Block
6		Anzahl unbenutzter Blöcke auf Spur 51
7-10		Bit-Karte der verfügbaren Blöcke auf Spur 51
11-140		Bit-Karte der verfügbaren Blöcke, Spuren 52-77

8250 BAM-Format (1. Block) auf Spur 38, Sektor 00

Byte	Daten	Definition
0-1	38-03	Spursektor vom 2. BAM-Block
2	67	ASCII "c" identifiziert DOS 2.7-Format
3	00	Reserviert für künftige DOS-Benutzung
4	01	Niedrigste Spur-Nummer eingetragen im 1. BAM-Block
5	51	Höchste Spur-Nummer (+1) eingetragen im 1. BAM-Block
6		Anzahl unbenutzter Blöcke auf Spur 1
7-10		Bit-Karte der verfügbaren Blöcke auf Spur 1
11-255		Bit-Karte der verfügbaren Blöcke, Spur 2-50

8250 BAM-Format (2. Block) auf Spur 38, Sektor 03

Byte	Daten	Definition
0-1	38-06	Spursektor vom 3. BAM-Block
2	67	ASCII "c" identifiziert DOS 2.7-Format
3	00	Reserviert für künftige DOS-Benutzung
4	51	Niedrigste Spur-Nummer eingetragen im 2. BAM-Block
5	101	Höchste Spur-Nummer (+1) eingetragen im 2. BAM-Block
6		Anzahl unbenutzter Blöcke auf Spur 51
7-10		Bit-Karte der verfügbaren Blöcke auf Spur 51
11-255		Bit-Karte der verfügbaren Blöcke, Spur 52-100

8250 BAM-Format (3. Block) auf Spur 38, Sektor 06

Byte	Daten	Definition
0-1	38-09	Spursektor vom 4. BAM-Block
2	67	ASCII "c" identifiziert DOS 2.7-Format
3	00	Reserviert für künftige DOS-Benutzung
4	101	Niedrigste Spur-Nummer eingetragen im 3. BAM-Block
5	151	Höchste Spur-Nummer (+1) eingetragen im 3. BAM-Block
6		Anzahl unbenutzter Blöcke auf Spur 101
7-10		Bit-Karte der verfügbaren Blöcke auf Spur 101
11-255		Bit-Karte der verfügbaren Blöcke, Spur 102-150

8250 BAM-Format (4. Block) auf Spur 38, Sektor 09

Byte	Daten	Definition
0-1	39-01	Spursektor vom 1. Block des Inhaltsverzeichnisses
2	67	ASCII "c" identifiziert DOS 2.7-Format
3	00	Reserviert für künftige DOS-Benutzung
4	151	Niedrigste Spur-Nummer eingetragen im 4. BAM-Block
5	155	Höchste Spur-Nummer (+1) eingetragen im 4. BAM-Block
6		Anzahl unbenutzter Blöcke auf Spur 151
7-10		Bit-Karte der verfügbaren Blöcke auf Spur 151
11-30		Bit-Karte der verfügbaren Blöcke, Spur 152-154

D9060/D9090 BAM-Block-Format auf Spur 1, Sektor 0 (normale Lokation)

Byte	Bedeutung
0-1	Spur-Sektor-Zeiger zum nächsten BAM-Block (FFFF hex. = letzter BAM-Block)
2-3	Spur-Sektor-Zeiger zu vorherigem BAM-Block (FFFF hex. = erster BAM-Block)
4	Niedrigste Spur-Nummer eingetragen in diesem BAM-Block
5	Höchste Spur-Nummer (+1) eingetragen in diesem BAM-Block
6	Anzahl unbenutzter Blöcke auf dieser Spur
7-10	Bit-Karte der verfügbaren Blöcke auf dieser Spur
11-255	Bit-Karte der nächsten 49 Spuren

Struktur der BAM-Einträge für eine Spur - Alle DOS-Versionen

Jede Spur hat fünf Bytes zur Erstellung ihrer Bit-Karte zugeordnet. Ist ein Karten-Bit = 1, bedeutet das, daß der entsprechende Block verfügbar ist. Belegte Blöcke sind durch Karten-Bit = 0 gekennzeichnet. Die Bit-Zuordnung ist so festgelegt, daß die in der Wertigkeit höchsten Bits innerhalb der Karten-Bytes den Blöcken mit den niedrigsten Nummern zugeordnet sind.

