COMMODORE

DISK DRIVE
1570/71

Bedienungshandbuch

DISK DRIVE *1570/71*

Alle in diesem Handbuch gegebenen Informationen wurden überprüft und sind daher zuverlässig. Für eventuelle sachliche Fehler kann jedoch keinerlei Verantwortung übernommen werden. Die Bedienungsanleitung dient nur Ihrer Information. Technische Änderungen jederzeit vorbehalten.

Commodore Büromaschinen GmbH 1985

DISK DRIVE 1570/71

Bedienungshandbuch

Commodore

INHALT

KAPITEL 1

ALLGEMEINE HINWEISE

1.1 HAUPTEIGENSCHAFTEN

Die 1570/71 ist ein vielseitiges Diskettenlaufwerk, das mehrere Diskettenformate und Datenübertragungsraten verarbeiten kann. Die Diskettenformate gehen von einseitigen Disketten mit einfacher Schreibdichte zu doppelseitigen Disketten mit doppelter Schreibdichte. Die 1570/71 kann mit einer Vielzahl von Computern verwendet werden, einschließlich dem Commodore 128 Personal Computer, C 64, PLUS/4, C 16, C 116 und VC 20.
Wird das Diskettenlaufwerk 1570/71 mit dem Commodore 128 Personal Computer benutzt, so bietet es folgende Möglichkeiten:

- **Schnelle serielle Datenübertragung und Standarddatenübertragungsrate –** die 1570/71 wählt automatisch die richtige Datenübertragungsrate (schnell oder langsam), um den drei Betriebsarten bei dem Commodore 128 Personal Computer (C128-Modus, C64-Modus und CP/M-Modus) gerecht zu werden.
- **Möglichkeit, im MFM-Format mit doppelter Schreibdichte zu lesen und zu schreiben –** Dadurch kann auf CP/M-Softwarebibliotheken anderer Personal Computer zugegriffen werden.
- **Bei der 1571 Datenaufzeichnung auf doppelseitigen Disketten mit doppelter Schreibdichte –** Die 1571 ermöglicht eine Speicherkapazität von 336K pro Diskette (168K pro Seite). Wird das Diskettenlaufwerk 1571 mit dem Commodore 64 benutzt, so unterstützt es die Standarddisketten mit einfacher Schreibdichte im GCR-Format, die mit den Commodore 1541-, 1551-, 1570-, 4040-, und 2031-Diskettenlaufwerken benutzt werden.

1.2 LIEFERHINWEIS

CP/M-Disketten werden mit dem Commodore 128 geliefert. Informationen über den Betrieb des CP/M stehen in den Benutzerhandbüchern des Commodore 128.

1.3 AUFBAU DIESES HANDBUCHS

Diesem Handbuch liegt folgende Einteilung zugrunde:
In den ersten beiden Kapiteln werden im wesentlichen allgemeine Hinweise zum Arbeiten mit der 1570/71 gegeben.

Das nächste Kapitel beschäftigt sich mit den zur Arbeit mit der 1570/71 erforderlichen BASIC-Befehlen.
Ein weiteres Kapitel widmet sich dem DOS SHELL, einem Anwenderprogramm, das zahlreiche Disk Utilities (u. a. Kopieren, Wiederherstellen gelöschter Dateien, Sortieren u. dgl.) enthält.
Die folgenden Kapitel über sequentielle und relative Dateien sowie Direktzugriffsdateien und das Kapitel über Floppy-Systembefehle richten sich hauptsächlich an den erfahrenen Anwender.
Der Anhang ist als Nachschlagewerk anzusehen.

KAPITEL 2

ARBEITEN MIT DER 1570/71

2.1 INBETRIEBNAHME

1. Als erstes prüfen Sie den Versandkarton auf Beschädigungen.
   Wurde der Karton beschädigt und fürchten Sie, daß das Diskettenlaufwerk ebenfalls beschädigt wurde, so nehmen Sie Verbindung mit dem Händler auf.
2. Prüfen Sie den Inhalt des Kartons.
   Außer der 1570/71 und diesem Handbuch muß der Karton folgende Teile enthalten: dreiadriges Netzkabel, Schnittstellenkabel, Test-/Demodiskette und eine Garantiekarte, die von Ihnen ausgefüllt und an Commodore zurückgeschickt werden muß.
3. Nehmen Sie die Transportsicherung aus dem Diskettenlaufwerk heraus.
   Die Sicherung soll die Innenseite des Laufwerks während des Transports schützen. Um sie herauszunehmen, drehen Sie den Hebel auf der Vorderseite des Laufwerks gegen den Uhrzeigersinn (s. Abb. 1) und nehmen die Transportsicherung heraus.

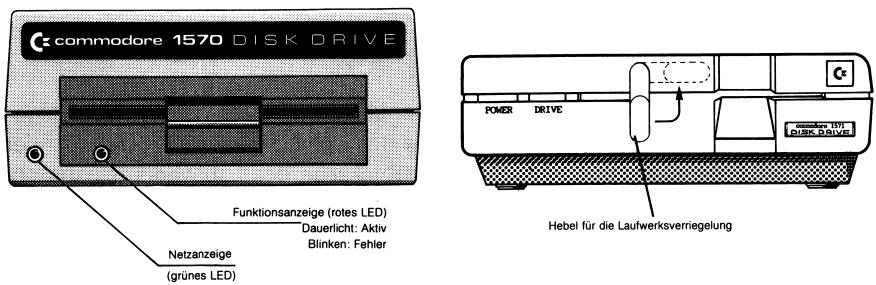
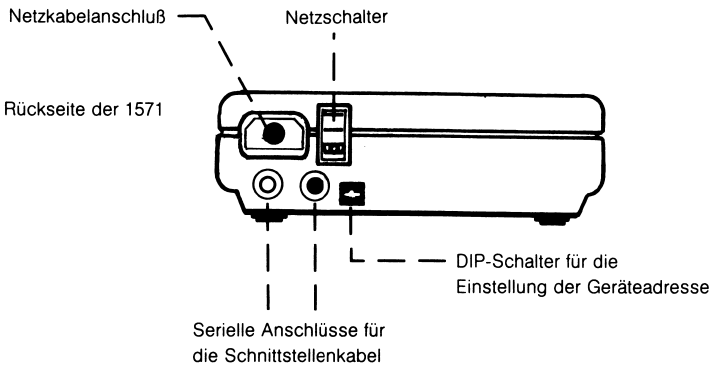


Abbildung 1 Vorderseiten der Commodore 1570/71 Diskettenlaufwerke

4. Schließen Sie das Netzkabel an.
   Prüfen Sie, ob der Netzschalter auf der Rückseite des Laufwerks (s. Abb. 2) auf OFF steht. Schließen Sie das Kabel an der in Abbildung 2 angegebenen Stelle an. Das andere Ende des Kabels schließen Sie an einer geerdeten Steckdose an. Schalten Sie das Laufwerk noch nicht ein.



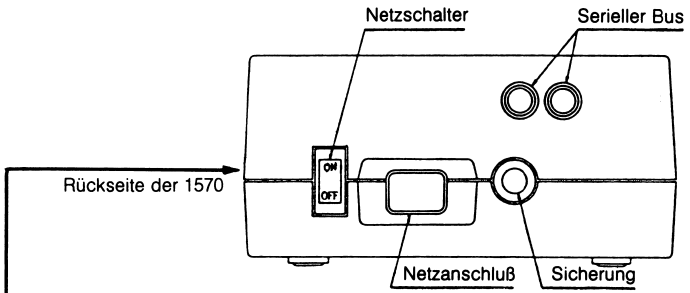


Abbildung 2 Anschluß des Netzkabels und der Schnittstellenkabel an der 1570/71

5. Schließen Sie das Schnittstellenkabel an.
   Vergewissern Sie sich, daß der Computer und andere Peripheriegeräte ausgeschaltet sind. Stecken Sie das eine Ende des Schnittstellenkabels in einen der beiden Schnittstellenanschlüsse auf der Rückseite des Laufwerks (s. Abb. 2). Das andere Ende des Kabels schließen Sie an der Rückseite des Computers an.

- **Betriebssystem 1570**

  Die 1570 arbeitet mit dem Commodore DOS, Version 3.0/1570. Eine entsprechende Meldung erhalten Sie unmittelbar nach dem Einschalten bei der Abfrage des Floppyfehlerkanals.
  ("PRINT DS$" bei BASIC 7.0)

  Die 1570 arbeitet im Gegensatz zur 1571 mit nur einem Schreib-/Lesekopf.

- **Unterschiede zur 1571**

  Die Commodore 1570 Diskettenstation verfügt über ein Laufwerk mit einem Klappenverschluß.

6. Schalten Sie das Laufwerk ein.
   Nachdem alle Anschlüsse vorgenommen wurden und das Laufwerk leer ist, können Sie die Peripheriegeräte in beliebiger Reihenfolge einschalten. Den Computer schalten Sie jedoch als letztes Gerät ein. Nachdem alles eingeschaltet ist, wird ein Selbsttest des Laufwerks durchgeführt. Wird dieser erfolgreich beendet, so leuchtet die grüne Anzeige einmal auf, während die rote Netzanzeige ständig brennt. Blinkt die grüne Anzeige weiter, so liegt unter Umständen ein Problem vor. Hier wird auf die Hinweise zur Fehlerbehebung verwiesen.

● Gerateadresse 1570 siehe Seite 22

2.2 HINWEISE ZUR FEHLERBESEITIGUNG

Problem	Mögliche Ursache	Fehlerbehebende Maßnahme
Rote Netzanzeige leuchtet nicht auf.	Stromzufuhr nicht eingeschaltet.	Vergewissern Sie sich, daß der Netzschalter auf ON steht.
	Netzkabel nicht angeschlossen.	Prüfen Sie, ob beide Enden des Netzkabels richtig eingesteckt sind.
	Kein Strom in der Steckdose.	Kontrollieren Sie die Sicherungen.
Grüne Anzeige blinkt ständig.	Der Selbsttest des Laufwerks wurde nicht erfolgreich ausgeführt.	Schalten Sie das System einen Augenblick aus und wiederholen Sie die Operation. Blinkt die Anzeige noch immer, so schalten Sie das Laufwerk aus, entfernen das Schnittstellenkabel und schalten wieder ein. Bleibt das Problem weiter bestehen, so nehmen Sie Verbindung mit dem Händler auf. Ergab sich ein Unterschied, sobald das Schnittstellenkabel abgenommen war, so

Problem	Mögliche Ursache	Fehlerbehebende Maßnahme
		liegt es wahrscheinlich an dem Kabel selbst oder an anderer Stelle in dem System.
Die Programme können nicht geladen werden, und der Computer gibt folgende Fehlermeldung aus: „DEVICE NOT PRESENT ERROR" (Einheit nicht verfügbar).	Das Schnittstellenkabel ist nicht richtig angeschlossen oder das Laufwerk nicht eingeschaltet.	Vergewissern Sie sich, daß das Kabel richtig angeschlossen und das Laufwerk eingeschaltet ist.
Die Programme können nicht geladen werden, der Computer und das Diskettenlaufwerk geben jedoch keine Fehlermeldung aus.	Die Störung kann von einem anderen Teil des Systems ausgehen.	Trennen Sie alle an den Computer angeschlossenen Geräte ab. Wird das Problem dadurch behoben, so schließen Sie sie nacheinander wieder an. Das Gerät, das als letztes angeschlossen wurde, bevor das Problem erneut auftritt, ist wahrscheinlich die Fehlerursache. Dieses Problem wird dadurch verursacht, daß ein Maschinenprogramm in den BASIC-Bereich geladen wurde.
Die Programme können nicht geladen werden, und die grüne Laufwerksanzeige blinkt.	Diskettenfehler.	Fragen Sie den Fehlerkanal ab, um den Fehler zu bestimmen. Danach beheben Sie den Fehler wie in Anhang 9.2 beschrieben. Der Fehlerkanal wird in Kapitel 3 erläutert.

Problem	Mögliche Ursache	Fehlerbehebende Maßnahme
		(Beim Laden der Programme vergewissern Sie sich, ob Sie die Programmnamen richtig geschrieben und die richtigen Satzzeichen eingefügt haben. Verwenden Sie eine andere Kopie des Programms.)
Ihre eigenen Programme werden einwandfrei geladen, die kommerziellen Programme und die Programme von anderen 1570/71-Laufwerken werden jedoch nicht richtig geladen.	Entweder ist die Diskette fehlerhaft oder das Diskettenlaufwerk ist nicht richtig justiert.	Können verschiedene Programme aus verschiedenen Quellen nicht richtig geladen werden, so muß der Händler das Diskettenlaufwerk neu justieren.
Ihre eigenen Programme, die vorher geladen werden konnten, können nicht mehr geladen werden. Auf neu formatierten Disketten gesicherte Programme werden jedoch geladen.	Ältere Disketten sind beschädigt.	Hier wird auf die Behandlungshinweise für Disketten im nächsten Abschnitt verwiesen. Von den Sicherungsdisketten müssen neue Kopien angefertigt werden.
	Das Diskettenlaufwerk ist nicht richtig justiert.	Der Händler muß das Diskettenlaufwerk neu justieren.

2.3 AUFSTELLUNG UND PFLEGE DER 1570/71

1. Das Laufwerk sollte stets gut belüftet sein.
   Einige Zentimeter Platz gewährleisten die Luftzirkulation auf allen Seiten und verhindern so einen Hitzestau innerhalb des Laufwerks.
2. Benutzen Sie Commodore-Disketten.
   Minderwertige Disketten können zu einem erhöhten Verschleiß bei dem Schreib-/

Lese-Kopf des Laufwerks führen. Wird eine außergewöhnlich laute Diskette benutzt, so könnte diese zu einem zusätzlichen Verschleiß führen. Sie muß ausgetauscht werden.

3. Die Commodore 1570/71 muß bei normaler Benutzung einmal im Jahr gereinigt werden.
   Auf einige Elemente muß besonders geachtet werden: die beiden Schreib-/Lese-Köpfe müssen gereinigt werden (mit 91%igem Isopropylalkohol auf einem Wattestäbchen). Die Schienen, auf denen der Kopf bewegt wird, müssen geschmiert werden (mit einem besonderen Molybden-Schmiermittel, nicht mit Öl). Von dem Schreibschutz-Senor muß Staub entfernt werden. Da für diese Elemente besonderes Material oder besondere Teile erforderlich sind, sollten diese Arbeiten am besten von einem Commodore Vertrags-Servicecenter ausgeführt werden. Möchten Sie die Arbeiten selbst ausführen, so fragen Sie den Händler nach dem entsprechenden Material. WICHTIGER HINWEIS: Wird die Commodore 1570/71 selbst repariert, so erlöschen die Garantieansprüche.

2.4 EINLEGEN EINER DISKETTE

Um eine Diskette einzulegen, öffnen Sie zuerst die Laufwerksöffnung, indem Sie den Hebel um eine Viertelumdrehung bis zum Anschlag entgegen dem Uhrzeigersinn drehen, wobei der Hebel parallel zu der horizontalen Öffnung auf der Vorderseite des Laufwerks liegen muß.
Fassen Sie die Diskette an der Seite gegenüber der ovalen Öffnung an und halten Sie sie mit dem Etikett nach oben und der Schreibschutzkerbe nach links (s. Abb. 3). Nun schieben Sie die Diskette gerade mit dem ovalen Ausschnitt zuerst und dem Etikett zuletzt in die Laufwerksöffnung ein. **Die Diskette muß bis zum Anschlag eingeschoben werden. Sie darf weder gebogen noch mit Gewalt eingeschoben werden.**
HINWEIS: Ist die Schreibschutzkerbe mit einem Streifen abgedeckt, so kann der Inhalt der Diskette nicht geändert oder ergänzt werden. Dadurch wird ein versehentliches Löschen wichtiger Informationen verhindert. Wird eine Diskette ohne Schreibschutzkerbe geliefert, so kann der Inhalt dieser Diskette nicht geändert werden.
Leere Disketten sind beim Erwerb in der Regel nur mit dem Herstelleretikett versehen.
Sobald die Diskette richtig eingeschoben ist, schließen Sie das Laufwerk, indem Sie den Laufwerkshebel um eine Viertelumdrehung im Uhrzeigersinn drehen, bis der Hebel senkrecht über der Öffnung stehen bleibt.
Warnung: Kann der Hebel nur schwer bewegt werden, so halten Sie sofort an. Unter Umständen haben Sie die Diskette nicht richtig oder nicht vollständig eingeschoben. In diesem Fall muß die Diskette erneut eingelegt werden, bis der Laufwerkshebel problemlos gedreht werden kann.

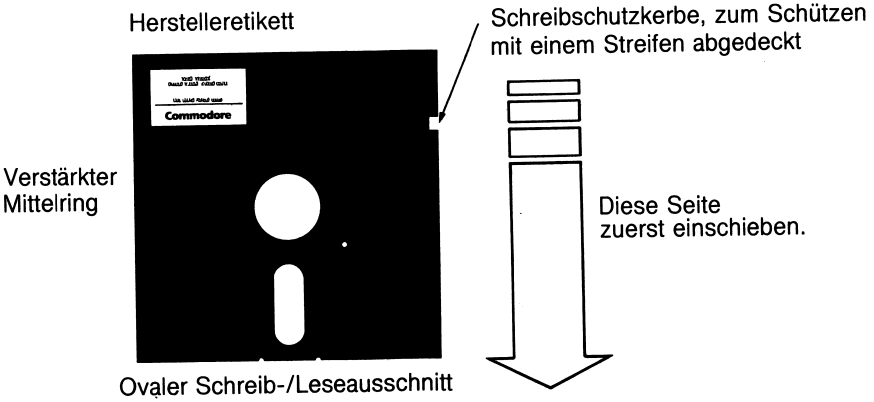


Abbildung 3 Einlegen einer Diskette

2.5 DISKETTENBEHANDLUNG

1. Berühren Sie **niemals** die freiliegenden Teile einer Diskette.
   Dies gilt insbesondere für den Schreib-/Leseausschnitt und die Öffnung in der Mitte.
2. Biegen Sie die Diskette **niemals.**
   Disketten werden zwar als Floppy Disks bezeichnet, dürfen dennoch **nicht gebogen** werden.
3. Die Disketten (und das Diskettenlaufwerk) dürfen **nicht in die Nähe von Magnetfeldern** gebracht werden.
   Dies umfaßt die Elektromagnete in Telefonen, Fernsehgeräten, Schreibtischlampen und Rechnerkabeln.
4. Kaufen Sie Disketten mit verstärktem Mittelring.
   Auch wenn das Laufwerk einer Diskette im allgemeinen richtig zentriert, können Daten von einer Diskette, deren Nabe nicht genau in der Mitte zentriert ist, nur schwer wiederhergestellt werden. Mit verstärktem Mittelring kann das Laufwerk eine Diskette leicht zentrieren.
5. Nehmen Sie die Diskette heraus, bevor Sie das Laufwerk ausschalten, ansonsten können sämtliche oder ein Teil der Daten auf der Diskette verloren gehen.
6. Nehmen Sie eine Diskette nicht aus dem Laufwerk heraus, während die grüne Anzeige aufleuchtet.
   Diese Anzeige leuchtet auf, wenn das Laufwerk benutzt wird. Wird die Diskette bei laufendem Laufwerk herausgenommen, so können die gerade auf die Diskette geschriebenen Daten verloren gehen.
7. Stecken Sie die Diskette, sobald Sie sie aus dem Laufwerk herausnehmen, **immer** in ihre Schutzhülle zurück.

Disketten müssen vor Asche, Feuchtigkeit, Staub und Krümeln geschützt werden. Lagern Sie die Disketten aufrecht in ihren Schutzhüllen.

Bitte beachten Sie alle diese Hinweise in Ihrem Interesse stets auf das Strikteste. Eine Diskette ist ein empfindlicher Datenträger, und ein möglicher Datenverlust kann sehr ärgerlich sein. Fertigen Sie aus diesem Grund von Ihren wichtigen Disketten stets eine Sicherheitskopie (BACKUP) an, die Sie nur im Falle eines Datenverlustes und nur als Kopiervorlage verwenden. Auch die Originaldiskette von käuflicher Software sollte nicht benutzt werden, sondern nach Erstellung einer Arbeitskopie sicher verwahrt werden.

2.6 ARBEITEN MIT FREMDSOFTWARE AUF DISKETTE

In dem Benutzerhandbuch für die Software wird beschrieben, wie das Programm in den Computer geladen wird. Dennoch haben wir nachfolgend einige allgemeine Richtlinien aufgenommen. Mit diesem Verfahren laden Sie auch Programme oder Dateien von Ihren eigenen Disketten. Zur Demonstration benutzen Sie die Test-/ Demo-Diskette, die mit dem Diskettenlaufwerk geliefert wird.

1. Schalten Sie das System ein.
2. Legen Sie die Diskette ein.
3. Benutzen Sie einen VC 20, Commodore 64 oder den Commodore 128 Personal Computer im C64 Modus, so geben Sie LOAD "HOW TO USE", 8 ein.
   Wird ein PLUS/4 oder der Commodore 128 Personal Computer im C128-Modus benutzt, so geben Sie DLOAD "HOW TO USE" ein.
4. Betätigen Sie die RETURN-Taste.
5. Danach wird folgende Meldung auf dem Bildschirm ausgegeben:

   SEARCHING FOR "HOW TO USE"
   LOADING
   READY.
   ■

6. Geben Sie RUN ein.
7. Betätigen Sie die RETURN-Taste.

WICHTIGER HINWEIS
Wird in diesem Handbuch das Format für einen Befehl angegeben, so unterliegt dieses einem bestimmten Schema. Sämtliche groß geschriebenen Angaben müssen genau so eingegeben werden, wie sie angegeben werden. Benutzen Sie jedoch NICHT die SHIFT-Taste bei der Eingabe dieser Befehle. Durch Kleinbuchstaben wird angegeben, was an dieser Stelle stehen muß. Sämtliche Angaben in Klammern sind optional.

In dem Format des auf der nächsten Seite angegebenen HEADER-Befehls müssen beispielsweise das Wort HEADER, der Großbuchstabe I in Iid, der Großbuchstabe D in Ddrive und der Großbuchstabe U in Uunit so eingegeben werden, wie angegeben. (Ddrive und Uunit sind optional.)
Wird der Diskettenname angegeben, so bedeutet dies, daß Sie einen Namen für die Diskette eingeben müssen. Sie müssen jedoch entscheiden, welcher Name benutzt wird. Auch das id in Iid ist Ihnen überlassen, genau so wie drive in Udrive. Bei der 1570/71 ist drive in Ddrive immer 0. Bei einem Doppellaufwerk kann jedoch 0 oder 1 benutzt werden. Allerdings gibt es gewisse Einschränkungen für die zu benutzenden Werte. In jedem Fall werden diese Einschränkungen sofort im Anschluß an das Format angegeben. (So kann der Diskettenname beispielsweise nicht mehr als 16 Zeichen umfassen, während unit normalerweise den Wert acht hat.)
Außerdem müssen sämtliche Satzzeichen genau an der Stelle und so eingegeben werden, wie in dem Format angegeben.
Schließlich betätigen Sie die RETURN-Taste am Ende jedes Befehls.
Um ein anderes Programm oder eine andere Datei zu laden, geben Sie einfach ihren Namen anstelle von HOW TO USE in Anführungszeichen an. HINWEIS: Das Programm HOW TO USE ist der Schlüssel zu der Test-/Demo-Diskette. Wird dieses Programm geladen und mit RUN ausgeführt, so liefert es Instruktionen zur Benutzung der restlichen Programme auf der Diskette. Um festzustellen, welche Programme auf der Test-/Demo-Diskette stehen, wird auf den Abschnitt 2.8 verwiesen.
Kann ein Programm mit der oben angegebenen Methode nicht richtig geladen oder ausgeführt werden, so handelt es sich unter Umständen um ein Programm in Maschinensprache. Wenn Sie selbst jedoch mit einer höheren Programmiersprache arbeiten, benötigen Sie keine Kenntnisse der Maschinensprache. In dem jeweiligen Benutzerhandbuch für ein Programm wird angegeben, ob das Programm in Maschinensprache geschrieben ist. Wenn ja oder wenn Sie Schwierigkeiten beim Laden eines bestimmten Programms haben, so fügen Sie einfach eine ",1" (Komma und die Ziffer 1) am Ende des Befehls hinzu.

2.7 VORBEREITUNG EINER NEUEN DISKETTE

Um auf einer Diskette Daten speichern zu können, muß sich auf ihr eine Struktur aus Spuren und Sektoren sowie ein Inhaltsverzeichnis befinden. Diese kann bei einer neuen Diskette mit dem HEADER- oder NEW-Befehl auf die Diskette geschrieben werden („Formatieren"). Es sollten Disketten mit der Bezeichnung SS/DD (1570) bzw. DS/DD (1571), besonders dann, wenn eine Diskette mit MFM-Format beschrieben werden soll. (DD = double density = doppelte Schreibdichte).

Folgender Befehl kann eingegeben werden, wenn der Rechner das BASIC 4.0 oder 7.0 benutzt (C128 im C128-Modus, PLUS /4):

HEADER "name" [,Iid][,Ddrive][,Uunit]

„name" ist ein beliebiger Name für die Diskette und hat eine Länge von maximal 16 Zeichen (einschließlich Leerzeichen). „id" kann zwei beliebige Zeichen umfassen, solange diese kein BASIC-Schlüsselwort (wie beispielsweise IF oder ON) alleine oder mit dem vorangestellten Großbuchstaben I darstellen. „drive" ist gleich 0. Der Parameter [,Uunit] kann entfallen, es sei denn, die Geräteadresse wurde wie in Anhang 9.1 beschrieben geändert und ist ungleich 8 (Wert für „unit" = Geräteadresse).
Der Befehl für den C64, VC 20 oder C128 im C64-Modus lautet:

OPEN 15,unit,15,"N[drive]:name,id"
CLOSE 15

unit, drive, name und id entsprechen den obigen Angaben. Der OPEN-Befehl wird im nächsten Kapitel erläutert. Für den Augenblick übernehmen Sie ihn einfach unverändert.
WARNUNG: Wenn Sie Disketten verwenden, die durch Umdrehen zweiseitig beschrieben worden sind, sollten Sie, wenn Sie mit der 1571 arbeiten, darauf achten, daß sich die 1571 im 1541-Modus befindet; andernfalls werden beide Seiten der Diskette formatiert, und es kommt zu einem unerwünschten Verlust der Daten auf der anderen Diskettenseite. Sie können den 1541-Modus erreichen, indem Sie die 1571 nach Entnahme der Diskette kurz aus- und danach wieder einschalten oder indem Sie auf dem Kommandokanal (sa = 15) den Befehl "U0>M0" ausgeben.

HINWEIS FÜR FORTGESCHRITTENE

Sollen Variablen für den Diskettennamen oder die ID benutzt werden, so gilt folgendes Format:

C128, PLUS/4: **HEADER (A$), I(B$),D0**
C64: **OPEN15,unit,15:PRINT#15,"N0:"+A$+B$0:CLOSE 15**

A$ enthält den Diskettennamen (maximal 16 Zeichen).
B$ enthält die ID (maximal zwei Zeichen).

Nachdem eine bestimmte Diskette formatiert wurde, kann sie jederzeit neu formatiert werden. Sie können ihren Namen schneller ändern und alle Dateien schneller löschen, wenn Sie die Disketten-ID im HEADER-Befehl nicht angeben.

2.8 DAS DISKETTENINHALTSVERZEICHNIS

Ein Disketteninhaltsverzeichnis ist eine Liste der Dateien auf einer Diskette. Um das Inhaltsverzeichnis mit dem C128 oder PLUS/4 auf dem Bildschirm darzustellen, geben Sie das Wort DIRECTORY auf einer leeren Zeile ein und betätigen die RETURN-Taste. Bei dem C128 können Sie auch einfach die F3-Taste betätigen. Durch den DIRECTORY-Befehl wird nichts im Speicher gelöscht, so daß Sie ein Verzeichnis jederzeit aufrufen können – selbst aus einem Programm heraus. Mit dem C64-Inhaltsverzeichnis-Befehl LOAD"**$**",8 (RETURN-Taste), LIST (RETURN-Taste) wird das Inhaltsverzeichnis in den Speicher geladen und ein dort stehendes Programm gelöscht.

Paßt ein Verzeichnis nicht ganz auf den Bildschirm, so rollt es nach oben, bis die letzte Zeile erreicht ist. Soll das Rollen vorübergehend angehalten, gestoppt oder verlangsamt werden, so werden in dem Benutzerhandbuch für den jeweiligen Computer die Tasten angegeben, die in diesem Fall benutzt werden müssen.

Damit Sie eine Vorstellung des Inhaltsverzeichnisses erhalten, laden Sie das Verzeichnis von der Test-/Demodiskette.

Die 0 links oben in der obersten Zeile entspricht der Laufwerksnummer der 1570/71 (bei einem Doppellaufwerk lautet diese Nummer 0 oder 1). Als nächstes wird der Diskettenname, gefolgt von der Disketten-ID ausgegeben – beide werden beim Formatieren der Diskette festgelegt.

Mit 2A am Ende der obersten Zeile wird angegeben, daß die 1570/71 die Version 2A des Commodore Disk Operating System (DOS) benutzt; A ist die Formatkennzeichnung. Jede der restlichen Zeilen enthält drei Informationen über die Dateien auf der Diskette. Auf der linken Seite jeder Zeile wird die Größe der Datei in Blöcken zu je 254 Zeichen angegeben. Vier Blöcke entsprechen etwa 1K im Speicher des Computers. In der Mitte der Zeile wird der Name der Datei in Anführungszeichen angegeben. Sämtliche Zeichen in Anführungszeichen sind Bestandteil des Dateinamens. Auf der rechten Seite jeder Zeile steht eine drei Buchstaben umfassende Abkürzung des Dateityps. Die Dateitypen werden in späteren Kapiteln erläutert.

Dateitypen

PRG – Programmdateien
SEQ – Sequentielle Dateien
REL – Relative Dateien
USR – Benutzerdateien
DEL – Gelöschte Dateien (dieser Dateityp wird nicht angezeigt)

HINWEIS: Direktzugriffsdateien werden normalerweise nicht im Inhaltsverzeichnis angezeigt. Sie werden im Kapitel 7 beschrieben.

In der letzten Zeile eines Verzeichnisses wird angegeben, wie viele Blöcke zur Benutzung bereitstehen. Diese Zahl geht von 664 (im 1541-Modus und bei der 1570) und 1328 (im 1571-Modus) auf einer neu formatierten Diskette bis zu 0 bei einer vollen Diskette.

2.8.1 SELEKTIVE DISKETTENINHALTSVERZEICHNISSE

Durch Änderung des Verzeichnisbefehls LOAD kann eine Art Unterverzeichnis erstellt werden, in dem nur ein einzelner ausgewählter Dateityp aufgelistet wird. So können Sie beispielsweise eine Auflistung sämtlicher sequentiellen Dateien (Kapitel 5) oder eine von allen relativen Dateien (Kapitel 6) anfordern. Dieser Befehl hat folgendes Format:

LOAD"$[drive]:joker=dateityp",unit (beim C64)

wobei Joker eine bestimmte Dateigruppe angibt und Dateityp die aus einem Buchstaben bestehende Abkürzung der nachfolgend aufgeführten Dateitypen darstellt:

P = Programm
S = Sequentiell
R = Relativ
U = Benutzer

Bei dem C128 und dem PLUS/4 lautet dieser Befehl:

DIRECTORY"joker=dateityp"[,Uunit][,Ddrive]

Einige Beispiele:

LOAD"$0:*=R",8 lädt DIRECTORY"*=R" und zeigt sämtliche relativen Dateien an.

LOAD"$0:Z*=R",8 lädt und DIRECTORY"Z*=R" zeigt ein Unterverzeichnis an, das aus sämtlichen relativen Dateien besteht, die mit dem Buchstaben Z beginnen. (Das Sternchen (*) wird in dem Abschnitt 2.9 näher erläutert.)

2.8.2 AUSDRUCKEN DES DISKETTENINHALTSVERZEICHNISSES

Um einen Audruck eines Inhaltsverzeichnisses zu erhalten, benutzen Sie die folgenden Befehle:

```
LOAD"$",8
OPEN4,4:CMD4:LIST
PRINT#4:CLOSE4
```

2.8.3 NICHT-GESCHLOSSENE DATEIEN

Ein Indikator, auf den Sie gelegentlich in einer Verzeichnisstelle stoßen werden, nachdem Sie mit dem Sichern von Programmen und Dateien begonnen haben, ist ein Stern (*), der direkt vor dem Dateityp einer Datei mit einer Länge von Null Blöcken steht. Mit ihm wird angegeben, daß die Datei nach dem Erstellen nicht richtig abgeschlossen wurde und nicht verläßlich ist. Diese „nicht richtig abgeschlossenen" Dateien müssen normalerweise von der Diskette gelöscht und neu geschrieben werden. Sie werden jedoch nicht mit dem SCRATCH-Befehl gelöscht. Sie können nur sicher mit dem VALIDATE- oder COLLECT-Befehl gelöscht werden. Einer dieser Befehle muß normalerweise benutzt werden, wenn eine nicht richtig abgeschlossene Datei auf der Diskette entdeckt wird. All diese Befehle werden in den folgenden Kapiteln beschrieben.
Zu der obigen Warnung gibt es zwei Ausnahmefälle: Der eine besteht darin, daß VALIDATE und COLLECT nicht auf Disketten benutzt werden können, die Direktzugriffsdateien enthalten (Kapitel 7). Die andere Ausnahme besteht darin, daß, wenn die Informationen in der nicht richtig abgeschlossenen Datei von besonderer Bedeutung waren und nicht ersetzt werden können, der Teil der Datei wiederhergestellt werden kann, der richtig geschrieben wurde. Diese Option wird im nächsten Kapitel beschrieben.

2.9 DATEIMEHRFACHNAMEN (JOKER)

Sie können spezielle Joker-Zeichen (Wildcard-Zeichen) benutzen, um ein Programm mit einem nur teilweise angegebenen Namen zu laden oder um die vorher beschriebenen selektiven Inhaltsverzeichnisse zu enthalten.
Bei den beiden Joker-Zeichen handelt es sich um den Stern (*) und das Fragezeichen (?). Der Unterschied zwischen diesen beiden Zeichen besteht darin, daß der Stern für alle auf dieses Sternchen folgende Zeichen steht, während das Fragezeichen nur für dieses eine Zeichen steht. Hier einige Beispiele und ihre Ergebnisse:
Mit LOAD"A*",8 wird die erste Datei auf der Diskette, die mit A beginnt, unabhängig davon, was auf diesen Buchstaben weiter folgt, geladen.
Mit DLOAD"SM?TH" wird die erste Datei geladen, die mit SM beginnt, mit TH endet und über ein beliebiges dazwischenliegendes Zeichen verfügt.
Mit DIRECTORY"Q*" wird ein Verzeichnis der Dateien angezeigt, deren Name mit Q beginnen.
LOAD"*",8 ist ein Sonderfall. Wird ein Stern alleine als Name benutzt, so steht es für die als letztes benutzte Datei (bei dem C64 und C128 im C64-Modus).

Mit LOAD"0:*",8 wird die erste Datei auf der Diskette geladen (C64 und C128 im C64-Modus).
Mit DLOAD"*" wird die erste Daei auf der Diskette geladen (PLUS/4 und C128 im C128-Modus).

Geräteadresse

Die Geräteadresse kann sowohl soft- als auch hardwaremäßig geändert werden. Die softwaremäßige Änderung erfolgt wie im Handbuch der 1571 beschrieben; die DIP-Switches für die hardwaremäßige Umstellung sind nach dem Öffnen des Gehäuses auf der rechten Seite der Leiterplatte über dem Laufwerk zu erreichen. Um Auseinandersetzungen über Garantieansprüche zu vermeiden, sollte ein Umstellen der Schalter nur von einem Commodore-Fachhändler vorgenommen werden.

KAPITEL 3

BASIC-BEFEHLE FÜR DIE ARBEIT MIT DISKETTENLAUFWERKEN

3.1 BASIC 2.0

In diesem Kapitel werden die Diskettenbefehle beschrieben, die mit dem VC 20, Commodore 64 oder Commodore 128 Personal Computer im C64-Modus benutzt werden. Hier handelt es sich um BASIC-2.0-Befehle.
Sie können Befehlsdaten über einen sogenannten Befehlskanal an das Laufwerk senden. Als erstes wird der Kanal mit folgendem Befehl geöffnet:

```
OPEN15,8,15
```

Bei der ersten 15 handelt es sich um eine Datei- oder Kanalnummer. Auch wenn hier eine beliebige Zahl von 1 bis 255 benutzt werden kann, wurde 15 gewählt, da diese Zahl mit der Sekundäradresse 15 übereinstimmt, bei der es sich um die Adresse des Befehlskanals handelt. Die mittlere Nummer ist die Primäradresse, besser bekannt als Geräteadresse. Diese Nummer ist im allgemeinen 8, es sei denn, sie wird vom Benutzer geändert (siehe Anhang 9.1).
Nachdem der Kanal geöffnet wurde, senden Sie Informationen mit dem PRINT#-Befehl an das Diskettenlaufwerk und empfangen Informationen mit dem INPUT#-Befehl. Der Kanal muß mit dem Befehl CLOSE15 geschlossen werden.
In den folgenden Beispielen wird dargestellt, wie der Befehlskanal benutzt wird, um eine nichtformatierte Diskette mit den NEW-Befehl vorzubereiten:

```
OPEN15,8,15
PRINT#15,"NEW[laufwerk]:diskettenname,id"
CLOSE15
```

Diese beiden ersten Anweisungen können wie folgt kombiniert, und der NEW-Befehl kann abgekürzt werden:

```
OPEN15,8,15,"N[laufwerk]:diskettenname,id"
```

Ist der Befehlskanal schon geöffnet, so müssen Sie folgendes Format benutzen.

Wird versucht, einen schon geöffneten Kanal zu öffnen, so wird die Fehlermeldung "FILE OPEN" (Datei geöffnet) ausgegeben.

PRINT#15,"N[laufwerk]:diskettenname,id"

3.1.1 FEHLERABFRAGE

Blinkt die grüne Laufwerksanzeige, so müssen Sie bei BASIC 2.0 ein kleines Programm schreiben, um die Ursache des Fehlers festzustellen. Dadurch gehen schon im Speicher stehende Programmvariablen verloren. Nachfolgend ein Programm zur Fehlerprüfung:

```
10 OPEN 15,8,15
20 INPUT#15,EN,EM$,ET,ES
30 PRINT EN,EM$,ET,ES
40 CLOSE 15
```

Dieses kleine Programm liest den Fehlerkanal in vier BASIC-Variablen (die nachfolgend beschrieben werden) und druckt die Ergebnisse auf dem Bildschirm aus. Unabhängig davon, ob ein Fehler vorhanden ist oder nicht, wird eine Meldung angezeigt. Ist jedoch ein Fehler vorhanden, so löscht das Programm den Fehler aus dem Diskettenspeicher und schaltet die Fehleranzeige bei dem Diskettenlaufwerk aus.
Sobald die Meldung auf dem Bildschirm angezeigt wird, kann ihre Bedeutung in Anhang 9.2 nachgeschlagen werden. Dort wird auch beschrieben, wie der Fehler behoben werden kann.
Für die Benutzer, die eigene Programme schreiben möchten, wird nachfolgend eine kleine Subroutine zur Fehlerabfrage angegeben, die in eigenen Programmen verwendet werden kann:

```
59980 REM FEHLER-KANAL LESEN
59990 INPUT#15,EN,EM$,ET,ES
60000 IF EN>1 THEN PRINT EN,EM$,ET,ES:STOP
60010 RETURN
```

Hierbei wird davon ausgegangen, daß Kanal 15 schon vorher in dem Programm geöffnet wurde, und daß diese Datei am Ende des Programms geschlossen wird. Die Subroutine liest den Fehlerkanal und setzt die Ergebnisse in die Variablen EN (Fehlernummer), EM$ (Fehlermeldung), ET (Fehlertrack) und ES (Fehlersektor). Von diesen vier Variablen braucht nur EM$ eine Zeichenfolge zu sein. Sie können

auch andere Variablennamen auswählen, obwohl sich diese Namen für diesen Zweck eingebürgert haben.
Zwei Fehlernummern bezeichnen keine Fehler: 0 bedeutet, daß alles in Ordnung ist. Mit 1 wird angegeben, wie viele Dateien durch einen SCRATCH-Befehl (der später in diesem Kapitel beschrieben wird) gelöscht wurden. Wird ein anderer Fehlerstatus ausgelesen, so wird die Fehlermeldung in Zeile 60000 ausgedruckt und das Programm abgebrochen.
Da es sich um eine Subroutine handelt, kann mit dem BASIC-Befehl GOSUB im Direkt-Modus oder aus einem Programm auf sie zugegriffen werden. Die RETURN-Anweisung in Zeile 60010 sorgt dafür, daß in den Direkt-Modus bzw. zur nächsten Anweisung zurückgesprungen wird.
Hinweis 1: Es wird empfohlen, Kanal 15 einmal am Anfang eines Programms zu öffnen und die Datei erst am Ende des Programms zu schließen, nachdem alle anderen Dateien schon geschlossen wurden. Wird sie einmal bei Beginn geöffnet, so ist die Datei offen, wann immer sie für Diskettenbefehle an anderer Stelle in dem Programm benötigt wird.
Hinweis 2: Hält BASIC mit einem Fehler an, während Dateien geöffnet sind, so beendet BASIC diese, ohne sie richtig auf der Diskette abzuschließen. Um sie richtig auf der Diskette zu schließen, wird folgender Befehl eingegeben:

```
CLOSE 15:OPEN 15,8,15,"I0":CLOSE 15
```

Dadurch wird der Befehlskanal geöffnet und sofort wieder zusammen mit allen anderen Diskettendateien geschlossen. Wird eine Diskettendatei sowohl in BASIC als auch auf der Diskette nicht richtig geschlossen, so kann die ganze Datei verlorengehen.
Hinweis 3: Bei einer Fehlermeldung handelt es sich nicht immer um einen Fehler. So wird beispielsweise Fehler 73 "CBM DOS 3.0 1570" oder "CBM DOS 3.0 1571" angezeigt, wenn der Diskettenfehlerkanal beim Einschalten des Computers gelesen wird, bevor Diskettenbefehle gesendet werden. Auf diese Art und Weise kann bequem überprüft werden, welche DOS-Version gerade benutzt wird. Wird diese Meldung jedoch später nach anderen Diskettenbefehlen angezeigt, so weist sie darauf hin, daß die für das Formatieren der Diskette benutzte DOS-Version nicht mit der DOS-Version im Laufwerk übereinstimmt. DOS ist das Diskettenbetriebssystem.
Hinweis 4: Um das Laufwerk zurückzusetzen, geben Sie folgenden Befehl ein: OPEN 15,8,15,"UJ":CLOSE 15. Dies gilt auch für das Senden von UI+ oder UI.

3.1.2 SAVE

Mit dem SAVE-Befehl wird ein Programm oder eine Datei zur späteren Benutzung auf

einer Diskette abgespeichert. Bevor Sie Daten mit SAVE auf einer Diskette abspeichern können, muß die Diskette, wie zuvor beschrieben, formatiert worden sein.

FORMAT DES SAVE-BEFEHLS

SAVE "[laufwerk:]dateiname",gerät

„dateiname" ist eine Zeichenfolge aus maximal 16 Zeichen, vor der die Laufwerksnummer und ein Doppelpunkt stehen und auf die die Gerätenummer der Diskette, normalerweise 8, folgt.
Der SAVE-Befehl kann jedoch nicht beim Kopieren von Programmen benutzt werden, die nicht in dem BASIC-Textbereich stehen, wie beispielsweise „DOS 5.1" bei dem C64. Um dieses Programm und ähnliche in der Maschinensprache geschriebene Programme zu kopieren, wird ein Monitorprogramm in Maschinensprache benötigt.

BEISPIEL

Nachfolgend die Syntax für das Sichern einer Kopie von „PROGRAMM" mit SAVE:

SAVE "0:PROGRAMM",8

FORMAT VON MONITOR SAVE

S"[laufwerk:]dateiname",gerät,startadresse,endadresse+1

„laufwerk" entspricht der Laufwerksnummer, d.h. 0 bei der 1570/71; „dateiname" ist ein beliebiger, gültiger Dateiname aus maximal 14 Zeichen, wobei zwei Zeichen für Laufwerksnummer und Doppelpunkt berücksichtigt werden. „gerät" ist eine zweistellige Geräteadresse, normalerweise 08; die führende 0 ist obligatorisch. Der abzuspeichernde Adreßbereich wird in Hexadezimalzahlen angegeben, jedoch ohne führendes Dollarzeichen ($).

3.1.2.1 SAVE WITH REPLACE

Ist eine Datei schon vorhanden, so kann eine neue Datei nicht mehr unter demselben Namen gesichert werden, da das Diskettenlaufwerk nur eine Kopie eines bestimmten Dateinamens und -typs pro Diskette zuläßt. Dieses Problem kann mit den später beschriebenen Befehlen RENAME und SCRATCH umgangen werden. Möchten Sie jedoch nur ein Programm oder eine Datendatei durch eine geänderte Version ersetzen, so gibt es noch einen bequemeren Befehl, der als SAVE WITH REPLACE

oder (@SAVE) bekannt ist. Mit dieser Befehlsoption wird das Diskettenlaufwerk angewiesen, eine in dem Diskettenverzeichnis gefundene Datei mit demselben Namen durch die neue Datei zu ersetzen.

FORMAT VON SAVE WITH REPLACE

SAVE "@laufwerk:dateiname",gerät

wobei die üblichen Parameter gelten, nur daß ein führendes @-Zeichen hinzugefügt wird. Die Angabe „laufwerk:" ist hier erforderlich.

BEISPIEL:

SAVE"@0:PROGRAMM",8

Bei der Ausführung des Befehls wird die neue Version zunächst vollständig abgespeichert und danach erst die alte Version gelöscht. Dadurch besteht eine geringe Gefahr, daß beispielsweise durch einen Stromausfall im Laufe des Verfahrens sowohl die alte als auch die neue Version der Datei zerstört werden kann. Bei der alten Kopie geschieht nichts, bevor die neue Version nicht vollständig abgespeichert worden ist.
Achtung – verwenden Sie den Befehl @SAVE nicht auf einer fast vollen Diskette. Verwenden Sie diesen Befehl nur, wenn genügend Platz auf der Diskette vorhanden ist, um eine zweite vollständige Kopie des zu ersetzenden Programms abspeichern zu können. Aufgrund der Arbeitsweise von @SAVE befinden sich sowohl die alte als auch die neue Version der Datei zu einem bestimmten Zeitpunkt gleichzeitig auf der Diskette. Dies stellt einen Schutz vor Programmverlust dar. Ist nicht genügend Platz für die zweite Kopie auf der Diskette vorhanden, so wird nur so viel der neuen Version gesichert, wie Platz vorhanden ist. Nach Ausführung des Befehls wird in dem Inhaltsverzeichnis angegeben, daß die neue Version vorhanden ist, jedoch nicht genügend Blöcke im Speicher belegt, um mit der Kopie im Speicher übereinzustimmen. Der VERIFY-Befehl (der nachfolgend beschrieben wird) entdeckt dieses Problem bedauerlicherweise nicht, da der Teil, der gesichert wurde, richtig abgespeichert wurde.

3.1.3 VERIFY

Obwohl das Verifizieren bei einem Diskettenlaufwerk nicht so erforderlich ist wie bei einer Kassette, kann mit dem BASIC-Befehl VERIFY noch einmal geprüft werden, ob eine Programmdatei richtig auf der Diskette gesichert wurde. Er wird wie der LOAD-Befehl verwendet, und er vergleicht jedes Zeichen in dem Programm mit dem entsprechenden Zeichen im Speicher des Computers, ohne ihn zu verändern.