Byte	Bedeutung
1	Gegenwärtige Anzahl der verfügbaren Blöcke für diese Spur
2	Bit-Karten Blöcke 0-7. Bit 7 = Block 0, Bit 0 = Block 7
3	Bit-Karten Blöcke 8-15. Bit 7 = Block 8, Bit 0 = Block 15
4	Bit-Karten Blöcke 16-23. Bit 7 = Block 16, Bit 0 = Block 23
5 *	Bit-Karten Blöcke 24-31. Bit 7 = Block 24, Bit 0 = Block 31

* Nicht bei 2031 und 4040.

Directory Header Formate (Disketten-Kennung)

2031 Directory Header auf Spur 18, Sektor 00

Byte	Daten	Definition
1-143		Reserviert für 2031 BAM
144-161		Disketten-Name, aufgefüllt mit Leerzeichen (Shift)
162-163		Diskette ID-Nummer
164	160	Leerzeichen (Shift)
165-166	50, 65	ASCII "2a" identifiziert DOS-Version und Format
167-170	160	Leerzeichen (Shift)
171-255	00	Nicht benutzt

4040 Directory Header auf Spur 18, Sektor 00

Byte	Daten	Definition
1-143		Reserviert für 4040 BAM
144-161		Disketten-Name, aufgefüllt mit Leerzeichen (Shift)
162-163		Diskette ID-Nummer
164	160	Leerzeichen (Shift)
165-166	50, 65	ASCII "2a" identifiziert DOS-Version und Format
167-170	160	Leerzeichen (Shift)
171-255	00	Nicht benutzt

8050 Directory Header auf Spur 39, Sektor 00

Byte	Daten	Definition
0-1	38-00	Spur-Sektor-Zeiger zum ersten BAM-Block
2	67	ASCII "c" identifiziert DOS 2.5-Format
3	00	Reserviert für künftige DOS-Benutzung
4-5		Nicht benutzt
6-21		Disketten-Name, aufgefüllt mit Leerzeichen (Shift)
22-23	160	Leerzeichen (Shift)
24-25		Diskette ID-Nummer
26	160	Leerzeichen (Shift)
27-28	50, 67	ASCII "2c"identifiziert DOS-Version und Format
29-32	160	Leerzeichen (Shift)
33-255	00	Nicht benutzt

8250 Directory Header auf Spur 39, Sektor 00

Byte	Daten	Definition
0-1	38-00	Spur-Sektor-Zeiger zum ersten BAM-Block
2	67	ASCII "c" identifiziert DOS 2.7-Format
3	00	Reserviert für künftige DOS-Benutzung
4-5		Nicht benutzt
6-21		Disketten-Name, aufgefüllt mit Leerzeichen (Shift)
22-23	160	Leerzeichen (Shift)
24-25		Diskette ID-Nummer
26	160	Leerzeichen (Shift)
27-28	50, 67	ASCII "2c"identifiziert DOS-Version und Format
29-32	160	Leerzeichen (Shift)
33-255	00	Nicht benutzt

D9060/D9090 Directory Header auf Spur 0, Sektor 0

Byte	Daten	Definition
0-1		Spur-Sektor-Zeiger zu Fehler-Spur-/Sektorliste
2-3	00-255	Identifiziert DOS 3.0-Format
4-5	76-00	Spur-Sektor des 1. Blocks vom Inhaltsverzeichnis
6-7	01-00	Spur-Sektor des 1. BAM-Blocks

Inhaltsverzeichnis Block-Formate - Alle DOS-Versionen

2031 Inhaltsverzeichnis-Blöcke - Spur 18, Sektor 01 bis 18

4040 Inhaltsverzeichnis-Blöcke - Spur 18, Sektor 01 bis 18

8050 Inhaltsverzeichnis-Blöcke - Spur 39, Sektor 01 bis 29

8250 Inhaltsverzeichnis-Blöcke - Spur 39, Sektor 01 bis 29

D9060/D9090 Inhaltsverzeichnis-Blöcke beginnen auf Zylinder 76 (alle Spuren), Sektoren 00 bis 31. Dann werden zur Erweiterung, sofern benötigt, weitere Blöcke benutzt, um ein unbegrenztes Inhaltsverzeichnis zu erhalten.