Unterscheidet sich das abgespeicherte Programm auf der Diskette auch nur in einem Bit von dem Programm im Speicher, so wird dies mit „VERIFY ERROR" (Prüffehler) angezeigt. Dies bedeutet nicht, daß die Kopie fehlerhaft ist. Müssen die Kopien jedoch genau identisch sein, so liegt ein Problem vor.
Natürlich hat es keinen Sinn, die Kopie eines Programms auf der Diskette mit VERIFY zu überprüfen, nachdem das Original nicht mehr im Speicher steht. Liegt keine Vorlage für den Vergleich vor, so wird stets ein Fehler angezeigt. Die Diskettenkopie wird stets automatisch geprüft, wenn sie auf die Diskette geschrieben wird.

FORMAT DES VERIFY-BEFEHLS

VERIFY"[laufwerk:]dateiname",gerät,verschiebungsflag

„laufwerk" ist eine optionale Laufwerksnummer, „dateiname" ist eine Zeichenfolge, die einem Dateinamen mit oder ohne Joker-Zeichen entspricht, während „geräteadresse" die Nummer der Disketteneinheit darstellt, normalerweise 8. Ist das Verschiebungsflag vorhanden und hat es den Wert 1, so wird die Datei an der Speicheradresse beginnend überprüft, an der sie ursprünglich gesichert wurde; andernfalls wird sie von der Adresse an überprüft, an der sich BASIC-Programme im Speicher befinden.
Eine Alternativ-Form des Befehls sieht folgendermaßen aus:

VERIFY"*",gerät

Mit diesem Befehl wird die als letztes benutzte Datei überprüft, ohne daß ihr Name oder die Laufwerksnummer eingegeben werden muß. Dieser Befehl kann jedoch im Anschluß an SAVE-WITH-REPLACE nicht einwandfrei benutzt werden, da die letzte benutzte Datei die gelöschte Datei war und das Laufwerk versucht, die gelöschte Datei mit dem Programm im Speicher zu vergleichen. Hier entsteht zwar kein Schaden, jedoch wird immer „VERIFY ERROR" (Prüffehler) angezeigt. Um VERIFY im Anschluß an @SAVE benutzen zu können, muß mindestens ein Teil des zu überprüfenden Dateinamens angegeben werden.
Ein weiterer Hinweis zu VERIFY: Wird eine verschobene BASIC-Datei mit VERIFY überprüft, so wird nahezu immer ein Fehler angegeben, der auf Änderungen in den Link-Pointern der BASIC-Programme während der Verschiebung zurückzuführen ist. Am besten werden Dateien mit VERIFY überprüft, die von Maschinen desselben Typs und derselben Speichergröße abgespeichert wurden. So kann beispielsweise ein von einem PLUS/4 abgespeichertes BASIC-Programm nicht problemlos mit einem C64 überprüft werden, selbst wenn das Programm auf beiden Maschinen einwandfrei ausgeführt werden kann. Dies sollte keine Rolle spielen, da Dateien nur dann auf anderen Maschinen als denen, auf denen sie erstellt wurden, überprüft

werden, wenn zwei Diskettendateien miteinander verglichen werden. In diesem Fall wird eine Datei geladen und mit der anderen Datei verglichen. Dies kann nur auf derselben Maschine und mit derselben Speichergröße wie beim Erstellen der Datei vorgenommen werden.

3.1.4 SCRATCH

Mit dem SCRATCH-Befehl können unerwünschte Dateien gelöscht und der von ihnen belegte Platz zur Benutzung durch andere Dateien freigegeben werden. Mit diesem Befehl können eine einzelne oder mehrere Dateien gleichzeitig mit Hilfe von Joker-Zeichen gelöscht werden.

FORMAT DES SCRATCH-BEFEHLS

PRINT#15,"SCRATCH[laufwerk]:dateiname"

oder in der abgekürzten Form:

PRINT#15,"S[laufwerk]:dateiname"

„dateiname" kann ein beliebiger Dateiname oder eine Kombination von Zeichen und Joker-Zeichen darstellen. Wie üblich wird davon ausgegangen, daß der Befehlskanal schon als Datei 15 geöffnet wurde. Auch wenn es nicht absolut erforderlich ist, wird empfohlen, die Laufwerksnummer in SCRATCH-Befehlen anzugeben. Sie ist bei der 1570/71 stets 0.
Wird der Fehlerkanal nach einem SCRATCH-Befehl beim Auslesen des Fehlerkanals (s. entsprechenden Abschnitt in diesem Kapitel) überprüft, so geht aus dem Wert für ET (Fehlertrack) hervor, wie viele Dateien gelöscht wurden. Enthält die Diskette beispielsweise Programmdateien namens „Test", „Train", „Truck" und „Tail", so können alle vier Dateien zusammen mit anderen Dateien, die mit dem Buchstaben „T" beginnen, mit folgendem Befehl gelöscht werden:

PRINT#15,"S0:T*"

Um sich zu vergewissern, ob die Dateien gelöscht wurden, können Sie, wenn Sie die zuvor beschriebene Fehlerabfrage-Subroutine auf Ihrem Programm zusammen eingegeben haben, folgenden Befehl eingeben:

GOSUB 59990

Mit diesem Befehl wird die Subroutine zur Fehlerprüfung aufgerufen, die vorher in diesem Kapitel beschrieben wurde. Handelte es sich bei den vier aufgeführten

Dateien um die einzigen Dateien, die mit „T" beginnen, so wird folgendes auf dem Bildschirm ausgegeben:

```
01,FILES SCRATCHED,04,00

READY.
```

Mit „04" wird angegeben, daß vier Dateien gelöscht wurden.

3.1.4.1 WEITERE INFORMATIONEN ÜBER SCRATCH

SCRATCH ist ein leistungsfähiger Befehl, der mit Vorsicht benutzt werden sollte, damit wirklich nur die Dateien gelöscht werden, die auch wirklich gelöscht werden sollen. Wird dieser Befehl mit Jokern benutzt, so wird empfohlen, daß Sie zuerst dieselben Joker in einem DIRECTORY-Befehl benutzen. Auf diese Weise sehen Sie genau, welche Dateien gelöscht werden. Es kommt also zu keinen unerfreulichen Überraschungen, wenn dieselben Joker in dem SCRATCH-Befehl benutzt werden.

3.1.4.2 UNSCRATCH

Wird eine Datei versehentlich gelöscht, so besteht immer noch eine Möglichkeit, sie wiederherzustellen. Wie bei dem BASIC-Befehl NEW wird bei SCRATCH nicht wirklich eine Datei selbst gelöscht; es werden nur die Zeiger zu der Datei in dem Disketteninhaltsverzeichnis gelöscht. Hierzu befinden sich auf der Test-/Demo-Diskette ein „Unscratch"- sowie das DOS-SHELL-Programm.
Wichtiger Hinweis: Haben Sie eine Datei versehentlich gelöscht und wollen dies rückgängig machen, dürfen Sie vor dem Aufruf eines Unscratch-Programms nichts mehr auf der Diskette abspeichern.
HINWEIS: Wird eine Datei innerhalb von DOS SHELL (siehe Kapitel 4) versehentlich mit SCRATCH gelöscht, so kann dies mit der Funktion RESTORE FILES wieder rückgängig gemacht werden.

3.1.4.3 NICHT-GESCHLOSSENE DATEIEN

Eine nicht richtig abgeschlossene Datei darf niemals gelöscht werden. Bei diesen Dateien steht in der Auflistung des Inhaltsverzeichnisses ein Stern (*) vor dem Dateityp. Der Stern bedeutet, daß die Datei nicht richtig abgeschlossen wurde und daß demzufolge keine gültige Kette von Sektorverbindungen vorhanden ist, die der SCRATCH-Befehl beim Löschen der Datei verfolgen kann.

Wird eine derartige Datei mit SCRATCH gelöscht, so besteht die Gefahr, daß Sektoren freigegeben werden, die noch von anderen Programmen oder Dateien benötigt werden. Dadurch entsteht ein ständiger Schaden für diese Programme oder Dateien, wenn später weitere Dateien auf der Diskette angelegt werden. Stoßen Sie auf eine nicht richtig abgeschlossene Datei oder entdecken Sie zu spät, daß Sie eine derartige Datei gelöscht haben, so überprüfen Sie die Diskette sofort mit dem VALIDATE-Befehl, der später in diesem Kapitel beschrieben wird. Wurden Dateien auf der Diskette abgelegt, nachdem die nicht richtig abgeschlossene Datei gelöscht wurde, so empfiehlt es sich, die gesamte Diskette auf eine neue Diskette zu kopieren. Dies muß jedoch mit einem Dateikopierprogramm (file copy) und nicht mit einem Sicherungsprogramm (backup copy) geschehen, andernfalls ergibt sich wieder dasselbe Problem auf der neuen Diskette. Sobald die neue Diskette erstellt wurde, vergleichen Sie die Anzahl der freien Blöcke in ihrem Inhaltsverzeichnis mit der Anzahl der freien Blöcke auf der Originaldiskette. Stimmen die Zahlen überein, so ist wahrscheinlich kein Schaden entstanden. Wenn nicht, wurde möglicherweise zumindest eine Datei auf der Diskette zerstört. In diesem Fall muß die ganze Diskette sofort überprüft werden.

3.1.4.4 GESCHÜTZTE DATEIEN

Gelegentlich enthält eine Diskette eine oder mehrere geschützte Dateien – Dateien, die nicht mit dem SCRATCH-Befehl gelöscht werden können. Derartige Dateien können anhand des „<"-Zeichens erkannt werden, das in dem Verzeichniseintrag unmittelbar auf den Dateityp folgt. Soll eine geschützte Datei gelöscht werden, so müssen Sie ein Diskettenmonitorprogramm für Disketten benutzen, um Bit 6 des Dateityp-Bytes in dem Verzeichniseintrag auf der Diskette zu löschen. Um eine Datei zu sperren, müssen Sie Bit 6 desselben Bytes setzen.

3.1.5 RENAME

Mit dem RENAME-Befehl kann der Name eines Programms oder einer anderen Datei in dem Disketteninhaltsverzeichnis geändert werden. Da nur das Verzeichnis von diesem Befehl betroffen ist, arbeitet RENAME sehr schnell.

FORMAT DES RENAME-BEFEHLS

PRINT# 15,"RENAME[laufwerk]:neuer name = alter name"

oder in abgekürzter Form:

PRINT# 15,"R[laufwerk]:neuer name = alter name"

wobei „neuer name" dem der Datei zuzuweisenden Namen entspricht, während „alter name" dem jetzt benutzten Namen entspricht. Bei „neuer name" kann es sich um einen beliebigen gültigen Dateinamen aus maximal 16 Zeichen handeln. Es wird davon ausgegangen, daß Datei 15 schon als Befehlskanal geöffnet wurde.
Eine Vorsichtsmaßnahme: Vergewissern Sie sich, ob die umzubenennende Datei richtig abgeschlossen wurde, bevor sie umbenannt wird.

BEISPIELE

Direkt vor dem Sichern einer neuen Kopie eines „Kalender"-Programms, könnten Sie beispielsweise folgenden Befehl eingeben:

```
PRINT#15,"R0:KALENDER/BACKUP=KALENDER"
```

Um ein Programm namens „BOOT", das momentan als erstes Programm auf einer Diskette steht, an eine andere Stelle im Inhaltsverzeichnis zu verschieben, könnten Sie zunächst auch folgenden Befehl eingeben:

```
PRINT#15,"R0:TEMPORAER=BOOT"
```

Danach ist mit Hilfe des COPY-Befehls, der später beschrieben wird, eine Kopie der Datei anzufertigen. Dadurch wird „TEMP" zu einer neuen Kopie von „BOOT". Die Folge wird dann mit einem SCRATCH-Befehl beendet, um „BOOT" zu löschen.

3.1.6 UMBENENNEN UND LÖSCHEN PROBLEMATISCHER DATEIEN

Gelegentlich können Sie auf eine Datei stoßen, die einen seltsamen Dateinamen hat, wie beispielsweise ein einzelnes Komma (",") oder eine Datei, die ein Leerzeichen mit SHIFT-Taste enthält. (Ein Leerzeichen mit SHIFT-Taste sieht wie ein normales Leerzeichen aus. Kann eine Datei mit einem Leerzeichen in dem Namen jedoch nicht richtig geladen werden und ist ansonsten alles richtig, so handelt es sich wahrscheinlich um ein Leerzeichen mit SHIFT-Taste.) Möglicherweise können sie auch auf eine Datei stoßen, die nicht ausdruckbare Zeichen enthält. Jede dieser Dateien kann sich als problematisch erweisen. So sind beispielsweise Komma-Dateien eine Ausnahme zu der Regel, daß zwei Dateien nicht denselben Namen aufweisen können. Da es nicht möglich sein sollte, eine Datei zu erstellen, deren Name nur aus einem Komma besteht, erwartet das DOS auch nicht, daß Sie dies doch machen.
Dateinamen mit einem Leerzeichen mit SHIFT-Taste können sich ebenfalls als problematisch erweisen, da die Diskette das Leerzeichen mit SHIFT-Taste als Ende des Dateinamens interpretiert und alle Angaben ausdruckt, die auf das Anführungszeichen folgen, welches das Ende eines Dateinamens im Inhaltsverzeichnis mar-

kiert. Diese Technik kann nützlich sein, da sie die Benutzung eines langen Dateinamens zuläßt. Das DOS erkennt einen kleinen Teil dieses Dateinamens als den ganzen Namen, ohne daß Joker benutzt werden.
Bei problematischen Dateinamen können Sie in solchen Fällen die CHR$-Funktion benutzen. Dadurch können sie meistens in einem RENAME-Befehl aufgenommen werden. Gelingt dies nicht, so können Sie die Joker-Zeichen auch in einem SCRATCH-Befehl benutzen. Dadurch erhalten Sie eine Möglichkeit, den Namen ohne die problematischen Zeichen anzugeben. Dies bedeutet jedoch gleichzeitig auch den Verlust der Datei.
Ist es Ihnen beispielsweise gelungen, eine Datei namens "MOVIES mit einem zusätzlichen Anführungszeichen vor dem Dateinamen zu erstellen, so können Sie diesen Namen mit dem CHR$-Äquivalent für ein Anführungszeichen im RENAME-Befehl in "MOVIES" umbenennen.

```
PRINT#15,"R0:MOVIES="+CHR$(34)+"MOVIES"
```

Mit der Funktion CHR$(34) wird ein Anführungszeichen in die Befehlsfolge gesetzt, ohne daß sich dies auf BASIC auswirkt. Bei einem Dateinamen, der ein Leerzeichen mit SHIFT-Taste enthält, wird ähnlich vorgegangen, allerdings wird hier die Funktion CRH$(160) benutzt.
In den Fällen, in denen selbst dieses Verfahren nicht benutzt werden kann, beispielsweise wenn die Diskette eine Komma-Datei (eine Datei namens ",") enthält, können Sie das Problem folgendermaßen lösen:

```
PRINT#,"S0:?"
```

In diesem Beispiel werden sämtliche Dateien mit aus einem Zeichen bestehenden Namen gelöscht.
Je nach Problem muß bei der Auswahl der Joker-Zeichen, die sich nur auf die gewünschte Datei auswirken, sehr umsichtig vorgegangen werden. Unter Umständen müssen Sie andere Dateien zuerst umbenennen, um zu verhindern, daß sie gelöscht werden.
In einigen Fällen ist es einfacher, noch benötigte Dateien auf eine andere Diskette zu kopieren und die problematischen Dateien auf der alten Diskette zu belassen.

3.1.7 COPY

Mit dem COPY-Befehl können Sie eine zusätzliche Kopie jedes Programms oder jeder Datei auf der Diskette anfertigen. Bei einem Einzellaufwerk wie der 1570/71 muß die Kopie auf derselben Diskette erfolgen. Das bedeutet, daß Sie der kopierten Datei einen anderen Namen zuweisen müssen. Mit diesem Befehl

können auch bis zu vier sequentielle Dateien miteinander kombiniert werden. Dateien werden in der Reihenfolge verknüpft, in der sie im Befehl angegeben werden. Die zu kopierenden und die anderen Dateien auf der Diskette werden nicht geändert. Die Dateien müssen abgeschlossen werden, bevor sie kopiert oder verknüpft werden.

FORMAT DES COPY-BEFEHLS

PRINT#15,"COPY[laufwerk]:neue datei=alte datei"

oder in der abgekürzten Form:

PRINT#15,"C[laufwerk]:neue datei=alte datei"

BEISPIELE:

PRINT#15,"COPY0:BACKUP=ORIGINAL

PRINT#15,"C0:BACKUP=ORIGINAL

wobei „laufwerk" der Laufwerksnummer, „neue datei" der Kopie und „alte datei" dem Original entspricht.

FORMAT DER VERKETTUNGSOPTION

PRINT#15,"C[laufwerk]:neue datei=datei 1,datei 2,datei 3,datei 4"

wobei „laufwerk" stets 0 ist.

HINWEIS: Die Länge einer Befehlsfolge (Befehl und Dateinamen) ist auf 41 Zeichen beschränkt.

BEISPIELE

Nachdem die Datei namens „BOOT" im Beispiel des letzten Abschnittes in „TEMPORAER" umbenannt worden ist, können Sie mit dem COPY-Befehl eine zusätzliche Kopie des Programms an anderer Stelle auf der Diskette unter dem Originalnamen anfertigen:

PRINT#15,"C0:BOOT=TEMPORAER"

Nachdem einige kleine sequentielle Dateien erstellt wurden, die problemlos zusam-

men mit einem benutzten Programm in den Speicher passen, können Sie die Verkettungsoption benutzen, um die Dateien in einer Masterdatei zu kombinieren, selbst wenn das Ergebnis für den Speicher zu groß ist. Allerdings müssen Sie sich vergewissern, daß die neue Datei auf den restlichen Platz auf der Diskette paßt; sie wird genau so groß, wie die einzelnen Dateien zusammen.

```
PRINT#15,"C0:A-Z=A-G,H-M,N-Z"
```

HINWEIS: Doppellaufwerke können diesen Befehl besser ausnutzen, indem Dateien von der Diskette in dem einen Laufwerk auf die im anderen kopiert werden können. Um dies mit der 1570/71 auszuführen, suchen Sie die benötigten Programme auf der Test-/Demo-Diskette oder benutzen das DOS-SHELL-Programm, das in Kapitel 4 beschrieben wird.

3.1.8 VALIDATE

Der VALIDATE-Befehl erstellt eine neue Block Availability Map (BAM) (d.h. Blockbelegungstabelle) der eingelegten Diskette. Hierbei werden nur die Sektoren als belegt gekennzeichnet, die noch von gültigen und korrekt abgeschlossenen Dateien benutzt werden. Alle anderen Sektoren (Blöcke) werden als frei gekennzeichnet und können danach von neuen Dateien belegt werden. Sämtliche nicht richtig abgeschlossenen Dateien werden automatisch gelöscht. Aus dieser kurzen Funktionsbeschreibung ergibt sich weder die Leistungsfähigkeit noch die Gefahr des VALIDATE-Befehls. Sein großer Vorteil liegt darin, daß er viele Disketten, deren Inhaltsverzeichnisse oder Block Availability Maps durcheinander geraten sind, wieder in einen einwandfreien Zustand versetzt. Immer, wenn die von den Dateien auf einer Diskette benutzten Blöcke plus den als frei angegebenen Blöcken nicht mehr 664 (im 1541-Modus und bei der 1570) bzw. 1328 (im 1571-Modus) Blöcke ergeben, sollte ein VALIDATE durchgeführt werden. Es gibt jedoch eine Ausnahme, die in diesem Kapitel noch beschrieben wird. Auch wenn eine Diskette nicht korrekt abgeschlossene Datei enthält, die durch einen Stern (*) neben dem Dateityp im Inhaltsverzeichnis markiert wird, sollte diese Datei mit VALIDATE gelöscht und die von ihr belegten Blöcke freigegeben werden. Mit Ausnahme der im folgenden beschriebenen Fälle wird unbedingt empfohlen, die Diskettenbelegung mit VALIDATE neu zu ordnen, falls irgendwelche Zweifel an ihrer Belegung auftreten.

Nun zu den bereits angekündigten Ausnahmen. Disketten, die Direktzugriffsdateien, wie in Kapitel 7 beschrieben, enthalten, dürfen nicht mit VALIDATE bearbeitet werden. Die meisten Direktzugriffsdateien verfügen über keine Verkettung, die jedoch von VALIDATE benötigt wird, und so kann die Benutzung von VALIDATE in solchen Fällen zu einer Freigabe sämtlicher Blöcke der Direktzugriffsdateien führen, wobei ihr ganzer Inhalt verlorengeht, wenn andere Dateien hinzugefügt werden. Es sei denn, es werden ausdrücklich andere Angaben gemacht, benutzen Sie den VALIDATE-Befehl niemals auf einer Diskette, die Direktzugriffsdateien enthält.

HINWEIS: Direktzugriffsdateien sind nicht gleichbedeutend mit relativen Dateien, die in Kapitel 6 beschrieben werden. Mit relativen Dateien kann VALIDATE problemlos benutzt werden.

FORMAT des VALIDATE-BEFEHLS

```
PRINT#15,"VALIDATE[laufwerk]"
```

oder als Abkürzung

```
PRINT#15,"V[laufwerk]"
```

wobei „laufwerk" der Laufwerksnummer entspricht. Wie üblich wird davon ausge-

gangen, daß Datei 15 als Befehlskanal geöffnet wurde und nach Ausführung des Befehls wieder geschlossen wird.

BEISPIEL:

PRINT#15,"V0" oder **PRINT#15,"V"**

3.1.9 INITIALIZE

Ein Befehl, der bei der 1570/71 nicht sehr häufig benötigt wird, jedoch gelegentlich von Wert ist, ist der INITIALIZE-Befehl. Bei der 1570/71 und nahezu allen anderen Commodore-Laufwerken wird diese Funktion automatisch ausgeführt, sobald eine neue Diskette eingelegt wird. (Durch die Lichtschranke, die den Schreibschutz prüft, wird festgestellt, wenn eine Diskette ausgewechselt wird.)
Wird INITIALIZE als Befehl oder automatisch beim Wechsel der Diskette durchgeführt, so wird die BAM der aktuellen Diskette erneut in einen Pufferspeicher gelesen und kann vom DOS weiterverarbeitet werden. Diese Information muß stets korrekt sein, damit neue Dateien richtig auf der Diskette abgespeichert werden können. Da die Initialisierung automatisch erfolgt, muß dieser Befehl nur benutzt werden, wenn die Belegungsinformation im Speicher der Floppystation aufgrund eines bestimmten Ereignisses fehlerhaft geworden ist oder – und dies ist weitaus häufiger der Fall – wenn die neu eingelegte Diskette mit derselben ID formatiert worden ist wie die herausgenommene.

FORMAT DES INITIALIZE-BEFEHLS	BEISPIEL
PRINT#15,"INITIALIZE[laufwerk]"	**PRINT#15,"INITIALIZE0"**

oder in der abgekürzten Form:

PRINT#15,"I[laufwerk]"	**PRINT#15,"I0"**

Auch hier wird davon ausgegangen, daß der Befehlskanal mit Datei 15 geöffnet wurde und „laufwerk" gleich 0 ist.
INITALIZE wird auch bei der Verwendung von Reinigungsdisketten benötigt. Reinigungsdisketten sind jedoch unter normalen Umständen und bei normaler Reinigung und Pflege nicht erforderlich; werden sie jedoch dennoch verwendet, so genügt das folgende kurze Programm, um für die für den Reinigungsvorgang notwendige Rotation zu sorgen.

```
10 OPEN 15,8,15
20 FOR I=1 TO 15
30 PRINT#15,"I0"
40 NEXT I
50 CLOSE 15
```

Bei diesem Programm wird eine INITIALIZE-Schleife verwendet, um den Laufwerksmotor etwa 20 Sekunden lang eingeschaltet zu lassen.

3.2 BASIC 7.0

In diesem Kapitel werden die Diskettenbefehle beschrieben, die mit dem Commodore 128 Personal Computer (im C128-Modus) verwendet werden. Hier handelt es sich um BASIC 7.0, das Befehle von BASIC 2.0, BASIC 3.5 und BASIC 4.0 umfaßt, die alle verwendet werden können.