Byte	Definition
0-1	Spur-Sektor-Zeiger zum nächsten Inhaltsverzeichnis-Block
2	Datei-Typ
3-4	Spur-Sektor-Zeiger zum ersten Datei-Block
5-20	Dateiname, aufgefüllt mit Leerzeichen (Shift)
21-22	Spur-Sektor des 1. Seitensektors bei relativer Datei
23	Satzlänge bei relativer Datei
24-27	Reserviert für künftige Datei-Information
28-29	Spur-Sektor-Zeiger für Ersatz
30-31	Anzahl der von der Datei belegten Blöcke
32-255	Weitere 7 Datei-Einträge (wie Bytes 2-31 + 2 zusätzliche unbenutzte Bytes)

Anmerkung zu Inhaltsverzeichnis Block-Formaten - alle DOS-Versionen:

1. 32 Bytes pro Datei-Eintrag, außer erstem Eintrag (30 Bytes)
2. Insgesamt 8 Datei-Einträge pro Inhaltsverzeichnis-Block
3. Datei-Typen:

Gelöschte Datei	$00
Sequentielle Datei	$01
Programm-Datei	$02
Benutzer-definiert	$03
Relative Datei	$04

4. Bei abgeschlossenen Dateien sind die Datei-Typ-Codes logisch ODER-verknüpft mit $80.
5. Spur-Eintrag von 00 in Byte 0 zeigt den letzten benutzten Block im Inhaltsverzeichnis. Der Sektor-Eintrag zeigt das nächste zu benutzende Byte.

Programm-Dateien

Byte	Definition
0-1	Spur-Sektor-Zeiger zum nächsten Programm-Block
2-255	Bis zu 254 Bytes des BASIC Programmtextes. Das Ende der Datei wird durch drei aufeinanderfolgende Bytes mit $00 markiert.

Sequentielle und relative Dateien

Byte	Definition
0-1	Spur-Sektor-Zeiger zum nächsten sequentiellen Daten-Block
2-255	Bis zu 254 Datenbytes mit "Carriage Return" als Begrenzung zwischen den Datenabschnitten.

Bemerkung: Spur-Verkettung von $00 in Byte 0 zeigt den letzten Datenblock an. Die Sektor-Verkettung ist dann die nächste Byte-Position, um Daten zu empfangen. Das Ende relativer Dateien wird durch das Lesen von $ff angezeigt.

Seitensektor-Format bei relativen Dateien

Byte	Definition
0-1	Spur-Sektor-Zeiger zum nächsten Seitensektor
2	Seitensektor-Anzahl
3	Relative Satzlänge
4-5	Spur-Sektor-Zeiger - 1. Seitensektor
6-7	Spur-Sektor-Zeiger - 2. Seitensektor
8-9	Spur-Sektor-Zeiger - 3. Seitensektor
10-11	Spur-Sektor-Zeiger - 4. Seitensektor
12-13	Spur-Sektor-Zeiger - 5. Seitensektor
14-15	Spur-Sektor-Zeiger - 6. Seitensektor
16-255	Spur-Sektor-Zeiger zu 120 Daten-Blöcken Insgesamt 720 Blöcke (max. 182.8 K Bytes) pro Datei

DOS 2.7 und DOS 3.0 Super-Seitensektor enthält Spur/Sektor-Zeiger zu 127 Gruppen von 6 Seitensektoren wie oben. Die maximale Dateigröße ist 23,25 M Bytes.

Teil 7: DOS-Fehlermeldungen

Abfragen der Fehlermeldungen

Die Ausführung des folgenden Programms zeigt die Fehlermeldungen auf dem Bildschirm und setzt den Fehlerindikator des Gerätes zurück:

BASIC 3.0

```
open 1,8,15
input#1,a,b$,c,d
print a,b$,c,d

input#1,a,b$,c,d,e      (Nur benutzt mit 8250, DOS 2.7)
print a,b$,c,d,e
```

BASIC 4.0

```
print ds$
```

a = Fehlernummer, b$ = Fehlermeldung, c = Spur, d = Sektor,
e = Laufwerknummer.
Fehlermeldungen, die von einer 8250 abgefragt wurden, enthalten die Laufwerknummer als fünfte Variable. "print ds$" beim BASIC 4.0 gibt automatisch die Laufwerknummer aus.

Zusammenfassung der CBM Disk-Fehlermeldungen

0	Kein Fehler vorhanden.
1	Antwort auf Dateilöschung ("scratch"). Kein Fehlerzustand.
2-19	Nicht benutzte Fehlermeldungen, sollten ignoriert werden.
20	"Block Header" auf der Disk nicht gefunden.
21	Synchronisationszeichen nicht gefunden.
22	Datenblock nicht vorhanden.
23	Prüfsummenfehler in den Daten.
24	Byte Decodierungsfehler.
25	Schreibüberprüfungsfehler.
27	Prüfsummenfehler im "Header" (Vorsatz).
28	Schreibfehler.
29	Diskette nicht initialisiert.
30	Allgemeiner Syntaxfehler.
31	Ungültiger Befehl.
32	Befehl zu lang.
33	Ungültiger Dateiname.
34	Keine Datei angegeben.
39	Befehlsdatei nicht gefunden.
50	Satz nicht vorhanden.
51	Überlauf im Satz.
52	Datei zu groß.
60	Datei zum Schreiben eröffnet.
61	Datei nicht eröffnet.
62	Datei nicht gefunden.
63	Datei besteht bereits.
64	Datei-Typ-Diskrepanz.
65	Kein Block.
66	Falsche Spur oder falscher Sektor.
67	Falsche Systemspur oder Sektor.
70	Keine Kanäle verfügbar.
71	Fehler im Inhaltsverzeichnis.
72	Disk voll beschrieben oder Inhaltsverzeichnis voll.
73	Einschaltmeldung, oder Schreibversuch mit DOS-Diskrepanz.
74	Laufwerk nicht bereit.
75	Format Geschwindigkeits-Fehler.
76	Controller-Fehler.

Beschreibung der DOS-Fehlermeldungen

Bemerkung: Fehlernummern kleiner als 20 sollten ignoriert werden, außer Fehlernummer 01, welche Informationen über die Anzahl der mit "scratch" gelöschten Dateien gibt.

20: READ ERROR (Block-Header nicht gefunden)
Der Disk-Controller kann den Header (Vorsatz) eines Datenblocks nicht finden. Ursache ist entweder eine illegale Sektor-Nummer oder der Block-Header wurde zerstört.

21: READ ERROR (Laufwerk nicht bereit)
Zeigt einen Hardware-Fehler an, zerstörter Header.

22: READ ERROR (Datenblock nicht vorhanden)
Der Disk-Controller wurde aufgefordert, einen Datenblock zu lesen, der nicht richtig beschrieben wurde. Diese Fehlermeldung tritt in Verbindung mit den "block"-Befehlen auf und zeigt eine falsche Spur- oder Sektor-Anfrage an.

23: READ ERROR (Prüfsummenfehler im Datenblock)
Diese Fehlermeldung zeigt an, daß ein Fehler in einem oder mehreren Datenbytes vorhanden ist. Die Daten wurden in den DOS-Speicher gelesen, aber die Datenprüfsumme ist fehlerhaft. Diese Meldung kann auch auf Erdungsprobleme hindeuten.

24: READ ERROR (Byte Dekodierfehler)
Ein Hardware-Fehler verursachte ein ungültiges Bitmuster im Datenbyte. Diese Meldung kann auch auf Erdungsprobleme hindeuten.

25: WRITE ERROR (Schreib-Überprüfungsfehler)
Diese Meldung wird, erzeugt wenn der Controller einen Unterschied zwischen den geschriebenen Daten und den Daten im DOS-Speicher feststellt.

27: READ ERROR (Prüfsummenfehler im Datenblock-Header)
Der Controller hat einen Fehler im Header (Vorspann) des Datenblocks entdeckt. Der Block wurde nicht in den DOS-Seicher gelesen.
Diese Meldung kann auch auf Erdungsprobleme hindeuten.

28: WRITE ERROR (Zu langer Datenblock)
Der Controller versucht, die Synchronisationsmarke des nächsten Vorspanns zu finden, nachdem ein Datenblock geschrieben wurde. Dieser Fehler wird durch ein falsches Diskettenformat (die Daten reichen bis in den nächsten Block) oder durch einen Hardwarefehler verursacht.