3.2.1 FEHLERABFRAGE

Wenn die Laufwerksanzeige (grüne Anzeige) blinkt, so können Sie die Fehlerursache mit folgendem Befehl ermitteln:

```
PRINT DS$
```

Unabhängig davon, ob ein Fehler vorhanden ist oder nicht, wird eine Meldung ausgegeben. Ist ein Fehler vorhanden, so wird er mit diesem Befehl gleichzeitig aus dem Diskettenspeicher gelöscht. Die Fehleranzeige beim Diskettenlaufwerk wird ausgeschaltet.
Sobald die Meldung auf dem Bildschirm angezeigt wird, können ihre Bedeutung und die fehlerbehebende Maßnahme in Anhang 9.2 nachgeschlagen werden.
Für die Benutzer, die eigene Programme schreiben, wird nachfolgend eine kleine Subroutine zur Fehlerabfrage angegeben, die in eigenen Programmen verwendet werden kann:

```
59990 REM FEHLER-KANAL LESEN
60000 IF DS>1 THEN PRINT DS$:STOP
60010 RETURN
```

Die Subroutine liest den Fehlerkanal und setzt die Ergebnisse in die reservierten Variablen DS und DS$. Sie werden automatisch von BASIC aktualisiert.
Zwei Fehlernummern bezeichnen keine Fehler: 0 bedeutet, daß alles in Ordnung ist, während 1 angibt, wie viele Dateien durch einen SCRATCH-Befehl, der später in

diesem Kapitel beschrieben wird, gelöscht wurden. Bei einem anderen Fehlerstatus wird die Fehlermeldung in Zeile 60000 ausgedruckt und das Programm angehalten. Da es sich hier um eine Subroutine handelt, kann mit dem BASIC-Befehl GOSUB entweder im Direkt-Modus oder aus einem Programm auf sie zugegriffen werden. Die RETURN-Anweisung in Zeile 60010 sorgt dafür, daß in den Direkt-Modus bzw. zur nächsten Anweisung zurückgesprungen wird.

3.2.2 SAVE

Mit diesem Befehl wird ein Programm in eine Datei abgespeichert, so daß es wieder benutzt werden kann. Die Diskette muß formatiert werden, bevor Dateien auf ihr abgespeichert werden können.

FORMAT DES SAVE-BEFEHLS

DSAVE "Dateiname" [,Dlaufwerk] [,Ugerät]

Dieser Befehl kann beim Kopieren von Programmen, die nicht in BASIC geschrieben wurden, nicht verwendet werden. Programme in Maschinensprache werden mit dem BSAVE-Befehl oder dem eingebauten Monitorbefehl S abgespeichert.

FORMAT DES BSAVE-BEFEHLS

BSAVE "Dateiname" [,Dlaufwerk] [,Ugerät] [Bbank]
,Pstartadresse TO Pendladresse+1

wobei die üblichen Optionen dieselben sind und "bank" einer der 16 Speicherbänke des C128 entspricht. Der abzuspeichernde Adreßbereich wird in Dezimalform angegeben. Hier wird darauf hingewiesen, daß die anzugebende Endadresse ein Byte hinter der letzten abzuspeichernden Stelle stehen muß.
Um das beim C128 und PLUS/4 implementierte Monitorprogramm aufzurufen, geben Sie MONITOR ein, und um es zu verlassen, geben Sie X ein und betätigen die RETURN-Taste.

FORMAT VON MONITOR SAVE

S"[laufwerk:]dateiname",gerät,startadresse,endadresse+1

„laufwerk" ist die Laufwerksnummer, bei der 1570/71 eine Null. „dateiname" ist ein gültiger Dateiname aus maximal 14 Zeichen (wobei zwei Zeichen für die Laufwerksnummer und den Doppelpunkt vorgesehen sind); „gerät" ist eine zweistellige

Gerätenummer(-adresse), bei der 1570/71 normalerweise 08 (die führende Null ist obligatorisch). Der abzuspeichernde Adreßbereich wird in Hexadezimalzahlen (Zahlen zur Basis 16) eingegeben, jedoch ohne führende Dollarzeichen (bei dem PLUS/4). Beim C128 brauchen die Adressen nicht unbedingt als Hexadezimalzahlen angegeben zu werden. Hier wird nochmals darauf hingewiesen, daß die einzugebende Endadresse eine Stelle hinter der letzten abzuspeichernden Stelle stehen muß.

3.2.2.1 SAVE WITH REPLACE

Ist eine Datei bereits vorhanden, so kann eine neue Datei nicht mehr unter demselben Namen abgespeichert werden, da das Diskettenlaufwerk einen bestimmten Dateinamen pro Diskette nur einmal zuläßt. Dieses Problem kann mit den RENAME- und SCRATCH-Befehlen umgangen werden, die später in diesem Kapitel beschrieben werden. Möchten Sie nur ein Programm oder eine Datei durch eine andere Version ersetzen, so ist ein anderer Befehl bequemer. Diese unter dem Namen SAVE WITH REPLACE oder @SAVE bekannte Option weise das Diskettenlaufwerk an, die Datei mit dem im Disketteninhaltsverzeichnis angegebenen Namen durch die neue Datei zu ersetzen.

FORMAT VON SAVE WITH REPLACE

DSAVE"@dateiname" [,Dlaufwerk] [,Ugerät]

Bei diesem Verfahren wird folgendermaßen vorgegangen: Die neue Version wird zunächst vollständig abgespeichert, danach wird die alte gelöscht und der Inhaltsverzeichniseintrag so geändert, daß er sich auf die neue Version bezieht. Auf diese Weise besteht nur eine geringe Gefahr, daß durch einen Stromausfall während des Verfahrens sowohl die alte als auch die neue Kopie der Datei zerstört werden. Mit der alten Kopie geschieht nichts, bevor nicht die neue Kopie richtig abgespeichert wurde.
Achtung: Verwenden Sie einen@SAVE-Befehl nicht auf einer beinahe vollen Diskette. Benutzen Sie ihn nur, wenn auf der Diskette ausreichend Platz für eine zweite vollständige Kopie des zu ersetzenden Programms vorhanden ist. Aufgrund der Arbeitsweise von @SAVE befinden sich zu einem bestimmten Zeitpunkt sowohl die alte als auch die neue Version der Datei gleichzeitig auf der Diskette. Dies stellt einen Schutz vor Programmverlust dar. Ist nicht genügend Platz auf der Diskette vorhanden, um die zweite Kopie aufzunehmen, so wird nur soviel von der neuen Version abgespeichert, wie Platz vorhanden ist. Nachdem der Befehl ausgeführt wurde, ist die neue Version im Inhaltsverzeichnis vorhanden; sie belegt jedoch nicht genügend Blöcke und stimmt somit mit der Kopie im Speicher nicht vollständig überein.

3.2.3 DVERIFY

Dieser Befehl vergleicht das im Speicher befindliche Programm Byte für Byte mit einem auf der Diskette abgespeicherten Programm. Dieser Vergleich schließt auch die BASIC-Zeilenpointer, die auf die nächste Zeile weisen, mit ein. Bei verschiedenen Speicherkonfigurationen können diese unterschiedlich sein. Dies bedeutet, daß ein mit dem C64 auf der Diskette abgespeichertes und in einen C128 geladenes Programm nicht richtig überprüft werden kann, da die Zeilenpointer auf andere Speicherpositionen zeigen. Unterscheidet sich die Diskettenkopie des Programms von dem Original im Speicher, so wird die Fehlermeldung „VERIFY ERROR" (Prüffehler) angezeigt. Dies bedeutet nicht unbedingt, daß ein abgespeichertes BASIC-Programm fehlerhaft ist. Wird jedoch davon ausgegangen, daß beide Versionen (im Speicher und in der Datei) 100%ig identisch sein müssen, so liegt in diesem Fall ein Problem vor.

FORMAT DES DVERIFY-BEFEHLS

DVERIFY "dateiname" [,Dlaufwerk] [,Ugerät]

Mit der folgenden Version des Befehls wird normalerweise eine gerade abgespeicherte Datei überprüft:

DVERIFY"*"

Dieser Befehl kann jedoch nach SAVE-WITH-REPLACE nicht verwendet werden, da es sich bei der zuletzt benutzten Datei um die gelöschte Datei handelt und das Laufwerk in diesem Fall versucht, die gelöschte Datei mit dem Programm im Speicher zu vergleichen. Daraus entsteht zwar kein Schaden, allerdings wird stets die Fehlermeldung "VERIFY ERROR" (Prüffehler) ausgegeben. Um VERIFY im Anschluß an @SAVE zu benutzen, geben Sie mindestens einen Teil des zu überprüfenden Dateinamens in Jokerzeichen (*) an.

3.2.4 COPY

Mit dem Copy-Befehl können Sie eine zusätzliche Kopie jeder Datei anfertigen. Bei einem Einzellaufwerk wie der 1570/71 muß die Kopie jedoch auf derselben Diskette erfolgen. Dies hat bei Einzellaufwerken auch zur Folge, daß ihr ein anderer Name zugewiesen werden muß. Die Ausgangsdatei und die anderen Dateien auf der Diskette werden nicht geändert. Die Dateien müssen abgeschlossen werden, bevor sie kopiert oder verkettet werden können.

FORMAT FÜR DEN COPY-BEFEHL

COPY [Dlaufwerk,]"alter dateiname"TO[Dlaufwerk,]"neuer dateiname" [,Ugerät]

Sofern die Laufwerksnummern angegeben werden, sind beide gleich Null.
HINWEIS: Soll eine Datei von einer Diskette auf eine andere kopiert werden, so kann der COPY-Befehl dazu nicht verwendet werden; zu diesem Zweck kann das Kopier-Programm auf der Test-/Demo-Diskette oder das DOS-SHELL-Programm (siehe Kapitel 4) benutzt werden.

3.2.5 CONCAT

Mit dem CONCAT-Befehl können zwei sequentielle Dateien verkettet (kombiniert) werden.

FORMAT DES CONCAT-BEFEHLS

CONCAT [Dlaufwerk,]"hinzuzufügende datei"TO[Dlaufwerk,]"masterdatei" [,Ugerät]

In beiden Fällen kann die Laufwerksnummer wahlweise mit dem Wert Null angegeben werden. Die alte „masterdatei" wird gelöscht und durch eine neue „masterdatei" ersetzt, bei der es sich um die Verkettung der alten „masterdatei" mit der „hinzuzufügenden datei" handelt.

HINWEIS: Die Länge einer Befehlsfolge (Befehl und Dateinamen) ist auf 41 Zeichen beschränkt.

3.2.6 SCRATCH

Mit dem SCRATCH-Befehl können nicht mehr benötigte Dateien von Disketten gelöscht werden und der von ihnen belegte Platz zur Nutzung durch andere Dateien freigegeben werden. Mit ihm können eine einzige oder auch mehrere Dateien gleichzeitig mit Hilfe von Jokerzeichen (*) gelöscht werden.

FORMAT DES SCRATCH-BEFEHLS

SCRATCH "dateiname" [,Dlaufwerk] [,Ugerät]

„dateiname" entspricht einem beliebigen gültigen Dateinamen.

Als Sicherheitsmaßnahme gibt das System folgende Meldung aus:

ARE YOU SURE?

Sind Sie sich sicher, so betätigen Sie einfach Y und die RETURN-Taste. Wenn nicht, betätigen Sie nur die RETURN-Taste oder geben eine andere Antwort ein. In diesem Fall wird der Befehl rückgängig gemacht.
Die Anzahl der gelöschten Dateien wird automatisch angezeigt. Enthält die Diskette beispielsweise Dateien namens „TEST", „TRAIN", „TRUCK" und „TAIL", so können Sie alle vier Dateien zusammen mit anderen Dateien, die mit dem Buchstaben „T" beginnen, mit Hilfe des folgenden Befehls löschen:

SCRATCH "T*"

Handelte es sich bei den vier aufgeführten Dateien um die einzigen Dateien, die mit „T" beginnen, so wird folgende Meldung ausgegeben:

01,FILES SCRATCHED,04,00

READY.

Mit „04" wird angegeben, daß vier Dateien gelöscht wurden.
Sie können einen SCRATCH-Befehl innerhalb eines Programms ausführen. In diesem Fall wird jedoch kein Bedienerhinweis angezeigt.

3.2.6.1 WEITERE INFORMATIONEN ÜBER SCRATCH

SCRATCH ist ein überaus leistungsfähiger Befehl und muß vorsichtig benutzt werden, damit nur die Dateien gelöscht werden, die wirklich gelöscht werden sollen. Wird dieser Befehl mit Jokern benutzt, so wird empfohlen, daß dieselben Joker zuerst auf einen DIRECTORY-Befehl angewandt werden, damit Sie genau wissen, welche Dateien gelöscht werden. Auf diese Weise gibt es keine unliebsamen Überraschungen, wenn dieselben Joker im SCRATCH-Befehl benutzt werden.

3.2.6.2 UNSCRATCH

Wird eine Datei versehentlich gelöscht, so besteht dennoch eine Möglichkeit, sie wiederherzustellen. Wie der BASIC-Befehl NEW löscht SCRATCH die Datei selbst nicht wirklich; er löscht nur die Zeiger zu der Datei im Disketteninhaltsverzeichnis. Auf der Test-/Demo-Diskette ist ein „Unscratch"-Programm vorhanden.

HINWEIS: Wird eine Datei versehentlich innerhalb des DOS SHELL (s. Kapitel 4) gelöscht, so kann dies mit der DOS-SHELL-Funktion RESTORE FILES wieder rückgängig gemacht werden.

3.2.6.3 NICHT-GESCHLOSSENE DATEIEN

Eine unvollständige Datei darf niemals mit SCRATCH gelöscht werden. Unvollständige Dateien werden in der Auflistung des Inhaltsverzeichnisses durch einen Stern (*) vor der Angabe des Dateityps markiert. Der Stern bedeutet, daß die Datei bei der Erstellung nicht korrekt abgeschlossen wurde und daß demzufolge keine gültige Kette von Sektorverknüpfungen vorhanden ist, der durch den SCRATCH-Befehl beim Löschen der Datei gefolgt werden könnte. Wird eine derartige Datei mit SCRATCH gelöscht, so besteht die Gefahr, daß Sektoren freigegeben werden, die noch von anderen Programmen oder Dateien benötigt werden. Dadurch entsteht ein ständiger Schaden für diese anderen Programme oder Dateien, wenn später weitere Dateien zu der Diskette hinzugefügt werden.
Stoßen Sie auf eine nicht richtig abgeschlossene Datei oder stellen Sie zu spät fest, daß Sie eine derartige Datei gelöscht haben, so überprüfen Sie die Diskette sofort mit dem COLLECT-Befehl, der später in diesem Kapitel beschrieben wird. Wurden seit dem Löschen der nicht richtig abgeschlossenen Datei dem Inhaltsverzeichnis weitere Dateien hinzugefügt, so wird empfohlen, die ganze Diskette sofort auf eine neue Diskette zu kopieren. Hierzu ist jedoch ein Dateikopierprogramm (file copy) und kein Sicherungsprogramm (backup copy) zu verwenden, oder es ergibt sich auch auf der neuen Diskette wieder das gleiche Problem. Sobald die neue Kopie angefertigt wurde, vergleichen Sie die Anzahl freier Blöcke in ihrem Inhaltsverzeichnis mit der Anzahl freier Blöcke auf der Originaldiskette. Stimmen die Zahlen überein, so ist wahrscheinlich kein Schaden entstanden. Wenn nicht, so wurde u. U. mindestens eine Datei auf der Diskette zerstört. In diesem Fall müssen alle Dateien sofort überprüft werden.

3.2.6.4 GESCHÜTZTE DATEIEN

Gelegentlich enthalten Disketten eine oder mehrere geschützte Dateien, d. h. solche, die nicht mit dem SCRATCH-Befehl gelöscht werden können. Derartige Dateien können anhand des „<"-Zeichens erkannt werden, das unmittelbar auf den Dateityp in dem Verzeichniseintrag folgt.
Soll eine gesperrte Datei gelöscht werden, so können Sie beispielsweise ein Diskettenmonitorprogramm verwenden, um Bit 6 des Dateityp-Bytes in dem Verzeichniseintrag auf der Diskette zu löschen. Um eine Datei zu sperren, muß Bit 6 des o. g. Bytes gesetzt werden.

3.2.7 RENAME

Mit dem RENAME-Befehl können Sie den Namen einer Datei im Disketteninhaltsverzeichnis ändern. Da nur das Inhaltsverzeichnis von diesem Befehl beeinflußt wird, wird RENAME sehr schnell ausgeführt. Versuchen Sie, eine Datei mit einem schon im Verzeichnis stehenden Dateinamen umzubenennen, so antwortet der Computer mit der Fehlermeldung „FILE EXISTS" (Datei vorhanden). Eine Datei muß richtig abgeschlossen werden, bevor sie umbenannt werden kann.

FORMAT DES RENAME-BEFEHLS

RENAME [Dlaufwerk,] "alter name" TO [Dlaufwerk,] "neuer name" [,Ugerät]

wobei beide Laufwerksnummern, sofern vorhanden, gleich Null sind.

3.2.8 UMBENENNEN UND LÖSCHEN PROBLEMATISCHER DATEIEN

Gelegentlich können Sie auf Dateien mit seltsamen Dateinamen stoßen, wie beispielsweise einem einzelnen Komma (",") oder einem Dateinamen mit geshifteten Leerzeichen. Möglicherweise stoßen Sie auch auf Dateien, die nicht-ausdruckbare Zeichen umfassen. Jede dieser Dateien kann zu einem Problem werden. So stellen beispielsweise Kommadateien eine Ausnahme zu der Regel, daß zwei Dateien nicht denselben Namen aufweisen dürfen, dar. Da es nicht möglich sein sollte, eine Datei zu erstellen, deren Name nur aus einem Komma besteht, geht das DOS davon aus, daß Sie dies auch nicht tun.
Dateinamen mit einem geshifteten Leerzeichen können sich ebenfalls als problematisch erweisen, da die Diskette das geshiftete Leerzeichen als das Ende des Dateinamens interpretiert und sämtliche Angaben ausdruckt, die auf das Anführungszeichen folgen, das das Ende eines Namens in dem Verzeichnis markiert. Mit dieser Technik können lange Dateinamen verwendet werden und die Diskette einen kleinen Teil dieses Namens als gleichbedeutend mit dem ganzen Namen erkennen, ohne daß Joker benutzt werden müssen.
Bei problematischen Dateinamen können Sie in jedem Fall die CHR$-Funktion benutzen, um die besonderen Zeichen indirekt einzugeben. Dadurch können sie in einen RENAME-Befehl eingebaut werden. Gelingt dies nicht, so können auch die Joker (*) benutzt werden, die schon beim SCRATCH-Befehl beschrieben worden sind. Dadurch kann der Name ohne die problematischen Zeichen angegeben werden. Allerdings führt dies zu einem Verlust der Datei:
Ist es Ihnen beispielsweise gelungen, eine Datei namens "MOVIES mit einem zusätzlichen Anführungszeichen vor dem Dateinamen zu erstellen, so können Sie

diese Datei mit dem CHR$-Äquivalent eines Anführungszeichens in dem RENAME-Befehl in "MOVIES" umbenennen.

Beispiel:

```
RENAME (CHR$(34)+"MOVIES") TO "MOVIES"
```

Mit der Funktion CHR$(34) wird ein Anführungszeichen in die Befehlsfolge gesetzt, ohne daß sich dies im BASIC auswirkt. Bei einem Dateinamen mit einem geshifteten Leerzeichen wird ebenso vorgegangen, allerdings wird die Funktion CHR$(160) benutzt.
In den Fällen, in denen auch dies nicht hilft, beispielsweise wenn die Diskette eine Kommadatei (eine Datei namens ",") enthält, so können Sie das Problem folgendermaßen lösen:

Beispiel:

```
SCRATCH"?"
```

Mit diesem Befehl werden sämtliche Dateien mit aus einem Zeichen bestehenden Namen gelöscht.
Je nach Problem muß bei der Auswahl der Joker, die sich nur auf die gewünschte Datei beziehen, sehr umsichtig vorgegangen werden. Unter Umständen müssen Sie andere Dateien zuerst umbenennen, damit sie nicht gelöscht werden.
In einigen Fällen ist es einfacher, die nicht benötigten Dateien auf eine andere Diskette zu kopieren und die problematischen Dateien auf der alten Diskette stehen zu lassen.

3.2.9 COLLECT

Mit dem COLLECT-Befehl wird die Block Availability Map (BAM) einer Diskette neu erstellt, wobei nur die Sektoren als belegt gekennzeichnet werden, die noch von gültigen, korrekt abgeschlossenen Dateien und Programmen benutzt werden. Alle anderen Sektoren (Blöcke) bleiben für die spätere Benutzung frei. Sämtliche nicht korrekt abgeschlossenen Dateien werden automatisch gelöscht. Aus dieser kurzen Beschreibung des COLLECT-Befehls ergeben sich weder Leistungsfähigkeit noch Gefahr dieses Befehls. Sein großer Vorteil besteht darin, daß er die Belegungstabelle vieler Disketten einwandfrei wiederherstellen kann. Immer wenn die Summe der von den Dateien auf einer Diskette benutzten Blöcke und der als frei angegebenen Blöcke keine 664 (im 1541-Modus und bei der 1570) oder 1328 (im 1571-Modus) verfügbaren Blöcke auf einer Diskette ergibt, sollte der

COLLECT-Befehl ausgeführt werden (mit der einen nachfolgend angegebenen Ausnahme). Wann immer eine Diskette eine nicht richtig abgeschlossene Datei enthält, die im Inhaltsverzeichnis durch einen Stern(*) vor dem Dateityp gekennzeichnet ist, sollte für diese Diskette ebenfalls ein COLLECT ausgeführt werden. Ausgenommen von den im folgenden Absatz beschriebenen Fällen sollte bei Disketten stets ein COLLECT durchgeführt werden, wenn Zweifel an der Richtigkeit ihrer Belegungstabelle (BAM) bestehen. Vor der Ausführung des COLLECT sollte die Anzahl freier Blöcke vom Disketteninhaltsverzeichnis festgehalten werden. Hat sich die Anzahl der freien Blöcke geändert, so war ein Problem vorhanden. Wenn die Anzahl der freien Blöcke kleiner geworden ist, muß die Diskette höchstwahrscheinlich Datei für Datei auf eine neue Diskette kopiert werden. Hierzu kann der im vorhergehenden Abschnitt beschriebene COPY-Befehl, ein Dateikopierungsprogramm (file copy) oder das DOS SHELL, welches im nächsten Kapitel beschrieben wird, verwendet werden; ein Sicherungsprogramm (backup copy) darf nicht benutzt werden, weil hierbei evtl. noch vorhandene Fehler mitkopiert werden.

Die Ausnahmen, bei denen der COLLECT-Befehl nicht angewendet werden darf, liegen in Disketten, auf denen sich Direktzugriffsdateien, wie sie in Kapitel 7 beschrieben werden, befinden. Die meisten Direktzugriffsdateien weisen ihre Sektoren nicht so zu, als daß COLLECT sie erkennen könnte. Wird bei einer derartigen Diskette ein COLLECT-Befehl ausgeführt, so kann dies zur Freigabe sämtlicher Blöcke der Direktzugriffsdateien führen, wobei deren Inhalt ganz oder teilweise verloren geht, sobald weitere Dateien auf der Diskette abgelegt werden. Werden vom Hersteller (Softwarehaus) solcher Disketten nicht ausdrücklich andere Angaben gemacht, darf für eine Diskette mit Direktzugriffsdateien niemals ein COLLECT-Befehl ausgeführt werden. Direktzugriffsdateien sind nicht gleichbedeutend mit relativen Dateien, wie sie in Kapitel 6 beschrieben werden. COLLECT kann bei relativen Dateien problemlos durchgeführt werden.

FORMAT DES COLLECT-BEFEHLS

COLLECT [Dlaufwerk] [,Ugerät]

3.2.10 INITIALIZE

Für die Initialisierung ist kein Befehl in BASIC 7.0 vorhanden. Für den BASIC-2.0-Befehl INITIALIZE wird deshalb auf Kapitel 3.1.9 verwiesen.