29: DISK ID MISMATCH
Diese Meldung wird erzeugt, wenn der Controller veranlaßt wird, eine nicht initialisierte Diskette anzusprechen. Der Fehler kann ebenfalls auftreten, wenn die Diskette einen beschädigten Header hat.

30: SYNTAX ERROR (allgemeine Syntax)
Das DOS kann den zum Befehlskanal geschickten Befehl nicht interpretieren. Häufig verursacht durch Angabe von falschen Dateinamen oder falsch eingesetzten Namensabkürzungen.

31: SYNTAX ERROR (ungültiger Befehl)
Das DOS erkennt den Befehl nicht. Der Befehl muß in der ersten Position beginnen.

32: SYNTAX ERROR (Befehl zu lang)
Der gesendete Befehl ist länger als 58 Zeichen.

33: SYNTAX ERROR (ungültiger Datei-Name)
Namensabkürzung wurde beim "dopen"- oder "dsave"-Befehl falsch eingesetzt.

34: SYNTAX ERROR (keine Datei angegeben)
Der Dateiname wurde im Befehl weggelassen, oder das DOS erkennt ihn nicht als solchen. Häufiger Fehler: Ein ":" wurde im Befehl vergessen.

39: SYNTAX ERROR (ungültiger Befehl)
Dieser Fehler kann auftreten, wenn ein zum Befehlskanal (Sekundär-Adresse 15) geschickter Befehl für das DOS nicht erkennbar ist.

50: RECORD NOT PRESENT
Die Meldung entsteht beim Lesen auf der Disk nach dem letzten Satz mit "input#"- oder "get"-Befehlen. Diese Fehlermeldung erscheint auch, wenn der Satzzeiger einer rel. Datei hinter den letzten Satz gesetzt wird. Ist beabsichtigt, die Datei durch Hinzufügen eines neuen Satzes zu erweitern (mit einem "print#"-Befehl), kann diese Meldung ignoriert werden. "input" oder "get" Befehle sollte man nach diesem Fehler nicht eingeben, bevor nicht zu einem gültigen Satz positioniert wurde.

51: OVERFLOW IN RECORD
Die mit dem "print"-Befehl geschriebenen Daten überschreiten die definierte relative Satzlänge. Die Daten werden nur bis zur definierten Länge übertragen.
Typische Ursache ist, daß Carriage Returns oder Satzbegrenzer bei der Errechnung der Satzlänge nicht berücksichtigt werden.

52: FILE TOO LARGE
Die Satzposition innerhalb einer relativen Datei zeigt an, daß nicht genügend Blöcke auf der Diskette verfügbar sind, um die angegebene Anzahl von Sätzen zu erhalten.

60: WRITE FILE OPEN
Diese Meldung wird erzeugt, wenn eine Schreib-Datei, die nicht abgeschlossen wurde, zum Lesen eröffnet werden soll.

61: FILE NOT OPEN
Diese Meldung wird erzeugt, wenn auf eine Datei zugegriffen werden soll, die im DOS nicht eröffnet wurde. Manchmal wird in diesem Fall keine Fehlermeldung erzeugt. Der Befehl wird dann einfach ignoriert.

62: FILE NOT FOUND
Die angeforderte Datei befindet sich nicht auf der angegebenen Disk.

63: FILE EXISTS
Der Dateiname der Datei, die erzeugt werden soll, existiert bereits auf der Diskette.

64: FILE TYPE MISMATCH
Der Datei-Typ im "dopen"-Befehl stimmt nicht mit dem Datei-Typ im Eintrag des Inhaltsverzeichnisses überein. Die Datei auf der Disk bleibt unverändert erhalten.

65: NO BLOCK
Diese Meldung tritt in Verbindung mit einem "b-a"-Befehl auf. Sie zeigt an, daß der Block, der zugewiesen ("block-allocate") werden soll, bereits zugewiesen ist. Die Parameter zeigen die nächst höhere verfügbare Spur- und Sektor-Nummer. Sind die Parameter alle "Null", sind alle höher numerierten Blöcke belegt.

66: ILLEGAL TRACK AND SECTOR
Das DOS hat versucht, auf eine Spur oder einen Sektor zuzugreifen, der in dem Format, das benutzt werden soll, nicht existiert.

67: ILLEGAL SYSTEM T OR S
Spezielle Fehlermeldung zum Anzeigen einer illegalen Spur oder eines illegalen Sektors.

70: NO CHANNEL (kein Kanal verfügbar)
Der angeforderte Kanal ist nicht verfügbar, oder alle Kanäle werden benutzt. Maximal 5 sequentielle Dateien oder 3 relative Dateien können gleichzeitig für das DOS eröffnet sein. Direktzugriff-Kanäle können 6 eröffnete Dateien haben.

71: DIRECTORY ERROR
Die BAM stimmt nicht mit dem internen Zähler überein. Das Problem liegt in der BAM-Zuordnung, oder die BAM wurde im DOS-Speicher überschrieben. Zur Korrektur dieses Fehlers muß man die Disk neu initialisieren, was ein neues Speichern der BAM im DOS-Speicher bewirkt. Aktive Dateien können durch diese Aktion beendet werden.

72: DISK FULL
Entweder sind alle Blöcke auf der Diskette belegt, oder das Inhaltsverzeichnis ist vollgeschrieben. DISK FULL wird gemeldet, wenn noch zwei Blöcke zu Verfügung stehen, um die derzeitige Datei noch abzuschließen.

74: DRIVE NOT READY
Es wurde versucht, auf eine ungülige Gerätenummer zuzugreifen, oder die Disk ist nicht eingeschaltet oder hat noch nicht die Umdrehungsgeschwindigkeit.

75: FORMAT SPEED ERROR
Während des Formatierens überprüft die 8250, ob die Umdrehungsgeschwindigkeit innerhalb von 2 Millisekunden Abweichung von 200 Millisekunden pro Umdrehung (1%) liegt. Ist die Geschwindigkeit außerhalb dieses Bereichs, wird die Formatierung angehalten und die Disk-Fehlerleuchte leuchtet auf.

76: CONTROLLER ERROR
Zeigt Controller-Hardwarefehler an.

Teil 8: Anhang

Gegenüberstellung der Disk-Befehle

Die Gegenüberstellung der Disk-Befehle soll die Unterschiede zwischen den BASIC 3.0 und BASIC 4.0 -Befehlen darstellen. Die BASIC 3.0 -Befehle sind aufwärts kompatibel zu BASIC 4.0. Das bedeutet, daß alle BASIC 3.0 -Befehle auch an Rechnern funktionieren, die mit BASIC 4.0 ausgerüstet sind.
Unter Einsatz des Hilfsprogramms "UNIVERSAL WEDGE" kann anstelle der BASIC 3.0 -Befehlssyntax das hinter U.W. gezeigte Format benutzt werden.

BASIC 3.0	**BASIC 4.0**
UW: UNIVERSAL WEDGE	
save" x : Datei-Name ", y	dsave d x onu y ," Datei-Name "
UW: save x : Datei-Name , y	
load" x : Datei-Name ", y	dload d x onu y ," Datei-Name "
UW: / x : Datei-Name	
load"*", y : list	dload"*"
UW: x : Datei-Name	
load"$", y (Löscht Speicher)	directory (Speicher bleibt erhalten)
UW: $ (Speicher bleibt erhalten)	
10 open1,8,15 20 input#1,a,b$,c,d 30 print a,b$,c,d	?ds$ oder ?ds (ds = Fehlernummer) (ds$ = Fehlermeldung)
UW: return	

Für die folgenden Formate wird angenommen, daß vor allen "print#"-Befehlen ein "open1,8,15" geschrieben wird.

BASIC 3.0 **BASIC 4.0**

UW: UNIVERSAL WEDGE

Formatieren einer Diskette:

```
print#1,"n x : Disk.-Name , ID "     header" Disk.-Name "
                                          ,d x ,i ID   onu y
UW:   n x : Disk.-Name , ID
```

Initialisieren:

```
print#1,"i x "                       print#1,"i x "

UW:   i x
```

Gültigkeitsüberprüfung:

```
print#1,"v x "                       collect d x   onu y

UW:   v x
```

Kopieren (alle Dateien):

```
print#1,"c x2 = x1 "                 copy d x1  to d x2   onu y

UW:   c x2 = x1
```