KAPITEL 4

DAS DOS SHELL

DOS SHELL ist ein Programm für den Commodore 128 Personal Computer, das eine Alternative zur Ausführung der Diskettenlaufwerksbefehle darstellt. Bedienerhinweise führen den Benutzer Schritt für Schritt durch jede Operation, so daß die Benutzung der 1570/71 einfacher und verständlicher wird.
Das DOS SHELL wird automatisch geladen, wenn die Diskette eingelegt und der Computer eingeschaltet bzw. zurückgesetzt wird. Das DOS SHELL kann durch Betätigung der F1-Taste gestartet bzw. verlassen werden.

4.1 AUSWAHL DER SPRACHE FÜR DIE BEDIENERFÜHRUNG

Das DOS SHELL zeigt seine Meldungen in einer von vier Sprachen an: Englisch, Französisch, Deutsch oder Italienisch. Sobald das DOS SHELL aktiviert wird, werden die einzelnen Sprachen nacheinander etwa 6 Sekunden lang angezeigt. Um eine Sprache auszuwählen, betätigen Sie die Leertaste, sobald die Sprache auf dem Bildschirm angezeigt wird. Sie können auch die Wahl beschleunigen, indem Sie die CURSOR-DOWN-Taste betätigen.

HINWEIS: DOS SHELL kann im 40- oder 80-Zeichen-Modus benutzt werden. Der einzige Unterschied besteht in der Änderung des Bildschirmlayouts. Das Bildschirmformat (40 oder 80 Zeichen/Zeile) kann umgeschaltet werden, indem zunächst die ESC- und dann die X-Taste betätigt wird. Dies muß geschehen, bevor eine Funktion aus dem Hauptmenü gewählt wird (s. u.).

4.2 DAS HAUPTMENÜ

Nachdem eine Sprache ausgewählt wurde, wird auf dem Bildschirm das Hauptmenü angezeigt. Zur Auswahl einer Funktion bewegen Sie den „Cursor" mit den CURSOR-Tasten (CRSR) zum entsprechenden Menüpunkt und betätigen anschließend die Leertaste.

4.2.1 LAUFWERKS-/DRUCKER-SETUP

Das DOS SHELL selbst arbeitet intern mit zwei logischen Laufwerken A und B. Sie können sie so definieren, daß sie physikalisch einem einzigen Diskettenlaufwerk, zwei einzelnen Diskettenlaufwerken oder einem Doppellaufwerk entsprechen. Angenommen, Sie verfügen über ein einziges Diskettenlaufwerk, so entsprechen die üblichen Einstellungen für A und B einer Geräteadresse von 8 und einer Laufwerksnummer von 0. Hierbei handelt es sich übrigens um die Standardeinstellungen; anders ausgedrückt, wird DOS SHELL zum ersten Mal aufgerufen, so entspricht das Laufwerks-Setup einem einzigen Diskettenlaufwerk. Bei zwei einzelnen Diskettenlaufwerken ist die Zuweisung der Geräteadresse an eines der logischen Laufwerke zu ändern. Die Laufwerksnummern sind bei Null zu belassen. Bei einem Doppellaufwerk wird die Zuweisung der Laufwerksnummer eines der beiden Laufwerke geändert.

Diese Zuweisungen können Sie durch Anwahl des Menüpunkts „DISK/DRUCKER SETUP" vornehmen, indem Sie im Hauptmenü den „Cursor" – das revers dargestellte Feld wird im folgenden als „Cursor" bezeichnet – ggf. mit Hilfe der Cursortasten auf diese Funktion bringen und mit der Leertaste anwählen.

Um eine Einstellung zu ändern, bewegen Sie den „Cursor" nun mit den CURSOR-Tasten auf den gewünschten Wert und betätigen die Leertaste. Nachdem Sie die Zuweisungen vorgenommen haben, betätigen Sie die F7-Taste, um sie festzuhalten. Sollten Sie Ihre Meinung ändern, so betätigen Sie anstelle der F7- die STOP-Taste. In diesem Fall werden die vorhergehenden Einstellungen unverändert belassen. Die gleiche Geräteadresse der Druckereinheit wird auf die gleiche Art und Weise ausgewählt.

Um die Geräteadresse des tatsächlichen Laufwerks (des physikalischen Laufwerks) zu ändern, bewegen Sie den „Cursor" auf die Funktion „AENDERE LAUFWERK#" und betätigen die Leertaste. Danach werden Meldungen auf dem Bildschirm ausgegeben, mit der die Änderung der Geräteadresse Schritt für Schritt beschrieben wird. Sollten Sie Ihre Meinung ändern, bevor die einzelnen Schritte beendet sind, so betätigen Sie die STOP- oder die F5-Taste, um die Änderung der Geräteadresse nicht durchzuführen. Bei Betätigung der F5-Taste werden wieder die zur Zeit gültigen Zuweisungen zwecks erneuter Änderung angezeigt, während man durch Betätigung der STOP-Taste wieder zum Hauptmenü zurückkehrt.

HINWEIS: Wird das Setup für eine einzige Disketteneinheit vorgenommen, so wird bei Auswahl einer der folgenden Funktionen eine Meldung auf dem Bildschirm ausgegeben, mit der Sie gefragt werden, welches Laufwerk benutzt werden soll. Bei der Beschreibung dieser Funktionen wird davon ausgegangen, daß Sie nur über eine einzige Disketteneinheit verfügen. Arbeiten Sie mit mehr als einem Laufwerk, so müssen Sie diese Frage mit den Cursor-Tasten und der Leertaste beantworten.

4.2.2 AUSFÜHRUNG EINES PROGRAMMS

Mit der Funktion „PROGRAMMSTART" wird eine Programmdatei, die aus einer auf dem Bildschirm angezeigten Dateiliste ausgewählt wird, automatisch geladen und ausgeführt. (Bei dieser Funktion enthält die Dateienliste nur Programmdateien.) Mit den CURSOR-Tasten bewegen Sie den Cursor zur gewünschten Datei und betätigen danach die Leertaste, um diese Datei auszuwählen. Sollten Sie Ihre Meinung ändern, so betätigen Sie die F5-Taste, um die Auswahl der Datei wieder rückgängig zu machen. Soll die Datei geladen und ausgeführt werden, so betätigen Sie die F7-Taste. Betätigen Sie die STOP-Taste, bevor das Laden beendet ist, so wird die Funktion rückgängig gemacht.

HINWEIS: Sie können die Dateiliste ausdrucken, sobald sie für eine Funktion angezeigt wird, indem Sie die F3-Taste betätigen. Vergewissern Sie sich, ob der Drucker eingeschaltet und das Papier so eingelegt ist, daß am Seitenanfang gedruckt werden kann.

4.2.3 FORMATIEREN EINER DISKETTE

Wird die Funktion „DISK FORMATIEREN" ausgewählt, so wird eine Meldung auf dem Bildschirm ausgegeben, mit der Sie aufgefordert werden, eine leere Diskette in das Laufwerk einzulegen. Nachdem dies geschehen ist, betätigen Sie die Leertaste. Das DOS SHELL prüft dann, ob die Diskette schon formatiert wurde. Wenn ja, wird der Diskettensatz auf dem Bildschirm angezeigt, falls schon Dateien auf der Diskette stehen. Um die Formatierung vorzunehmen, geben Sie entweder einen neuen Diskettennamen ein oder akzeptieren den alten Namen und betätigen die RETURN-Taste. Wurde die Diskette vorher schon formatiert, so behält sie ihren alten ID-Code. Wenn nicht, generiert das DOS SHELL einen neuen zufälligen ID-Code. Sie können Ihren eigenen ID-Code eingeben, indem Sie ein Komma und eine zweistellige Zahl als letzte drei Zeichen bei der Eingabe des Diskettennamens vor Betätigung der RETURN-Taste eingeben.

4.2.4 DISKETTE (BAM) ORDNEN

Mit der Funktion „BAM ORDNEN" können Sie die Block Availability Map einer Diskette neu erstellen. (Hier wird auf die Beschreibung des VALIDATE-Befehls bzw. COLLECT-Befehls in Kapitel 3 verwiesen.)

4.2.5 KOPIEREN EINER DISKETTE (BACKUP COPY)

Mit der Funktion „DISK KOPIEREN" können Sie Sicherungskopien (‚backup copy') Ihrer Disketten anfertigen. Mit dem Laufwerk-Setup (Laufwerke A und B) wird

bestimmt, auf welche Art die Kopie ausgeführt wird: mit einem Einzellaufwerk, einem Doppellaufwerk oder zwei Einzellaufwerken.

Kopie mit einem Doppellaufwerk oder zwei Einzellaufwerken
Das DOS SHELL fragt Sie, von welchem Laufwerk kopiert werden soll. (Mit den Cursor-Tasten und der Leertaste wählen Sie A oder B.) Legen Sie nun die Originaldiskette in dieses Laufwerk und die Diskette, auf die kopiert werden soll, in das andere Laufwerk ein und betätigen Sie die Leertaste. Soll diese Funktion rückgängig gemacht werden, so drücken Sie die STOP-Taste. Mit der F5- oder der Leertaste kann eine Diskettenoperation erneut ausgeführt werden, falls es zu einem Fehler kommen sollte.

Kopieren mit einem Einzellaufwerk
Das DOS SHELL zeigt die folgende Warnung an: „PROGRAMM IM SPEICHER WIRD GELOESCHT". Betätigen Sie die STOP- oder die F5-Taste, um die Funktion an dieser Stelle abzubrechen. Soll fortgefahren werden, so betätigen Sie die Leertaste.
Im Laufe dieser Funktion sind die Original- und die Sicherungsdiskette nach den Bedienungshinweisen des DOS SHELL einzulegen. Dabei müssen die Disketten ein- bis viermal ausgetauscht werden, ehe der Kopiervorgang beendet ist. Wird die Sicherungsdiskette das erste Mal eingelegt, so wird sie mit demselben Diskettennamen wie die Originaldiskette formatiert, allerdings wird ein anderer ID-Code generiert. Wird das nächste Mal eine Diskette eingelegt, so wird ihr ID-Code geprüft, um zu sehen, ob es sich um die richtige Diskette handelt. Wenn nicht, wird eine Fehlermeldung auf dem Bildschirm ausgegeben. In diesem Fall betätigen Sie die Leertaste oder die F5-Taste, zum zu letzten Bedienungshinweis für das Auswechseln der Disketten zurückzukehren.
Wird die STOP-Taste betätigt, so wird die Funktion abgebrochen, und der Kopiervorgang ist unvollständig.

4.2.6 KOPIEREN VON DATEIEN

Mit der Funktion „COPY FILES" werden Kopien ausgewählter Dateien angefertigt, wobei die Art des Kopierens vom Laufwerk-Setup bestimmt wird. Nachdem Sie das Laufwerk ausgewählt haben, von dem die Dateien kopiert werden sollen, wird die Dateiliste angezeigt, so daß Sie die Dateien mit den Cursor-Tasten und der Leertaste auswählen können. Mit den Cursor-Tasten können Sie die Liste nach oben oder unten „scrollen".
Sobald Sie den Cursor neben eine zu kopierende Datei positioniert haben, betätigen Sie die Leertaste. Sollten Sie Ihre Meiung ändern, so betätigen Sie die F5-Taste, um

die Auswahl der Datei wieder rückgängig zu machen. Nachdem Sie die zu kopierenden Dateien ausgewählt haben, betätigen Sie die F7-Taste, um die Kopierfunktion auszuführen. Danach wird eine Meldung auf dem Bildschirm ausgegeben, mit der Sie gefragt werden: „FILE-LISTE OK ZUM KOPIEREN: N J". Antworten Sie mit nein (Standardantwort), so kehrt das DOS SHELL wieder zu der Dateiauswahl zurück. Beantworten Sie die Frage mit ja, so wird der Kopiervorgang je nach benutztem Kopiertyp ausgeführt.

Kopieren mit einem Doppellaufwerk oder zwei Einzellaufwerken

Nachdem Sie die Frage „FILE-LISTE OK ZUM KOPIEREN: N J" beantwortet haben, werden die Meldungen „ZIEL-DISKETTE EINLEGEN IN LAUFWERK: x" und „DANN <SPACE> DRUECKEN" angezeigt. Nachdem dies geschehen ist, wird die Meldung „ZIEL-DISKETTE FORMATIEREN" angezeigt. Die Standardantwort lautet nein. Wird die Frage mit ja beantwortet, so beginnt die Funktion „DISK FORMATIEREN" an der Stelle, an der Sie aufgefordert werden, einen neuen Diskettennamen für das Formatieren einzugeben.

Unabhängig davon, ob die Frage für das Formatieren mit ja oder nein beantwortet wird, vergleicht das DOS-SHELL-Programm den auf der Kopiediskette zur Verfügung stehenden Platz mit der Gesamtgröße der ausgewählten Datei(en). Kann die Kopiediskette nicht sämtliche Dateien aufnehmen, so wird eine Warnmeldung angezeigt. In diesem Fall können Sie die STOP-Taste betätigen, um die Funktion rückgängig zu machen. Soll fortgefahren werden, so betätigen Sie die Leertaste.

HINWEIS: Paßt die als nächstes zu kopierende Datei nicht auf die Kopiediskette, so werden bei jedem Kopiertyp folgende Meldungen angezeigt: „ZUWENIG SPEICHERPLATZ FUER ALLE FILES" und „ZIEL-DISKETTE VOLL! WEITERE DISK: N J" Lautet die Antwort nein (Standardantwort), so wird der Kopiervorgang beendet. Wird die Frage mit ja beantwortet, so kehrt die Funktion wieder zu der Meldung „ZIEL-DISKETTE EINLEGEN IN LAUFWERK: x" zurück.

Kopieren mit einem Einzellaufwerk

Nachdem die Frage „FILE-LISTE OK ZUM KOPIEREN" beantwortet wurde, wird folgende Warnung auf dem Bildschirm angezeigt: „PROGRAMM IM SPEICHER WIRD GELOESCHT". Darauf folgt die Instruktion „WEITER MIT <SPACE>". Betätigen Sie die F5- oder die STOP-Taste, um die Funktion abzubrechen und das evtl. im Speicher befindliche Programm unverändert zu lassen.

Nun legen Sie den Bedienungshinweisen entsprechend auf dem Bildschirm abwechselnd die Original- und die Zieldiskette ein. Bis zum Abschluß des Kopiervorgangs müssen die Disketten ein- bis viermal ausgewechselt werden.

Wird die Zieldiskette das erste Mal eingelegt, so wird die Meldung „ZIEL-DISKETTE FORMATIEREN: N J" angezeigt. Unabhängig davon, ob die Antwort ja oder nein lautet, vergleicht das DOS SHELL auch hier wieder den auf der Zieldiskette zur

Verfügung stehenden Platz mit der Gesamtgröße der zu kopierenden Dateien. Ist nicht ausreichend Platz auf der Zieldiskette vorhanden, so wird eine Warnung ausgegeben.

4.2.7 LÖSCHEN VON DATEIEN

Die Funktion „FILES LOESCHEN" löscht eine oder mehrere Dateien. Mit den Cursor- und der Leertaste werden die Datei(en) aus der auf dem Bildschirm angezeigten Dateienliste ausgewählt. Nachdem die Datei(en) ausgewählt wurde(n), betätigen Sie die F7-Taste. In diesem Fall wird folgende Meldung ausgegeben: „FILE-LISTE OK ZUM LOESCHEN: N J". Lautet die Antwort nein (Standardantwort), so kehrt die Funktion zu der Dateiauswahl zurück. Lautet die Antwort ja, so werden die Dateien gelöscht.
Betätigen Sie die F5-Taste, um die Dateiauswahl noch einmal von Anfang an zu beginnen. (Hierbei werden alle bisherigen Auswahlen rückgängig gemacht.) Betätigen Sie die STOP-Taste, um die Funktion abzubrechen.

4.2.8 WIEDERHERSTELLEN VON DATEIEN (UNSCRATCH)

Mit der Funktion „FILES WIEDER HERST" können gelöschte Dateien wiederhergestellt werden. Auf dem Bildschirm wird eine Liste nur mit den gelöschten Dateien angezeigt. Auch hier wählen Sie mit den Cursor-Tasten und der Leertaste die wiederherzustellenden Dateien. Soll die Auswahl rückgängig gemacht werden, so betätigen Sie die F5-Taste.
Nachdem Sie eine Datei ausgewählt haben, prüft das DOS-SHELL, ob sie nicht bereits durch andere Dateien überschrieben worden ist. In diesem Fall wird die Meldung „WIEDERHERSTELLEN UNMOEGLICH: dateiname" angezeigt. Ist alles in Ordnung, so wird die Dateiauswahl fortgesetzt.
Während die einzelnen Daten ausgewählt werden, wird folgende Meldung ausgegeben: „WAEHLE FILE-TYP: SEQ PRG USR". Mit den Cursor-Tasten und der Leertaste treffen Sie die Auswahl. Relative Dateien werden automatisch erkannt, so daß Sie diesen Typ nicht anzugeben brauchen.
Sobald die Dateiauswahl beendet ist, betätigen Sie die F7-Taste. Daraufhin wird die Meldung „FILE-LISTE OK ZUM WIEDERHERSTELLEN: N J" angezeigt. Lautet die Antwort nein (Standardantwort), so kehrt die Funktion zu Dateiauswahl zurück, lautet die Antwort ja, so werden die Dateien wiederhergestellt.

4.2.9 UMBENENNEN VON DATEIEN

Mit der Funktion „FILES UMBENENNEN" können Dateinamen geändert werden. Mit den Cursor-Tasten und der Leertaste wird eine umzubenennende Datei ausgewählt. Die Meldung „EINGABE NEUER NAME:" wird zusammen mit dem bisherigen Dateinamen angezeigt. Nun müssen Sie den neuen Namen eingeben oder die F5-Taste betätigen, um zur Dateiauswahl zurückzukehren.
Jeder neue Dateiname wird auf Einmaligkeit geprüft. Ist der Name nicht einmalig, so wird die Meldung „FEHLER! FILE NAME DOPPELT BITTE WIEDERHOLEN" angezeigt. Die Funktion kehrt nach Betätigung der Leertaste zu der Meldung „EINGABE NEUER NAME:" zurück.
Nachdem die Dateiauswahl beendet ist, betätigen Sie die F7-Taste. Danach wird die Meldung „FILE-LIST OK ZUM UMBENENNEN: N J" angezeigt. Lautet die Antwort nein, so kehren Sie wieder zu dem Verfahren der Dateiauswahl zurück. Lautet die Antwort ja, so werden die Dateinamen geändert. Betätigen Sie die F5-Taste, so wird die Funktion nicht ausgeführt, und das Verfahren der Dateiauswahl beginnt noch einmal von Anfang an.

4.2.10 SORTIEREN DES INHALTSVERZEICHNISSES

Mit der Funktion „DIRECTORY ORDNEN" kann die Reihenfolge geändert werden, in der die Dateinamen im Disketteninhaltsverzeichnis eingetragen sind. Nachdem diese Funktion ausgewählt wurde, wird folgende Frage auf dem Bildschirm ausgegeben: „DIRECTORY ALPHABETISCH ORDNEN: N J". Die Standardantwort lautet nein. Wird die Frage mit ja beantwortet, so werden die Dateinamen automatisch alphabetisch neu geordnet.
Soll das Inhaltsverzeichnis manuell neu geordnet werden, so wählen Sie eine Datei mit den Cursor-Tasten und der Leertaste aus und ziehen danach den Dateinamen mit Hilfe der Cursor-Tasten durch die Dateiliste an seine neue Position. Betätigen Sie anschließend die Leertaste, um den Dateinamen dort einzutragen.
Nachdem die F7-Taste betätigt wurde, um die Dateiauswahl zu beenden, wird die Meldung „DIRECTORY ZURUECKSCHREIBEN: N J" angezeigt. Lautet die Antwort ja, so wird die neu angeordnete Dateiliste in das Diskettenverzeichnis geschrieben. Lautet die Antwort nein (Standardantwort), so wird das Verfahren der Dateiauswahl erneut bei der Meldung „DIRECTORY ALPHABETISCH ORDNEN: N J" gestartet, wobei alle zuvor vorgenommenen Änderungen belassen werden.
Wird die F5-Taste betätigt, so wird das Verfahren der Dateiauswahl ebenfalls an dieser Stelle begonnen, ohne daß jedoch die zuvor vorgenommenen Änderungen berücksichtigt werden.

KAPITEL 5

SEQUENTIELLE DATEIEN

5.1 ORGANISATION VON DATEIEN

Eine Datei auf einer Diskette kann gewissermaßen mit einem Aktenschrank in einem Büro verglichen werden – eine Stelle, an der Dinge auf bestimmte Art und Weise abgelegt werden. Nahezu alles, was auf eine Diskette geschrieben wird, wird in einer bestimmten Datei abgespeichert. Bis jetzt wurden nur Programmdateien benutzt. Es gibt jedoch auch noch andere Dateien. In diesem Kapitel werden sequentielle Dateien beschrieben.
Eine Datei wird im wesentlichen zur Speicherung irgendwelcher Daten, beispielsweise der Werte von Programmvariablen benutzt, damit diese bei Beendigung des Programms nicht verlorengehen. Eine sequentielle Datei ist eine Datei, in der die einzelnen Daten „sequentiell", d. h. direkt hintereinander, abgespeichert sind. Möglicherweise sind Sie von der DATASSETTE® her bereits mit sequentiellen Dateien vertraut, da sequentielle Dateien auf einer Diskette mit den Dateien auf Kassette verglichen werden können.
Unabhängig davon, ob die Daten auf Kassette oder Diskette stehen, müssen sequentielle Dateien von Anfang an gelesen werden.
Werden sequentielle Dateien erstellt, so werden die Informationen (Daten) Byte für Byte über einen Puffer auf den magnetischen Datenträger übertragen. Sequentielle Dateien sind die am häufigsten verwendeten Dateien; auch Programm- und Userdateien sind sequentiell organisiert. Selbst das Inhaltsverzeichnis wird wie eine sequentielle Datei behandelt.
Um sequentielle Dateien richtig verwenden zu können, werden wir auf den nächsten Seiten noch einige weitere BASIC-Schlüsselwörter vorstellen. Sie werden dann in einem einfachen, aber nützlichen Programm kombiniert.

HINWEIS: Neben sequentiellen Dateien werden noch zwei andere Dateitypen sequentiell auf einer Diskette aufgezeichnet. Hier handelt es sich um Programmdateien und Userdateien. Wird ein Programm auf eine Diskette abgespeichert, so wird es folgerichtig von Anfang an bis zum Ende abgespeichert, genau wie die Informationen in einer sequentiellen Datei. Der Hauptunterschied besteht in den für den Zugriff auf die Datei verwendeten Befehlen. Userdateien können eher mit sequentiellen Dateien verglichen werden. Userdateien werden nahezu nie benutzt, könnten jedoch wie sequentielle Dateien behandelt werden.

Für erfahrene Benutzer bietet die Ähnlichkeit der verschiedenen Dateitypen die Möglichkeit, eine Programmdatei Byte für Byte (d. h. Zeichen für Zeichen) in den Computer einzulesen und in geänderter Form wieder auf die Diskette zu schreiben.

5.2 ÖFFNEN EINER DATEI

Eine der leistungsfähigsten Anweisungen bei Commodore-BASIC ist die OPEN-Anweisung. Mit ihr können Daten nahezu an jede beliebige Stelle gesendet werden. Ein Befehl, der all dies kann, ist natürlich recht komplex. OPEN-Anweisungen wurden schon in einigen Diskettenbefehlen verwendet.
Bevor wir uns dem Format der OPEN-Anweisung zuwenden, wollen wir uns die Peripherie in einem Commodore-Computersystem betrachten:

Geräteadresse	Einheit	Verwendungszweck
0	Tastatur	Empfängt die Eingabe vom Bediener des Computers.
1	DATASSETTE®	Sendet und empfängt Informationen von einer Datenkassette.
2	RS232C	Sendet und empfängt Informationen von einem Modem.
3	Bildschirm	Ausgabe auf Bildschirm.
4,5	Drucker	Ausgabe auf Drucker.
6	Printer/Plotter	Ausgabe auf Printer/Plotter.
8,9,10,11	Diskettenlaufwerke	Sendet und empfängt Informationen von der Diskettenstation.

Aufgrund der Flexibilität der OPEN-Anweisung kann eine einzige Programmanweisung Verbindung mit jeder dieser Einheiten oder sogar anderen Einheiten aufnehmen, je nach Wert eines einzelnen Zeichens in dem Befehl. Steht das Zeichen in einer Variablen, so kann bei jeder Benutzung dieses Programmteils eine andere Einheit verwendet werden, so daß Daten abwechselnd und problemlos an Diskette, Kassette, Drucker und Bildschirm gesendet werden können.

5.3 FEHLERABFRAGE

In Kapitel 3 wurde beschrieben, wie nach Abgabe von Diskettenbefehlen eine Fehlerabfrage durchgeführt wird. Diese Fehlerabfrage ist nach der Benutzung der Anweisungen zur Dateiverarbeitung sehr wichtig. Wird ein Diskettenfehler nicht

entdeckt, bevor eine andere Anweisung zur Verarbeitung von Dateien benutzt wird, so kann es zu einem nicht bemerkten Datenverlust kommen.
Die einwandfreie Befehlsabarbeitung wird zweckmäßigerweise geprüft, indem im Anschluß an jede Anweisung zur Dateiverarbeitung eine GOSUB-Anweisung zu einer Fehlerprüfroutine ausgeführt wird.

BEISPIEL:

BASIC 7.0:

```
840 DOPEN#4,"TAGESDATEN",D0,U8,W
850 GOSUB 59990:REM TEST AUF DISK-FEHLER
```

BASIC 2.0:

```
840 OPEN4,8,4,"TAGESDATEN",S,W
850 GOSUB 59990:REM TEST AUF DISK-FEHLER
```

FORMAT DER OPEN-ANWEISUNG FÜR SEQUENTIELLE DATEIEN:

BASIC 7.0
DOPEN#datei,"dateiname" [,Dlaufwerk] [,Ueinheit] [,W]

BASIC 2.0
OPEN datei,einheit,kanal,"[laufwerk:]dateiname[,dateityp][,richtung]"

„datei" ist eine ganze Zahl zwischen 1 und 255. Öffnen Sie keine Diskettendatei, deren Dateinummer größer ist als 127, da es ansonsten zu schwerwiegenden Problemen kommt. Nachdem die Datei geöffnet wurde, beziehen sich alle anderen Dateibefehle mit der hier angegebenen Nummer auf diese Datei. Nur eine Datei kann jeweils eine bestimmte Dateinummer benutzen.
„einheit" ist die Geräteadresse (Primäradresse) des zu verwendenden Geräts. Bei dieser Adresse handelt es sich um eine ganze Zahl im Bereich von 8 bis 11. Bei einer 1570/71 wird normalerweise der Wert 8 benutzt.
„kanal" ist eine Sekundäradresse, die der ausgewählten Einheit weitere Instruktionen darüber erteilt, wie weitere Befehle ausgeführt werden müssen. Bei der Verarbeitung von Diskettendateien wählt die Kanalnummer einen bestimmten Kanal, über den die Kommunikation für diese Datei vorgenommen werden kann. Der mögliche Bereich der Kanalnummern liegt zwischen 0 und 15, 0 ist jedoch für das Laden von Programmen, 1 für das Abspeichern von Programmen und 15 für den Befehls- und Fehlerkanal reserviert. Zwei verschiedene Diskettendateien dürfen nicht gleichzeitig mit der gleichen Kanalnummer geöffnet werden; es empfiehlt sich, für die Datei- und die Kanalnummer den gleichen Wert zu wählen.

„laufwerk" ist die Laufwerksnummer. Bei der 1570/71 ist sie immer gleich 0. Sie sollte mit angegeben werden, weil sonst nur 2 anstelle von 3 Kanälen gleichzeitig benutzt werden können. Soll eine schon vorhandene Datei überschrieben werden, so setzen Sie vor die Laufwerksnummer das „at"-Zeichen @, um OPEN-WITH-REPLACE anzufordern.
„dateiname" entspricht dem aus maximal 16 Zeichen bestehenden Dateinamen. In dem Namen sind Joker zulässig, wenn auf bestehende Dateien zugegriffen werden soll. Dies gilt jedoch nicht, wenn neue Dateien erstellt werden sollen.
Mit „dateityp" wird der gewünschte Dateityp angegeben: S = sequentiell, P = Programm, U = User, A = Anfügen und L = Länge einer relativen Datei.
Mit „richtung" wird die gewünschte Zugriffsart angegeben. Hier gibt es drei Möglichkeiten: R = Lesen, W = Schreiben und M = Ändern. Beim Erstellen einer Datei wird „W" (write) benutzt, um die Daten auf die Diskette zu schreiben. Soll die fertige Datei gelesen werden, so wird „R" (read) benutzt. Die Option „M" (modify) wird nur als letzte Möglichkeit verwendet, um Daten von einer nicht richtig abgeschlossenen Datei zurückzulesen. Wird dies versucht, so muß jedes Byte beim Lesen überprüft werden, um sicherzustellen, daß die Daten noch gültig sind, da derartige Dateien fehlerhafte Daten enthalten und kein richtiges Ende aufweisen.
„dateityp" und „richtung" brauchen nicht abgekürzt zu werden. Sie können (englisch) voll ausgeschrieben werden, um die Listings leserlicher zu gestalten.
Bei den Parametern „datei", „einheit" und „kanal" muß es sich um gültige numerische Konstanten, Variablen oder Ausdrücke handeln. Die weiteren Parameter sind durch eine Zeichenfolge, eine(n) String-Variable oder -Ausdruck anzugeben.
„W" ist die Option, die angegeben werden muß, wenn in die sequentielle Datei geschrieben werden soll. Wird „W" nicht angegeben, so wird die Datei zum Lesen geöffnet.
Maximal zehn Dateien einschließlich der von allen anderen Peripheriegeräten können gleichzeitig geöffnet sein. Bei den sequentiellen Diskettendateien dürfen maximal drei Dateien (oder zwei, wenn die Laufwerksnummer in dem OPEN-Befehl nicht angegeben wird) und der Befehlskanal gleichzeitig geöffnet sein.

BEISPIELE FÜR DAS ÖFFNEN SEQUENTIELLER DATEIEN:

Um eine sequentielle Datei mit Telefonnummern zu erstellen, könnten Sie folgende Befehle benutzen:

```
BASIC 7.0: DOPEN#2,"TELEFON",D0,U8,W
BASIC 2.0: OPEN 2,8,2,"0:TELEFON,SEQUENTIAL,WRITE"
```

oder

```
OPEN 2,8,2,"0:TELEFON,S,W"
```

Falls Sie schon eine „TELEFON"-Datei auf der Diskette haben, können Sie die Fehlermeldung „FILE EXISTS" (Datei vorhanden) umgehen, indem Sie einen @ OPEN-Befehl ausführen:

```
BASIC 7.0: DOPEN#2,"@:TELEFON",D0,U8,W
BASIC 2.0: OPEN 2,8,2,"@:TELEFON,S,W"
```

Mit diesem Befehl werden alle alten Telefonnummern gelöscht. Deshalb vergewissern Sie sich, daß die gelöschten Informationen nicht mehr benötigt werden. Nachdem Sie die Telefondatei geschrieben haben, nehmen Sie die Diskette heraus und schalten das System aus. Um die Daten in der Datei wieder einzulesen, öffnen Sie die Datei beispielsweise mit folgendem Befehl:

```
BASIC 7.0: DOPEN#8,"TELEFON",D0,U8
BASIC 2.0: OPEN 8,8,8,"0:TELEFON,S,R"
```

Hier spielt es keine Rolle, ob die Datei- und Kanalnummern mit den vorher benutzten Nummern übereinstimmen. Der Dateiname muß jedoch übereinstimmen. Hier kann eine abgekürzte Form des Dateinamens benutzt werden, wenn keine anderen Dateien, auf die dieselbe Abkürzung zutrifft, vorhanden sind:

```
BASIC 7.0: DOPEN#10,"TEL*",D0,U8
BASIC 2.0: OPEN 10,8,6,"0:TEL*,S,R"
```

Sind zu viele Telefonnummern abzuspeichern, so dauert der Zugriff auf die Daten evtl. sehr lang. In diesem Fall benutzen Sie mehrere ähnliche Dateinamen und wählen die richtige Datei mit einem Programm aus.

```
BASIC 7.0:
100 INPUT "WELCHE TELEFON-DATEI(1-3)";PH
110 IF PH<>1 AND PH<>2 AND PH<>3 THEN 100
120 DOPEN#2 "TELEFON"+STR$(PH),D0,U8

BASIC 2.0:
100 INPUT "WELCHE TELEFON-DATEI(1-3)";PH
110 IF PH<>1 AND PH<>2 AND PH<>3 THEN 100
120 OPEN 4,8,2, "TELEFON"+STR$(PH)+",S,R"
```

Soll eine Datei gelesen werden, so kann die Laufwerksnummer im OPEN-Befehl weggelassen werden. In diesem Fall können die Besitzer von Doppellaufwerken beide Disketten nach der Datei absuchen.

5.4 ERWEITERN EINER SEQUENTIELLEN DATEI

Mit dem APPEND-Befehl können Sie eine bestehende sequentielle Datei erneut öffnen und weitere Informationen am Ende dieser Datei anfügen. Anstelle der Parameter „typ" und „richtung" in der OPEN-Anweisung benutzen Sie ",A" für Append. Dadurch wird die Datei erneut geöffnet und der Schreib-/Lesekopf an das Ende der Datei gesetzt. Damit können weitere Daten zu den bereits vorhandenen ergänzt werden.

FORMAT DER APPEND-OPTION

BASIC 7.0: APPEND#datei,"dateiname" [,Dlaufwerk] [,Ugerät]
BASIC 2.0: OPEN datei, einheit, kanal, "[laufwerk:] dateiname,A"

Die Parameter entsprechen den Parametern auf der letzten Seite, mit Ausnahme des abschließenden „A", das an die Stelle der Parameter „typ" und „richtung" tritt.

BEISPIEL:

Wird beispielsweise ein Programm zur Erfassung und Auswertung von Zensuren geschrieben, so wäre es einfach, die neuen Noten jedes Schülers an das Ende der schon abgespeicherten Noten anzufügen. Um beispielsweise Daten zu der Datei „JOHN PAUL JONES" hinzuzufügen, geben Sie folgende Befehle ein:

BASIC 7.0: APPEND#1,"0:JOHN PAUL JONES",D0,U8
BASIC 2.0: OPEN 1,8,3,"0:JOHN PAUL JONES,A"

In diesem Fall weist das Disk Operating System (DOS) der Datei beim ersten Anfügen weiterer Informationen mindestens einen weiteren (Block) zu, selbst wenn nur ein Zeichen hinzugefügt wird. Unter Umständen werden Sie auch feststellen, daß die Dateigröße durch Benutzung des COLLECT- oder VALIDATE-Befehls nicht korrigiert wurde. Wird der verschwendete Platz zu einem Problem, so kann dies problemlos behoben werden, indem Sie die Datei auf dieselbe Diskette oder eine andere Diskette kopieren und die Originaldatei löschen. Nachfolgend eine Reihe von Befehlen, mit denen derartige Dateien unter dem Originalnamen auf die Originaldiskette kopiert werden:

```
RENAME "JOHN PAUL JONES" TO "TEMP"
COPY "TEMP" TO "JOHN PAUL JONES"
SCRATCH "TEMP"
```

5.5 SCHREIBEN VON DATEIDATEN – DER PRINT#-BEFEHL

Nachdem eine sequentielle Datei zum Schreiben (mit Typ und Richtung ",S,W") geöffnet worden ist, senden wir die Daten mit dem PRINT#-Befehl zur Diskettenstation, um sie auf der Diskette zu speichern. Sind Sie mit der BASIC-Anweisung PRINT vertraut, so werden Sie feststellen, daß PRINT# genau auf dieselbe Art und Weise zu verwenden ist; der einzige Unterschied liegt darin, daß die auf das Befehlswort folgende Parameterliste nicht an den Bildschirm, sondern an ein Peripheriegerät ausgegeben wird, selbst die Formatierzeichen (Komma, Semikolon, TAB, SPC) haben die gleiche Bedeutung wie bei den PRINT-Anweisungen.
So wird beispielsweise ein Komma zwischen Variablen in einer PRINT-Anweisung als Trennzeichen in Bildschirmanzeigen benutzt. Auf diese Weise werden sämtliche Elemente in dem jeweils nächsten vorher festgelegten Anzeigefeld ausgegeben (typischerweise die nächste Spalte, deren Nummer voll durch 10 dividiert werden kann). Wird dasselbe Komma zwischen Variablen für eine Diskettendatei angegeben, so wird es wieder als Trennzeichen benutzt und fügt wieder zusätzliche Leerzeichen in die Daten ein. Diesmal ist das Komma jedoch nicht sinnvoll, da die zusätzlichen Leerzeichen auf der Diskette verschwendet werden und zu weiteren Problemen führen können, wenn die Datei wieder in den Computer zurückgelesen werden soll. Deshalb muß das nachfolgend angegebene Format genau eingehalten werden, wenn Daten in eine Diskettendatei gesendet werden.

FORMAT DES PRINT#-BEFEHLS:

PRINT#datei,datenliste

wobei „datei" dieselbe Dateinummer darstellt, die in der aktuellen OPEN-Anweisung für die gewünschte Datei angegeben wurde. Wird auf eine bestimmte Datei zugegriffen, so muß die Dateinummer konstant bleiben, da mit ihr alle anderen Befehle zur Verarbeitung der Datei auf die richtige OPEN-Anweisung zurückgeführt werden können. Wird eine Dateinummer angegeben, so kann der Computer alle anderen wichtigen Angaben über die Datei finden.
Die „datenliste" ist dieselbe wie bei einer PRINT-Anweisung – eine Liste mit Konstanten, Variablen bzw. Ausdrücken, einschließlich Zahlen, Zeichenfolgen oder beidem. Es ist jedoch besser, wenn jede PRINT#-Anweisung für eine Diskette nur ein Datenelement umfaßt. Sollen mehrere Elemente gesendet werden, so müssen diese durch ein RETURN und nicht durch ein Komma voneinander getrennt werden. Semikola sind zulässig, werden jedoch in der Datei nicht aufgezeichnet und führen nicht zu zusätzlichen Leerzeichen. Mit ihnen werden Elemente in der Liste getrennt,

die ansonsten verwechselt werden könnten, wie beispielsweise eine Stringvariable, die unmittelbar auf eine numerische Variable folgt.

HINWEIS: Zwischen PRINT und # darf kein Leerzeichen stehen. Der Befehl darf auch nicht mit ?# abgekürzt werden; die richtige Abkürzung für PRINT# lautet pR.

BEISPIELE:

Um beispielsweise einige Zensuren von John Paul Jones in einer sequentiellen Diskettendatei abzuspeichern, die vorher zum Schreiben geöffnet wurde, werden folgende Befehle verwendet:

```
200 FOR KL=1 TO KU
210 PRINT#1,ZE(KL)
220 GOSUB 59990:REM TEST AUF DISK-FEHLER

230 NEXT KL
```

Hier wird davon ausgegangen, daß das Programm eine Subroutine zur Fehlerprüfung enthält, wie im letzten Kapitel angegeben.
Bei der Benutzung von PRINT# gibt es eine Ausnahme zu der erforderlichen Fehlerabfrage nach jeder Anweisung zur Dateiverarbeitung. Wird PRINT# benutzt, so kann ein Fehler nach dem Schreiben einer ganzen Gruppe von Daten mit einer Prüfung entdeckt werden, solange die Prüfung vorgenommen wird, bevor eine andere Anweisung zur Dateiverarbeitung oder ein anderer Diskettenbefehl benutzt wird. Unter Umständen sind Sie schon mit PRINT-Anweisungen vertraut, in denen verschiedene Elemente aufeinander folgen:

```
400 PRINTNA$;ST$;OT$
```

Um nun diese Variablen in eine sequentielle Diskettendatei mit der Nummer 5 und nicht auf den Bildschirm zu setzen, werden am besten drei separate PRINT#-Anweisungen wie folgt benutzt:

```
400 PRINT#5,NA$:REM NAME
410 PRINT#5,ST$:REM STRASSE
420 PRINT#5,OT$:REM ORT
```

Sollen diese Befehle kombiniert werden, so geschieht dies am sichersten wie folgt:

```
400 PRINT#5,NA$;CHR$(13);ST$;CHR$(13);OT
$
```

CHR$(13) ist das Zeilenschaltungszeichen (RETURN). Es wirkt sich genau so aus,

als würden die Druckelemente auf separate Zeilen gesetzt. Benutzen Sie dieses Verfahren häufig, so kann Platz und Zeit gespart werden, indem vorher einer Variablen der Wert CHR$(13) zugewiesen wird:

```
10 CR$=CHR$(13)
400 PRINT#5,NA$;CR$;ST$;OT$
```

Wurden Daten richtig in eine sequentielle Datei auf die Diskette geschrieben und werden diese dann auf den Bildschirm umgeleitet, so wird nämlich nur ein Datenelement pro Zeile angezeigt, wobei nachfolgende Elemente auf einer jeweils neuen Zeile stehen.

5.6 SCHLIESSEN EINER DATEI

Nachdem eine Datei nicht mehr benutzt wird, muß sie unbedingt mit CLOSE abgeschlossen werden. Während des Schreibens einer Datei werden Daten in einem Puffer des Speichers zwischengespeichert und erst auf die Diskette geschrieben, wenn der Puffer voll ist.
Bei dieser Arbeitsweise ist nahezu stets eine kleine Datenmenge im Puffer vorhanden, die noch nicht auf die Diskette geschrieben wurde und die verloren geht, wenn das Computersystem ausgeschaltet wird. Darüber hinaus gibt es Aufgaben der Diskettenverwaltung, wie beispielsweise die Aktualisierung der BAM (Block Availability Map – Belegungstabelle), der von der aktuellen Datei benutzten Sektoren, die während des normalen Schreibens einer Datei nicht ausgeführt werden. Aus diesem Grund ist eine CLOSE-Anweisung erforderlich. Sobald eine Datei fertig eingegeben wurde, sorgt die CLOSE-Anweisung für das Schreiben der restlichen Daten im Datenpuffer auf die Diskette, die Aktualisierung der BAM und den vollständigen Dateieintrag in dem Inhaltsverzeichnis.
Eine Datei muß immer geschlossen werden, wenn sie nicht mehr benutzt werden soll. Geschieht dies nicht, so kann es zu einem Verlust der ganzen Datei kommen. Der Diskettenbefehlskanal darf jedoch erst geschlossen werden, nachdem alle anderen Dateien geschlossen worden sind. Der Befehlskanal muß stets als erster geöffnet und als letzter in jedem Programm geschlossen werden.

FORMAT DER CLOSE-ANWEISUNG

BASIC 7.0: DCLOSE#datei [,Ugerät]
BASIC 2.0: CLOSE datei

wobei „datei" der Dateinummer entspricht, die in der aktuellen OPEN-Anweisung für die gewünschte Datei angegeben wurde.

BEISPIELE

Um die Datendatei 5 des Beispiels auf der vorhergehenden Seite abzuschließen, wird folgender Befehl benutzt:

BASIC 7.0: DCLOSE#5
BASIC 2.0: CLOSE 5

Wird bei BASIC 7.0 die DCLOSE-Anweisung ohne Parameter verwendet, so werden alle Diskettendateien durch diesen einzelnen Befehl geschlossen. Mit ein wenig mehr Aufwand kann der gleiche Zweck durch eine Programmschleife erreicht werden. Da kein Schaden entsteht, wenn eine nicht geöffnete Datei geschlossen wird, schließen Sie einfach jede Datei, von der Sie glauben, daß sie geöffnet ist, bevor Sie ein Programm beenden. Haben Sie Ihren Dateien stets Nummern zwischen 1 und 5 zugewiesen, so können Sie sie alle mit folgenden Befehlen abschließen:

```
9950 FOR I=1 TO 5
9960 CLOSE I
9970 GOSUB 59990:REM TEST AUF DISK-FEHLE
R
9980 NEXT I
```

Hierbei wird davon ausgegangen, daß das Programm eine Subroutine zur Fehlerprüfung enthält, wie im letzten Kapitel angegeben.

5.7 LESEN VON DATEIDATEN MIT DEM INPUT#-BEFEHL

Nachdem Informationen korrekt in eine Diskettendatei geschrieben worden sind, können sie mit einer INPUT#-Anweisung wieder in den Computer eingelesen werden. Genau wie die PRINT#-Anweisung der PRINT-Anweisung sehr ähnelt, ist INPUT# vom Gebrauch her nahezu identisch mit INPUT. Allerdings wird die Parameterliste hinter dem Befehlswort normalerweise aus einer Datei gelesen und nicht über die Tastatur eingegeben. Beide Anweisungen unterliegen denselben Einschränkungen, d. h. die Eingabe wird im Anschluß an ein Komma oder einen Doppelpunkt angehalten, Datenelemente, die für den BASIC-Eingabepuffer zu groß sind, werden nicht akzeptiert, und nicht-numerische Daten sind in einer numerischen Variablen nicht zulässig.

FORMAT DER INPUT#-ANWEISUNG

INPUT#datei,variablenliste

wobei „datei" dieselbe Dateinummer ist, wie die in der für die gewünschte Datei angegebenen aktuellen OPEN-Anweisung. Bei der „variablenliste" handelt es sich um einen oder mehrere gültige BASIC-Variablennamen. Soll mehr als ein Datenelement von einer INPUT#-Anweisung eingelesen werden, so sind die einzelnen Variablennamen durch Kommata voneinander zu trennen.

BEISPIELE:

Um die in dem o. g. Beispiel mit PRINT# abgespeicherten Zensuren wieder in den Computer einzulesen, können Sie folgende Befehle verwenden:

```
200 FOR KL=1 TO KU
210 INPUT #1,ZE(KL)
220 GOSUB 59990:REM TEST AUF DISK-FEHLER
230 NEXT KL
```

Auch hier wird davon ausgegangen, daß das Programm eine Subroutine zur Fehlerabfrage enthält, wie im letzten Kapitel angegeben.
Um die Adressen, die in dem anderen Beispiel für PRINT# abgespeichert worden sind, wieder in den Computer einzulesen, können zweckmäßigerweise folgende Befehle verwendet werden:

```
800 INPUT#5,NA$
810 GOSUB 59990:REM TEST AUF DISK-FEHLER

820 INPUT#5,ST$
830 GOSUB 59990:REM TEST AUF DISK-FEHLER

840 INPUT#5,OT$
850 GOSUB 59990:REM TEST AUF DISK-FEHLER
```

Bei vielen Programmen wird jedoch nicht so viel Wert auf Sicherheit gelegt und daher nur folgende Anweisungen verwendet:

```
800 INPUT#5,NA$,ST$,OT$
810 GOSUB 59990:REM TEST AUF DISK-FEHLER
```

Dieses abgekürzte Verfahren wird in erster Linie benutzt, wenn der Einlesevorgang in möglichst kurzer Zeit vor sich gehen soll und es unwahrscheinlich ist, daß falsche Daten gelesen werden könnten.

5.8 WEITERE INFORMATIONEN ÜBER DEN INPUT#-BEFEHL

5.8.1 GRENZEN DES INPUT#-BEFEHLS

Sobald Sie häufiger mit Dateien arbeiten, werden Sie auf zwei BASIC-Fehlermeldungen stoßen, nämlich „STRING TOO LONG ERROR" (Zeichenfolge zu lang) und „FILE DATA ERROR" (Fehler in den Dateidaten). Bei beiden Fehlermeldungen wird das Programm wahrscheinlich bei einer INPUT#-Anweisung abgebrochen. Diese Fehler können durch Fehler in einer PRINT#-Anweisung beim Schreiben der Datei verursacht worden sein.