Kopieren (einzelne Datei):

```
print#1,"c x2 : Dateiname 2          copy d x1 ," Dateiname 1 "
       = x1 : Dateiname 1 "              to d x2 ," Dateiname 2 "  onu y

UW:   c x2 : Dateiname 2
       = x1 : Dateiname 1
```

Anhängen von Dateien:

```
print#1,"c x2 : Dateiname 2          concat d x1 ," Dateiname 1 "
       = x1.1 : Dateiname 1.1            to d x2 ," Dateiname 2 "  onu y
       , x1.2 : Dateiname 1.2  ..."

UW:   c x2 : Dateiname 2
       = x1.1 : Dateiname 1.1
       , x1.2 : Dateiname 1.2  ...
```

Umbenennen von Dateien:

```
print#1,"r x : Dateiname neu         rename d x ," Dateiname alt "
       = Dateiname alt "                 to " Dateiname neu "  onu y

UW:   r x : Dateiname neu
       = Dateiname alt
```

Löschen von Dateien:

```
print#1,"s x : Dateiname "           scratch" Dateiname ",d x   onu y

      s x : Dateiname
```

Hardware-Änderung der Gerätenummer

Alle CBM Disk-Einheiten haben normalerweise die Gerätenummer 8. Diese kann über einen "m-w"-Befehl geändert werden, wird aber nach jedem Neueinschalten wieder 8 sein. Durch eine Änderung auf der Leiterplatte kann die neue Gerätenummer permanent festgelegt werden. Diese Hardware-Änderung ist bei den verschiedenen Disk-Modellen unterschiedlich.

Warnung: Diese Hardware-Änderung darf nur von qualifizierten CBM-Technikern durchgeführt werden. Änderungen von Unbefugten haben den Garantieverlust für das Gerät zur Folge!

Änderung 2031

Zwei Dioden steuern die Gerätenummer bei der 2031. Die Dioden liegen in der Nähe von I.C.-Chip U3J auf der Digital-Leiterplatte. Zum Ändern der Gerätenummer ist eine der Leitungen an einer oder beiden Dioden durchzutrennen:

Gerätenummer	CR18	CR19	
8	0	0	
9	0	1	(0 = unverändert)
10	1	0	(1 = Leitung durchgetrennt)
11	1	1	

Änderung 4040/8050/8250

Drei Pins (22, 23, 24) auf I.C.-Chip UE1 (auf der Digital-Leiterplatte) steuern die Gerätenummer bei diesen Einheiten. Diese Pins sind normalerweise auf Ground gelegt. Drei kleine runde Blöcke erkennt man direkt links von UE1. Pin 22 ist verbunden mit dem obersten dieser Blöcke (von der Vorderseite der Disk-Einheit her gesehen). Zum Ändern der Gerätenummer ist UE1 herauszunehmen. Dann müssen die entsprechenden Pins so herausgebogen werden, daß diese nach dem Wiedereinsetzen von UE1 keine Verbindung mehr haben.

Gerätenummer	Pin 22	Pin 23	Pin 24	
8	0	0	0	
9	0	0	1	(0 = unverändert)
10	0	1	0	(1 = Pin herausgebogen)
11	0	1	1	
12	1	0	0	
13	1	0	1	
14	1	1	0	
15	1	1	1	

Änderung D9060/D9090

Drei Pins (22, 23, 24) auf I.C.-Chip 7G (auf der obersten Leiterplatte) steuern die Gerätenummer dieser Einheiten. Diese Pins sind normalerweise auf Ground gelegt. Zum Ändern der Gerätenummer sind die entsprechenden Pins so herauszubiegen, daß sie keine Verbindung mehr haben.

Gerätenummer	Pin 22	Pin 23	Pin 24	
8	0	0	0	
9	0	0	1	(0 = unverändert)
10	0	1	0	(1 = herausgebogen)
11	0	1	1	
12	1	0	0	
13	1	0	1	
14	1	1	0	
15	1	1	1	

Commodore GmbH
Lyoner Straße 38
D-6000 Frankfurt/M. 71

Commodore AG
Aeschenvorstadt 57
CH-4010 Basel

Commodore GmbH
Fleschgasse 2
A-1130 Wien

Artikel-Nr. 558060