5.8.2 FEHLER „STRING TOO LONG"

Eine BASIC-Zeichenfolge kann maximal 255 Zeichen umfassen, obwohl die längste Zeichenfolge, die mit einer einzigen INPUT-Anweisung eingegeben werden kann, etwas weniger als zwei Textzeilen umfaßt. Diese Begrenzung ist auf die Größe des Eingabepuffers in den Commodore-Computern mit seriellem Bus zurückzuführen. Dieselbe Einschränkung gilt auch für INPUT#-Anweisungen. Enthält ein einzelnes Datenelement (Zeichenfolge oder Zahl), das von einer Diskettendatei mit einer INPUT#-Anweisung eingelesen wird, mehr als 88 (BASIC 2) bzw. 160 (BASIC 7) Zeichen, so bricht BASIC die Abarbeitung mit der Fehlermeldung „STRING TOO LONG ERROR" ab.

5.8.3 FEHLER „FILE DATA"

Die Fehlermeldung „FILE DATA ERROR" wird ausgegeben, wenn versucht wird, ein nicht-numerisches Zeichen in eine numerische Variable einzulesen. Bei der Eingabe eines numerischen Werts dürfen die Ziffern 0 bis 9, die Vorzeichen „+" und „−", der Dezimalpunkt „.", das Leerzeichen „ " und der Buchstabe „E" zur Angabe von Exponenten zur Basis 10 verwendet werden. Wird ein anderes Zeichen bei einer INPUT#-Anweisung für eine numerische Variable eingelesen, so wird die Fehlermeldung „FILE DATA ERROR" angezeigt und das Programm abgebrochen. Dieser Fehler wird im allgemeinen dadurch verursacht, daß die Reihenfolge, in der die Variablen in eine Datei geschrieben worden sind und dann wieder eingelesen werden, nicht übereinstimmt. Weitere Fehlermöglichkeiten bestehen in einem fehlenden RETURN beim Schreiben mehrerer Datenelemente in die Datei oder beim Auftreten von Kommata und Doppelpunkten, die in einem String auftreten, der nicht in Anführungszeichen eingeschlossen ist. Wenn ein Fehler in den Dateidaten aufgetreten ist, müssen Sie ihn korrigieren, indem Sie das Datenelement in eine Stringvariable einlesen und mit der BASIC-VAL-Anweisung wieder in eine Zahl

umsetzen, nachdem die nicht-numerischen Zeichen mit den Funktionen zur Stringverarbeitung, die in dem Benutzerhandbuch für den Computer beschrieben werden, entfernt sind.

5.8.4 KOMMATA UND DOPPELPUNKTE

Wie bereits erwähnt, können Kommata und Doppelpunkte Schwierigkeiten beim Einlesen verursachen, da sie das Datenelement beenden, in dem sie stehen. In diesem Fall werden die auf das Komma oder den Doppelpunkt folgenden Zeichen dieses Datenelements in die nächste INPUT#-Variable eingelesen. Sie haben in einer INPUT-Anweisung dieselbe Auswirkung und führen dort zu der Fehlermeldung „EXTRA INGNORED". Gelegentlich wird jedoch wirklich ein Komma oder ein Doppelpunkt innerhalb eines Datenelements benötigt, wie beispielsweise in einem Namen, der als „Nachname, Vorname" geschrieben wird. Das Problem wird in diesem Fall umgangen, indem diesen Datenelementen ein Anführungszeichen vorangesetzt wird. Im Anschluß an ein Anführungszeichen in einer INPUT- oder INPUT#-Anweisung werden alle anderen Zeichen mit Ausnahme eines weiteren Anführungszeichens und RETURN als Teil des aktuellen Datenelements akzeptiert.

BEISPIELE:

Um ein Anführungszeichen in ein Datenelement zu setzen, das in einer Datei gespeichert werden soll, setzen Sie CHR$(34) vor das Datenelement hinzu.

Beispiel:

```
PRINT#2,CHR$(34)+"MUELLER,FRITZ"
```

oder

```
PRINT#2,CHR$(34);"MUELLER,FRITZ"
```

Wird diese Möglichkeit häufig benutzt, so können Zeit und Platz gespart werden, indem vorher einer Variablen der Wert CHR$(34) zugewiesen wird, wie bereits zuvor bei CHR$(13) geschehen:

```
20 QT$=CHR$(34)
30 REM ....
390 REM ....
400 PRINT#5,QT$+NA$
```

In beiden Fällen wird das hinzugefügte Anführungszeichen von der INPUT- oder INPUT#-Anweisung aus den Daten entfernt, das Komma bzw. der Doppelpunkt bleiben jedoch Bestandteil der Daten.

5.9 ABSPEICHERN NUMERISCHER DATEN AUF DISKETTE

Bis jetzt haben wir uns mit der Speicherung von Zeichenfolgen befaßt. Nun wollen wir uns der Speicherung numerischer Daten zuwenden.
In dem Computer hängt der von einer numerischen Variablen belegte Platz ausschließlich vom Typ der Variablen ab. Einfache numerische Variablen belegen sieben Bytes (Zeichenpositionen) im Speicher. Echte Feldvariablen belegen 5 Bytes pro Feldelement und ganzzahlige Feldelemente belegen jeweils 2 Bytes. Wird dagegen eine numerische Variable oder ein beliebiger Typ in eine Datei geschrieben, so hängt der belegte Platz ausschließlich von deren Länge und nicht von ihrem Typ ab. Dies ist darauf zurückzuführen, daß numerische Daten in Form von Zeichenfolgen in eine Datei geschrieben werden, so als wäre die STR$-Funktion für diese Daten ausgeführt worden. Das erste Zeichen ist ein Leerzeichen, wenn die Zahl positiv ist. Ist die Zahl negativ, so ist das erste Zeichen ein Minuszeichen „−". Darauf folgt die Zahl Ziffer für Ziffer. Das letzte Zeichen ist ein Zeichen für die Cursorbewegung nach rechts.
In diesem Format können die Daten auf der Diskette später wieder in eine Stringvariable oder in eine numerische Variable eingelesen werden. Allerdings ist dieses Verfahren sehr speicherplatzaufwendig, und der von Zahlen mit unbekannter Länge benötigte Platz kann nur schwer geschützt werden. Aus diesem Grund setzen einige Programme alle numerischen Variablen in Zeichenfolgen um, bevor sie sie auf Diskette schreiben. Mit Stringfunktionen werden nicht benötigte Zeichen gelöscht. Diese Datenelemente können dennoch später wieder mit INPUT# in eine numerische Variable eingelesen werden. Fehler bei den Dateidaten können jedoch vermieden werden, indem alle Daten als Zeichenfolgen eingelesen und mit der VAL-Funktion in Zahlen umgesetzt werden, nachdem die Informationen im Computer stehen.
So setzt beispielsweise N$=RIGHT$(STR$(N),LEN(STR$(N))−1) eine positive Zahl N in eine Zeichenfolge N$ ohne das übliche führende Leerzeichen für das numerische Vorzeichen um. Anstatt PRINT#5,N zu schreiben, benutzen Sie nun PRINT#5,N$.

5.10 LESEN VON DATEIDATEN MIT DEM GET#-BEFEHL

Die GET#-Anweisung liest ein Byte (Zeichen) von der Diskettenstation ein. Wie der GET-Befehl zur Tastaturabfrage wird mit dieser Anweisung nur ein einzelnes Zeichen erfaßt. Im Gegensatz zur GET-Anweisung wird jedoch nicht bei der nächsten Anweisung fortgefahren, wenn keine Daten geholt werden können. GET# wird im wesentlichen dazu benutzt, Daten von einer Diskette zu lesen, die nicht mit der INPUT#-Anweisung verarbeitet werden können, entweder weil sie für den Eingabepuffer zu lang sind oder weil sie problematische Zeichen umfassen.

FORMAT DER GET#-ANWEISUNG:

GET#datei,variablenliste

„datei" entspricht der Dateinummer, die in der aktuellen OPEN-Anweisung für die gewünschte Datei angegeben wurde. Bei „variablenliste" handelt es sich um einen oder mehrere gültige BASIC-Variablennamen. Soll mehr als ein Datenelement mit einer GET#-Anweisung eingelesen werden, so sind die einzelnen Variablennamen durch Kommata voneinander zu trennen.
In der Praxis werden Sie wahrscheinlich nie auf eine GET- oder GET#-Anweisung stoßen, die mehr als einen Variablennamen umfaßt. Soll mehr als ein Zeichen eingelesen werden, so wird einer GET#-Anweisung in einer Schleife gegenüber einer GET#-Anweisung mit mehreren Parametern der Vorzug gegeben. Wie bei der INPUT#-Anweisung ist es sicherer, Stringvariablen zu verwenden, wenn die zu lesende Datei unter Umständen ein nicht-numerisches Zeichen enthält.
Die Daten in einer GET#-Anweisung werden Byte für Byte eingelesen, einschließlich der normalerweise unsichtbaren Zeichen, wie RETURN und den verschiedenen Steuerzeichen für den Cursor. Alle Zeichen, mit Ausnahme eines einzigen, werden richtig gelesen. Die Ausnahme ist CHR$(0), das ASCII-Null-Zeichen. Es unterscheidet sich von einer leeren Zeichenfolge (Leerstrings, in der Form A$=""), auch wenn Leerstrings häufig als Null-Strings bezeichnet werden. Bedauerlicherweise wird CHR$(0) in einer GET#-Anweisung in einen Leerstring umgesetzt. Das Problem wird dadurch gelöst, daß im Anschluß an eine GET#-Anweisung eine Abfrage auf einen Leerstring durchgeführt wird und eingelesene Leerstrings durch CHR$(0) ersetzt werden. Diese Methode wird in dem folgenden ersten Beispiel erläutert.

BEISPIELE:

Um eine Datei zu lesen, die ein CHR$(0)-Zeichen enthält, wie beispielsweise eine Programmdatei in Maschinensprache, lassen sich CHR$(0)-Bytes folgendermaßen korrigieren:

```
1100 GET#3, G$ : IF G$="" THEN G$=CHR$(0)
```

Wurde in einer Datei ein überlanger String abgespeichert, der nicht mit dem INPUT#-Befehl eingelesen werden kann (STRING TOO LONG), so kann er mit GET# mit Hilfe folgender Schleife wieder sicher in den Computer eingelesen werden:

```
3300 B$=""
3310 GET#1,A$
3320 IF A$<>CHR$(13) THEN B$=B$+A$:GOTO
3310
```

Bei einer derartigen Technik dürfen maximal 255 Zeichen zusammengesetzt werden. CHR$(0) wird ignoriert. Dies kann jedoch beim Erstellen einer Textfolge von Vorteil sein. Wird CHR$(0) in der Datei benötigt, so ist folgende Zeile zu verwenden:

```
3320 IF A$<>CHR$(13) THEN B$=B$+LEFT$(A$
+CHR$(0),1):GOTO 3310
```

GET#kann bei der Wiederherstellung fehlerhafter Dateien oder von Dateien mit unbekanntem Inhalt besonders nützlich sein. Mit der reservierten BASIC-Variablen ST (der Datei-Status-Variablen) kann festgestellt werden, wann eine richtig abgeschlossene Datei vollständig gelesen ist.

```
500 GET#2,S$
510 SU=ST :REM  DATEISTATUS MERKEN
520 PRINT S$;
530 IF SU=0 THEN 500 :REM  KEIN FEHLER A
UFGETRETEN UND DATEIENDE NICHT ERREICHT
540 IF SU<>64 THEN PRINT "STATUSFEHLER:
ST =";SU
```

Das Kopieren von ST in SU ist eine häufig nicht erforderliche Vorsichtsmaßnahme, muß jedoch vorgenommen werden, wenn eine andere Anweisung zur Dateiverarbeitung zwischen der Anweisung, mit der Daten aus der Datei gelesen werden, und der,

die zum erneuten Lesen zurückspringt, steht. Dies wäre beispielsweise erforderlich, wenn Zeile 520 folgendermaßen geändert würde:

```
520 PRINT#1,S$;
```

Würde der Status nicht in die Variable SU gerettet, so entspräche der in Zeile 530 überprüfte Dateistatus dem der Schreibdatei und nicht der Lesedatei.
Die folgende Tabelle bezieht sich auf einzelne Fehler. Treten mehrere Fehler gleichzeitig auf, so wird die Summe der Werte gebildet. (Jedem Bit des Statusbytes entspricht ein Fehler.)

MÖGLICHE WERTE DER STATUSVARIABLEN „ST" UND DEREN BEDEUTUNG

ST	BEDEUTUNG
0	Alles ist in Ordnung
1	Das Empfangsgerät war nicht verfügbar
2	Sendegerät nicht verfügbar (Time out)
4	Zu kurzer Datenblock in Kassettendatei
8	Zu langer Datenblock in Kassettendatei
16	Nicht behebbarer Lesefehler von der Kassette, Prüffehler
32	Prüfsummenfehler bei der Kassette – ein oder mehrere falsche Bytes wurden gelesen
64	Dateiende erreicht (EOI erkannt)
128	Gerät nicht vorhanden oder Bandendemarke auf der Kassette gefunden

Demoprogramm zur Arbeit mit sequentiellen Dateien
Verwenden Sie die folgenden Programme für Ihre ersten Experimente mit sequentiellen Dateien.

```
150 CR$=CHR$(13)
160 OPEN 15,8,15
170 PRINTCHR$(147):REM BILDSCHIRM LOESCH
EN
190 PRINT"** DATEI SCHREIBEN **"
210 PRINT
220 OPEN 2,8,2,"@0:SEQ FILE,S,W"
230 GOSUB 500
240 PRINT "GEBEN SIE EIN WORT, DANN EINE
NUMMER EIN"
250 PRINT "ODER 'ENDE,0' UM ZU BEENDEN"
260 PRINT
270 INPUT A$,B
280 PRINT#2,A$;CR$;B
290 GOSUB 500
300 IF A$<>"ENDE" THEN 270
310 PRINT
320 CLOSE 2
340 PRINT "** DIE GLEICHE DATEI ZURUECKL
ESEN **"
360 PRINT
370 OPEN 2,8,2,"0:SEQ FILE,S,R"
380 GOSUB 500
390 INPUT#2,A$,B
400 RS=ST
410 GOSUB 500
420 PRINT A$,B
430 IF RS=0 THEN 390
440 IF RS<>64 THEN PRINT "STATUS=";RS
450 CLOSE 2
455 CLOSE 15
460 END
480 REM ** DISK-FEHLER-TEST **
500 INPUT#15,EN,EM$,ET,ES
510 IF EN>0 THEN PRINTEN,EM$,ET,ES:STOP
520 RETURN
```

Demoprogramm in BASIC 7.0

```
150 CR$=CHR$(13)
170 PRINTCHR$(147):REM BILDSCHIRM LOESCH
EN
190 PRINT"** DATEI SCHREIBEN **"
210 PRINT
220 DOPEN#2,"@SEQ FILE",W
230 GOSUB 500
240 PRINT "GEBEN SIE EIN WORT, DANN EINE
NUMMER EIN"
250 PRINT "ODER 'ENDE,0' UM ZU BEENDEN"
260 PRINT
270 INPUT A$,B
280 PRINT#2,A$;CR$;B
290 GOSUB 500
300 IF A$<>"ENDE" THEN 270
310 PRINT
320 DCLOSE#2
340 PRINT "** DIE GLEICHE DATEI ZURUECKL
ESEN **"
360 PRINT
370 DOPEN#2,"SEQ FILE"
380 GOSUB 500
390 INPUT#2,A$,B
400 RS=ST
410 GOSUB 500
420 PRINT A$,B
430 IF RS=0 THEN 390
440 IF RS<>64 THEN PRINT "STATUS=";RS
450 DCLOSE#2
460 END
480 REM ** DISK-FEHLER-TEST **
500 IF DS>0 THEN PRINTDS$:STOP
510 RETURN
```

KAPITEL 6

RELATIVE DATEIEN

6.1 VORTEILE DES RELATIVEN ZUGRIFFS

Sequentielle Dateien sind sehr praktisch, wenn nur mit einem kontinuierlichen Datenstrom gearbeitet wird, d. h. mit Informationen, die als ganzes nacheinander gelesen oder geschrieben werden können. In einigen Situationen sind sequentielle Dateien jedoch nicht nützlich. Nachdem beispielsweise eine umfangreiche Liste mit Adressen geschrieben wurde, ist es nicht erforderlich, die ganze Liste erneut zu lesen, wenn nur der Eintrag für eine Person benötigt wird. Hier ist eine Art direkter Zugriff erforderlich, eine Möglichkeit, zu einer bestimmten Adresse in der Datei zu gelangen, ohne alle davorstehenden Einträge lesen zu müssen.
Als Beispiel vergleichen Sie einen Plattenspieler mit einem Kassettenrecorder. Die Kassette müssen Sie von Anfang bis zum Ende hören, wenn Sie eine bestimmte Stelle suchen, bei einer Schallplatte kann jedoch die Nadel jederzeit abgehoben und sofort an eine andere Stelle auf der Platte gesetzt werden. Das Diskettenlaufwerk kann in dieser Hinsicht mit einem Plattenspieler verglichen werden. In diesem Kapitel wird ein Dateityp beschrieben, der diese Flexibilität auch aufweist.
Tatsächlich können zwei verschiedene Arten von Direktzugriffsdateien bei Commodore Diskettenlaufwerken benutzt werden: relative Dateien und Direktzugriffsdateien. Relative Dateien sind für die meisten Datenverarbeitungsoperationen wesentlich bequemer. Für den fortgeschrittenen Benutzer stehen jedoch auch echte Befehle für Direktzugriffsdateien zur Verfügung. Sie werden im nächsten Kapitel beschrieben.

6.2 DATEIEN, DATENSÄTZE UND DATENFELDER

Bei den sequentiellen Dateien brauchen Sie sich keine Gedanken um die Organisation der Daten innerhalb einer Datei zu machen, solange die für das Schreiben der Datei benutzten Variablen mit den Variablen übereinstimmen, die die Daten wieder in den Computer zurücklesen. Damit der relative Zugriff jedoch durchgeführt werden kann, benötigen Sie eine strukturierte und vorhersehbare Umgebung für die Daten. Die benutzte Struktur kann mit der Struktur in normalen Aktenschränken verglichen werden. In einem normalen Büro werden beispielsweise sämtliche Kundenkarteien in einem einzigen Aktenschrank aufbewahrt. Innerhalb dieses Schrankes hat jeder Kunde einen eigenen Ordner mit seinem Namen, in dem sämtliche über diesen Kunden bekannten Angaben enthalten sind. Innerhalb jedes Ordners können Zettel

abgeheftet sein, die weitere Informationen über den Kunden enthalten, wie beispielsweise seine private Telefonnummer oder der Zeitpunkt seines letzten Auftrags.
In einem mit Computern ausgestatteten Büro gehört der Aktenschrank der Vergangenheit an. Das Prinzip einer Kartei mit sämtlichen Informationen über eine bestimmte Gruppe oder ein bestimmtes Thema bleibt jedoch bestehen. Die Aktenordner gehören auch der Vergangenheit an, die Unterteilung der Kartei in einzelne Einträge bleibt jedoch, nun als Datensätze, ebenfalls bestehen. Weiterhin gibt es keine Zettel in den einzelnen Einträgen mehr, sie werden durch Unterteilungen innerhalb der Einträge ersetzt, die als Felder bezeichnet werden. Jedes Feld ist groß genug, um eine Angabe über einen Datensatz der Datei aufzunehmen. Innerhalb jeder Datei sind viele Datensätze vorhanden, während jeder Datensatz typischerweise mehrere Felder umfaßt.
Eine relative Datei übernimmt die Organisation der Datensätze für Sie, indem diese von 1 bis zur höchsten Datensatznummer in Einerschritten durchnumeriert werden. Die Felder müssen jedoch vom Benutzer organisiert werden. Jeder Eintrag weist dieselbe Größe auf, allerdings brauchen bei der 1570/71 die Datensätze nicht alle auf dieselbe Art und Weise unterteilt zu werden, was jedoch normalerweise üblich ist. Ist im voraus genau bekannt, wo jedes Feld innerhalb jedes Datensatzes beginnt, so kann sogar schnell auf ein bestimmtes Feld innerhalb eines Datensatzes zugegriffen werden, ohne die anderen Felder lesen zu müssen. Daraus ergibt sich, daß die hohe Zugriffsgeschwindigkeit der Hauptgrund für die Benutzung relativer Diskettendateien ist. Gute relative Dateiprogramme können den Eintrag für eine gewünschte Person in weniger als 15 Sekunden aus tausend Einträgen herausfinden und lesen, eine Eigenschaft, die kein sequentielles Dateiprogramm hat.

6.3 GRENZEN BEI DER VERWENDUNG RELATIVER DATEIEN

Bei relativen Dateien brauchen Sie sich keine Gedanken darüber zu machen, an welcher Stelle genau auf der Diskettenoberfläche ein bestimmter Eintrag abgespeihert wird, ob er komplett in den aktuellen Diskettensektor paßt oder auf den nächsten verfügbaren Sektor erweitert werden muß. Diese Funktionen werden komplett vom DOS übernommen. Sie brauchen nur anzugeben, wie lang jeder Datensatz ist und wie viele Datensätze benötigt werden. Das DOS übernimmt den Rest und organisiert die Datensätze so, daß sie schnell in der Datei gefunden werden können, solange die richtige Datensatznummer (Position innerhalb der Datei) angegeben wird.
Sie brauchen nur darauf zu achten, daß die Datensätze dieselbe Größe aufweisen, und daß die gewählte Datensatzlänge zwischen 2 und 254 Zeichen liegt. Natürlich muß die ganze Datei auch auf die Diskette passen. Je mehr Datensätze benötigt werden, desto kürzer muß jeder Eintrag sein.

6.4 ERSTELLEN EINER RELATIVEN DATEI

Wird eine relative Datei das erste Mal benutzt, so wird die Datei durch die OPEN-Anweisung aufgebaut. Danach wird die Datei mit der OPEN-Anweisung erneut zum Lesen und Schreiben geöffnet.

FORMAT-ANWEISUNG ZUM ÖFFNEN EINER RELATIVEN DATEI:

BASIC 7.0:
DOPEN# datei, "dateiname", Lsatzlänge [,Dlaufwerk] [,Ugerät]
BASIC 2.0:
OPEN datei, gerät, kanal "[laufwerk:]dateiname, L,"+CHR$(satzlänge)

„datei" ist die Dateinummer, normalerweise eine ganze Zahl zwischen 1 und 127. „gerät" ist die Geräteadresse der zu verwendenden Diskettenstation, bei der 1570/71 normalerweise 8. Mit „kanal" wird ein bestimmter Kanal ausgewählt, über den die Kommunikation für diese Datei stattfinden kann; hier wird normalerweise eine Nummer zwischen 2 und 14 gewählt. „laufwerk" entspricht der Laufwerksnummer, bei der 1570/71 stets 0. „dateiname" entspricht dem Namen der Datei, der maximal 16 Zeichen umfaßt. Joker sind in dem Namen zulässig, wenn auf eine bestehende Datei zugegriffen wird, jedoch nicht, wenn eine neue Datei erstellt wird. Die „satzlänge" entspricht der Größe (in Bytes) jedes Datensatzes innerhalb der Datei einschließlich RETURNs, Anführungszeichen und anderen Sonderzeichen.
Hinweise:

1. Vor dem Dateinamen (bei BASIC 7.0) oder der Laufwerksnummer (bei BASIC 2.0) darf kein „at"-Zeichen @ stehen. Es gibt keinen Grund für das Ersetzen einer relativen Datei.
2. Die Angabe Lsatzlänge (bei BASIC 7.0) oder L,"+CHR$(satzlänge) (bei BASIC 2.0) ist nur erforderlich, wenn die relative Datei das erste Mal erstellt wird. Die Angabe kann zwar später auch benutzt werden, solange dieselbe Datensatzlänge wie beim Erstellen der Datei angegeben wird. Da relative Dateien völlig problemlos abwechselnd gelesen oder geschrieben werden können, braucht der Lese- oder Schreib-Modus beim Öffnen einer relativen Datei nicht angegeben zu werden.
3. Bei „datei", „gerät" und „kanal" muß es sich um gültige numerische Konstanten, Variablen oder Ausdrücke handeln. Der Rest des Befehls ist durch eine gültige Zeichenfolge (String), Stringvariable oder Stringausdruck anzugeben. Bei BASIC 7.0 müssen die Variablen bzw. Ausdrücke, die als Parameter von DOPEN gelten sollen, in Klammern eingeschlossen werden.
4. Bei der 1570/71 können nur eine relative und eine sequentielle Datei sowie der Befehlskanal gleichzeitig geöffnet sein. Sind jedoch eine sequentielle und eine

relative Datei gleichzeitig geöffnet, so kann kein Inhaltsverzeichnis angefordert werden.

BEISPIELE:

Um eine relative Datei namens „GRADES" mit der Satzlänge 100 zu erstellen oder erneut zu öffnen, ist folgender Befehl zu verwenden:

BASIC 7.0: DOPEN#2, "GRADES",L100,D0,U8
BASIC 2.0: OPEN 2,8,2,"GRADES,L,"'+CHR$(100)

Um eine schon erstellte unbekannte relative Datei nach Wahl des Benutzers erneut zu öffnen, werden folgende Befehle verwendet:

BASIC 7.0:
```
200 INPUT "WELCHE DATEI";FI$
210 DOPEN#5,(FI$),D0,U8
```

BASIC 2.0:
```
200 INPUT "WELCHE DATEI";FI$
210 OPEN 5,8,5,FI$
```

6.5 DER WAHLFREIE ZUGRIFF MIT DEM RECORD#-BEFEHL

Wird eine relative Datei das erste Mal geöffnet, so ist sie noch nicht vollständig zur Benutzung bereit. Sowohl um Zeit bei der späteren Benutzung der Datei zu sparen, als auch um eine einwandfreie Funktion der Datei zu gewährleisten, müssen verschiedene Einträge erstellt werden, bevor die Datei zum erstenmal geschlossen wird. Hierbei werden mindestens so viele Datensätze geschrieben, um zwei Diskettensektoren (512 Bytes) zu füllen. In der Praxis gehen die meisten Programme weiter und erstellen so viele Datensätze, wie das Programm wahrscheinlich benutzen wird. Diese Methode hat darüber hinaus den Vorteil, daß Probleme vermieden werden, wie wenn beispielsweise nicht genügend Platz für die gesamte Datei auf der Diskette vorhanden ist, bevor die Datei beendet ist.
Werden die ersten Daten in eine gerade erst geöffnete relative Datei geschrieben, so verhält sich diese wie eine sequentielle Datei, indem die mit der ersten PRINT#-Anweisung geschriebenen Datenelemente in Datensatz 1, die mit der zweiten PRINT#-Anweisung geschriebenen Elemente in Datensatz 2, usw. gesetzt werden. Dies bedeutet, daß jeder Eintrag mit einer einzigen PRINT#-Anweisung geschrieben werden muß, wobei eingefügte Zeilenschaltungszeichen (RETURN) innerhalb der Daten die Felder trennen, die später mit Hilfe einer oder mehrerer INPUT#-Anweisungen eingelesen werden. Es ist jedoch wesentlich besser, mit einem

RECORD#-Befehl ausdrücklich anzugeben, welche Datensatznummer gewünscht wird. Dadurch können Sie in beliebiger Reihenfolge auf die Datensätze zugreifen, wobei Sie problemlos innerhalb der Datei hin- und hergehen können.

FORMAT DES RECORD#-BEFEHLS

BASIC 7.0: RECORD#,datei,datensatz [,offset]
BASIC 2.0: PRINT#15,"P"+CHR$(kanal+96)+CHR$(datensatz low)+
CHR$(datensatz high)[+CHR$(offset)]

Hier entspricht „datei" der Dateinummer, die in der aktuellen DOPEN-Anweisung für die jeweilige Datei angegeben wurde. Der Parameter „datensatz" entspricht der gewünschten Datensatznummer, „kanal" entspricht der in der aktuellen OPEN-Anweisung für die jeweilige Datei angegebenen Kanalnummer. „datensatz low" entspricht dem niederwertigen Byte der gewünschten Datensatznummer, als 2 Bytes umfassende ganze Zahl ausgedrückt, während „datensatz high" dem höherwertigen Byte der gewünschten Datensatznummer entspricht. Schließlich stellt ein wahlweiser „offset"-Wert, sofern vorhanden, das Byte innerhalb des Datensatzes dar, bei dem ein folgender Lese- oder Schreibvorgang beginnen soll.
Um diesen Befehl voll zu verstehen, müssen Sie wissen, wie die ganzen Zahlen normalerweise in Computern mit dem 6502- und verwandten Mikroprozessoren abgespeichert werden. Bei der von dem Mikroprozessor benutzten Binärarithmetik kann eine vorzeichenlose ganze Zahl von 0 bis 255 durch ein einziges Byte dargestellt werden. Außerdem kann jede vorzeichenlose ganze Zahl von 0 bis 65535 in 2 Bytes abgespeichert werden, wobei ein Byte den Teil der Zahl enthält, der genau durch 256 geteilt werden kann, und der Rest in dem anderen Byte steht. In Maschinensprache werden derartig aufgeteilte Zahlen in umgekehrter Reihenfolge geschrieben, mit dem niederwertigen Byte (dem Rest der Division durch 256) zuerst, gefolgt von dem höherwertigen Byte. In Assemblerprogrammen, die mit dem Commodore-Assembler geschrieben werden, wird der niederwertige Teil einer 2 Bytes umfassenden Zahl angegeben, indem vor ihrem Kennsatz das Kleiner-als-Zeichen (<) gesetzt wird. Gleichermaßen wird der höherwertige Teil der Zahl durch das Größer-als-Zeichen (>) angegeben.

SICHERHEITSMASSNAHME: JEDER RECORD#-BEFEHL MUSS ZWEIMAL ANGEGEBEN WERDEN

Um die höchst unwahrscheinliche Möglichkeit der Zerstörung relativer Dateidaten auszuschließen, müssen die RECORD#-Befehle zweimal angegeben werden, bevor ein Datensatz gelesen wird.

BEISPIELE:

Um den Pointer (Datensatzzeiger) für Datei 2 auf Datensatz Nummer 3 zu setzen, geben Sie bei BASIC 7.0 folgenden Befehl ein:

```
RECORD#2,3
```

Um den Pointer (Datensatzzeiger) für Kanal 2 auf Datensatz Nummer 3 zu setzen, geben Sie bei BASIC 2.0 folgenden Befehl ein:

```
PRINT#15, "P"+CHR$(98)+CHR$(3)+CHR$(0)
```

CHR$(98) ergibt sich aus der Addition der Konstanten (96) mit der gewünschten Kanalnummer (2) zu (96 + 2 = 98). Auch wenn der Befehl funktionsfähig zu sein scheint, wenn 96 nicht der Kanalnummer hinzugefügt wird, sollte die Konstante normalerweise hinzugefügt werden, um die Kompatibilität mit der Arbeitsweise von RECORD# in BASIC 7.0 aufrecht zu erhalten.
Da 3 kleiner ist als 256, ist das höherwertige Byte in der Binärdarstellung gleich 0, und der ganze Wert paßt in das niederwertige Byte. Da vom Anfang des Datensatzes an gelesen oder geschrieben werden soll, ist kein Offset-Wert erforderlich.
Da diese Berechnungen sehr schnell mühsam werden, übernehmen die meisten Programme diese Aufgabe für den Benutzer. Nachfolgend ein Beispiel für ein Programm, bei dem die Datensatznummer eingegeben und in die erforderliche Form des nieder- und höherwertigen Bytes umgewandelt wird:

```
450 INPUT "GEWUENSCHTE RECORD-NUMMER";RE

460 IF RE<1 OR RE>65535 THEN 450
470 RH=INT(RE/256)
480 RL=RE-256*RH
490 PRINT#15,"P"+CHR$(98)+CHR$(RL)+CHR$(
RH)
```

Geht man davon aus, daß RH und RL wie in dem vorhergehenden Beispiel berechnet werden, so können die Programme auch Variablen für den erforderlichen Kanal, Eintrag und Offset verwenden:

```
570 INPUT "KANAL,RECORD & GEWUENSCHTE PO
SITION";CH,RE,OF
630 PRINT#15,"P"+CHR$(CH+96)+CHR$(RL)+CH
R$(RH)+CHR$(OF)
```

Abschluß des Erstellens einer relativen Datei

Nachdem Sie nun wissen, wie die OPEN- und RECORD#-Befehle benutzt werden, können Sie eigentlich schon eine relative Datei erstellen. Hier brauchen Sie nur noch zu wissen, daß es sich bei CHR$(255) um ein Sonderzeichen in einer relativen Datei handelt. Dieses Zeichen wird vom DOS benutzt, um die relativen Sätze beim Erstellen zu füllen, bevor sie vom Programm mit anderen Informationen gefüllt werden. Beschreiben Sie also den letzten Datensatz, den Sie wahrscheinlich in der Datei nicht mehr benötigen, mit Pseudodaten, die sich nicht auf die spätere Arbeit auswirken, so ist CHR$(255) die beste Lösung. Hier wird beschrieben, wie dieses Zeichen in einem Programm benutzt wird, das Sie für Ihre eigenen relativen Dateiprogramme verwenden können.

BASIC 2.0

```
1020 OPEN 15,8,15
1380 INPUT "NAME DER RELATI
VEN DATEI EINGEBEN";FI$
1390 INPUT "MAXIMALE RECORD-
ANZAHL EINGEBEN";NR
1400 INPUT "RECORD-LAENGE
EINGEBEN";RL
```

Öffnet den Befehlskanal.
Wählt die Dateiparameter aus.

```
1410 OPEN1,8,2,"0:"+FI$+"
,L,"+CHR$(RL)
1420 GOSUB 59990
1430 RH=INT(NR/256)
1440 RL=NR-256*RH
1450 PRINT#15,"P"+CHR$
(96+2)+CHR$(RL)+CHR$(RH)
1455 PRINT#15,"P"+CHR$
(96+2)+CHR$(RL)+CHR$(RH)
1460 GOSUB 59990
1470 PRINT#1,CHR$(255);
1480 GOSUB 59990
1510 CLOSE 1

1520 GOSUB 59990
9980 CLOSE 15
9990 END
59980 REM DISK-UBERPRUEFU
NGS-ROUTINE
```

Beginnt mit dem Erstellen der gewünschten Datei.
Prüft auf Diskettenfehler.
Berechnet die Längenwerte.

Zeiger zu der letzten Nummer des Datensatzes.
Wiederholung dieses Vorgangs aus Sicherheitsgründen.

Sendet das Standardzeichen.

Nun kann die Datei sicher geschlossen werden.

Der Befehlskanal wird geschlossen, bevor das Programm beendet wird.

```
59990 INPUT#15,EN,EM$,ET,ES
60000 IF EN>1 AND EN<>50
THEN PRINT EN,E
M$,ET,ES:STOP
60010 RETURN
```

Subroutine zur Fehlerabfrage; „RECORD NOT PRESENT" wird ignoriert.

BASIC 7.0

```
1380 INPUT "NAME DER RELAT
IVEN DATEI EINGEBEN";FI$
1390 INPUT "MAXIMALE RECORD-
ANZAHL EINGEBEN";NR
1400 INPUT "RECORD-LAENGE
EINGEBEN";RL
1410 DOPEN#1,(FI$),L(RL)

1420 GOSUB 60000
1450 RECORD#1(NR)

1455 RECORD#1(NR)
1460 GOSUB 60000
1470 PRINT#1,CHR$(255);
1480 GOSUB 60000
1510 DCLOSE 1

1520 GOSUB 60000
9980 CLOSE 15
9990 END

60000 IF DS>1 AND DS<>50
THEN PRINT DS,D S$:STOP
60010 RETURN
```

Wählt die Dateiparameter aus.

Beginnt mit dem Erstellen der gewünschten Datei.

Prüft auf Diskettenfehler.

Zeiger zur letzten Datensatznummer.

Sendet das Standardzeichen.

Nun kann die Datei sicher geschlossen werden.

Der Befehlskanal wird geschlossen, bevor das Programm beendet wird.

Subroutine zur Fehlerprüfung. „RECORD NOT PRESENT" wird ignoriert.

Hier müssen zwei Zeilen besonders erläutert werden. Wird Zeile 1470 ausgeführt, so arbeitet das Diskettenlaufwerk maximal einige Minuten, wobei sämtliche Einträge in der Datei bis zu der höchsten Datensatznummer erstellt werden, die in Zeile 1390 ausgewählt wurde. Dies ist normal und braucht nur einmal zu geschehen. Während

dieses Verfahrens hören Sie den Laufwerkmotor und gelegentlich einen leichten Klick-Ton, während der Schreib-/Lesekopf zur nächsten Spur weitergeht. Zweitens unterscheidet sich Zeile 60000 oben von der entsprechenden Zeile in der vorher angegebenen Subroutine zur Fehlerabfrage. Hier wird die Diskettenfehlernummer 50 ausdrücklich ignoriert, da sie generiert wird, sobald der Fehlerkanal in Zeile 1460 überprüft wird. Diese Meldung wird ignoriert, da das Nichtvorhandensein eines angeforderten Datensatzes nur dann einen Fehler darstellt, wenn dieser vorher erstellt wurde.

6.6 ERWEITERN EINER RELATIVEN DATEI

Was jedoch, wenn Sie die Anforderungen unterschätzen und eine relative Datei später erweitern müssen? Kein Problem! Fordern Sie einfach die benötigte Datensatznummer an, selbst wenn sie augenblicklich nicht in der Datei vorhanden ist. Ist noch kein derartiger Datensatz vorhanden, so wird er vom DOS erstellt, sobald Sie versuchen, Informationen in diesen Eintrag zu schreiben. Außerdem erstellt das DOS automatisch eventuell fehlende Datensätze mit niedrigerer Datensatznummer. Wird der erste Datensatz hinter dem momentan letzten Datensatz geschrieben, so gibt das DOS die Fehlermeldung „50, RECORD NOT PRESENT" (50, Datensatz nicht vorhanden) aus. Diese Fehlermeldung wird erwartet und ist korrekt.

6.7 SCHREIBEN VON DATEN IN DIE RELATIVE DATEI

Bei den Befehlen zum Lesen und Schreiben relativer Daten handelt es sich um dieselben PRINT#-, INPUT#- und GET#-Befehle, die auch schon im vorigen Kapitel über sequentielle Dateien benutzt wurden. Jeder Befehl wird wie dort beschrieben verwendet. Einige Aspekte des Zugriffs auf relative Dateien unterscheiden sich jedoch von dem auf sequentielle Dateien. Diese Unterschiede werden hier im folgenden beschrieben.

6.7.1 AUFBAU EINES RELATIVEN DATENSATZES

Wie bereits am Anfang dieses Kapitels angegeben, verfügt jeder relative Datensatz über eine feste Länge, einschließlich aller Sonderzeichen. Innerhalb dieser festen Länge gibt es zwei bekannte Möglichkeiten zur Organisation verschiedener individueller Informationsfelder. Einmal handelt es sich um das sogenannte freie Format, bei dem die Länge der einzelnen Felder von Datensatz zu Datensatz variiert und bei dem die einzelnen Felder durch ein Zeilenschaltungszeichen (RETURN) voneinander getrennt werden. (Jedes dieser Zeilenschaltungszeichen belegt ein Zeichen in dem

Eintrag.) Die andere Lösung besteht in der Verwendung von Feldern mit fester Länge, die durch ein RETURN voneinander getrennt werden können oder nicht. Werden Felder mit fester Länge nicht durch ein RETURN voneinander getrennt, so müssen Sie sicher sein, daß innerhalb jedes 88 bzw. 160 Zeichen umfassenden Teils des Datensatzes ein RETURN enthalten ist. (Der Wert 88 gilt für BASIC 2 und der Wert 160 für BASIC 7.) Ist dies nicht der Fall, so muß der Datensatz mit dem GET#-Befehl gelesen werden, was zu einem beträchtlichen Geschwindigkeitsverlust führt. Da jeder relative Eintrag problemlos mit einer einzigen PRINT#-Anweisung geschrieben wird, wird empfohlen, eine Kopie des aktuellen Datensatzes im Speicher zu erstellen, bevor dieser auf die Diskette geschrieben wird. Er kann mit Hilfe der vielen BASIC-Funktionen zur Verarbeitung von Zeichenfolgen in einer einzigen Zeichenfolgenvariablen zusammengefaßt und dann in einem durch diese Variable geschrieben werden.
Nun ein Beispiel hierzu: Wir schreiben ein vier Zeilen umfassendes Adreßetikett, das aus den vier Feldern „NAME", „STRASSE", „POSTLEITZAHL" und „STADT" besteht. Die gesamte Datensatzlänge beläuft sich auf 87 Zeichen. Dieses Adreßetikett kann auf zweierlei Art organisiert werden:

FELDER MIT FESTER LÄNGE

Feldname	Länge
NAME	27 Zeichen
STRASSE	27 Zeichen
POSTLEITZAHL	10 Zeichen
STADT	23 Zeichen
Gesamtlänge	87 Zeichen

FELDER MIT VARIABLER LÄNGE

Feldname	Länge
NAME	31 Zeichen
STRASSE	31 Zeichen
POSTLEITZAHL	11 Zeichen
STADT	26 Zeichen
Maximale Länge	99 Zeichen
Eingegebene Länge	87 Zeichen

Bei Feldern mit fester Länge ergibt die Addition der Feldlängen genau die Datensatzlänge. Da die Gesamtlänge innerhalb der Grenzen für die Größe des Eingabepuffers liegt, sind keine RETURNs erforderlich. Bei Feldern mit variabler Länge kann die Vielseitigkeit der aktuellen Adreßlänge berücksichtigt werden. Während ein Name 27 Buchstaben umfaßt, kann ein anderer nur 15 Buchstaben enthalten. Dieselbe Möglichkeit besteht bei der Länge für Straße und Stadt. Obwohl bei den Feldern mit variabler Länge jeweils ein Zeichen pro Feld für die RETURNs verloren geht, kann bei ihnen die Differenz zwischen der maximalen Feldlänge und der durchschnittlichen Feldlänge voll ausgenutzt werden. Ein Programm, das variable Feldlängen benutzt, muß die Gesamtlänge jedes Datensatzes bei der Eingabe berechnen, um zu gewährleisten, daß die Gesamtsumme der Felder den zur Verfügung stehenden Platz nicht überschreitet.

6.7.2 SCHREIBEN VON DATENSÄTZEN

Nachfolgend ein Beispiel für Programmzeilen zur Eingabe von Feldern mit variabler Länge mit dem obigen Dateiaufbau, wobei diese dann in einer einzigen Zeichenfolge zusammengefaßt und zu der Datensatznummer RE in Datei 3 gesendet werden. (Hier wird davon ausgegangen, daß es sich um eine relative Datei handelt, die Kanal Nummer 3 benutzt.)

BASIC 7.0:

```
100 INPUT "NUMMER DES DATENSATZES";RE
110 :
120 DOPEN#3,"MYRELFILE",L88
130 CR$=CHR$(13)
140 INPUT "NAME";NA$
150 IF LEN(A$)>30 THEN 140
160 INPUT "STRASSE";SA$
170 IF LEN(SA$)>30 THEN 160
180 INPUT "STADT";CS$
190 IF LEN(CS$)>25 THEN 180
200 INPUT "POSTLEITZAHL";ZP$
210 IF LEN(ZP$)>10 THEN 200
220 DA$=NA$+CR$+SA$+CR$+CS$+CR$+ZP$
230 IF LEN (DA$)<88 THEN 260
240 PRINT"RECORD IST ZU LANG"
250 GOTO 140
260 :
270 :
280 RECORD#3,(RE),1
290 IF DS=50 THEN PRINT#3,CHR$(255):GOSU
B 1000:GOTO 280
300 GOSUB 1000
310 PRINT#3,DA$
320 GOSUB 1000
330 RECORD#3,(RE),1
340 GOSUB 1000
350 DCLOSE#3:END
1000 IF DS<20 THEN RETURN
1002 :
1010 PRINT DS$:DCLOSE#3:END
```

BASIC 2.0:

```
100 INPUT "NUMMER DES DATENSATZES";RE
110 OPEN 15,8,15
120 OPEN 3,8,3,"MYRELFILE,L,"+CHR$(88)
130 CR$=CHR$(13)
140 INPUT "NAME";NA$
150 IF LEN(A$)>30 THEN 140
160 INPUT "STRASSE";SA$
170 IF LEN(SA$)>30 THEN 160
180 INPUT "STADT";CS$
190 IF LEN(CS$)>25 THEN 180
200 INPUT "POSTLEITZAHL";ZP$
210 IF LEN(ZP$)>10 THEN 200
220 DA$=NA$+CR$+SA$+CR$+CS$+CR$+ZP$
230 IF LEN (DA$)<88 THEN 260
240 PRINT"RECORD IST ZU LANG"
250 GOTO 140
260 RH=INT(RE/256)
270 RL=RE-256*RH
280 PRINT#15,"P"+CHR$(96+3)+CHR$(RL)+CHR
$(RH)+CHR$(1)
290 GOSUB 1000:IF EN=50 THEN PRINT#3,CHR
$(255):GOSUB 1000:GOTO 280
300 GOSUB 1000
310 PRINT#3,DA$
320 GOSUB 1000
330 PRINT#15,"P"+CHR$(96+3)+CHR$(RL)+CHR
$(RH)+CHR$(1)
340 GOSUB 1000
350 CLOSE 3:CLOSE 15:END
1000 INPUT#15,EN,EM$,ET,ES
1002 IF EN<20 THEN RETURN
1010 PRINT EM$:CLOSE 3:CLOSE 15:END
```

Sollen die obigen Programmzeilen für Felder mit fester Länge benutzt werden, so werden folgende Zeilen geändert:

BASIC 7.0:

```
100 INPUT "DATENSATZNUMMER";RE
110 :
120 DOPEN#3,"MYRELFILE",L88
130 BL$="":FORI=1TO27:BL$=BL$+CHR$(160):
NEXTI
140 INPUT "NAME";NA$
145 LN=LEN(NA$)
150 IF LN>27 THEN 140
155 NA$=NA$+LEFT$(BL$,27-LN)
160 INPUT "STRASSE";SA$
165 LN=LEN(SA$)
170 IF LN>27 THEN 160
175 SA$=SA$+LEFT$(BL$,27-LN)
180 INPUT "STADT";CS$
185 LN=LEN(CS$)
190 IF LN>23 THEN 180
195 CS$=CS$+LEFT$(BL$,27-LN)
200 INPUT "POSTLEITZAHL";ZP$
205 LN=LEN(ZP$)
210 IF LN>10 THEN 200
215 ZP$=ZP$+LEFT$(BL$,27-LN)
220 DA$=NA$+SA$+CS$+ZP$
260 :
270 :
280 RECORD#3,(RE),1
290 IF DS=50 THEN PRINT#3,CHR$(255):GOSU
B 1000:GOTO 280
300 GOSUB 1000
310 PRINT#3,DA$
320 GOSUB 1000
330 RECORD#3,(RE),1
340 GOSUB 1000
350 DCLOSE#3:END
1000 IF DS<20 THEN RETURN
1002 :
1020 PRINT DS$:DCLOSE#3:END
```

BASIC 2.0:

```
100 INPUT "NUMMER DES DATENSATZES";RE
110 OPEN 15,8,15
120 OPEN 3,8,3,"MYRELFILE,L,"+CHR$(88)
130 BL$="":FORI=1TO27:BL$=BL$+CHR$(160):
NEXTI
140 INPUT "NAME";NA$
145 LN=LEN(NA$)
150 IF LEN(A$)>30 THEN 140
155 NA$=NA$+LEFT$(BL$,27-LN)
160 INPUT "STRASSE";SA$
165 LN=LEN(SA$)
170 IF LEN(SA$)>30 THEN 160
175 SA$=SA$+LEFT$(BL$,27-LN)
180 INPUT "STADT";CS$
185 LN=LEN(CS$)
190 IF LEN(CS$)>25 THEN 180
195 CS$=CS$+LEFT$(BL$,23-LN)
200 INPUT "POSTLEITZAHL";ZP$
205 LN=LEN(ZP$)
210 IF LEN(ZP$)>10 THEN 200
215 ZP$=ZP$+LEFT$(BL$,10-LN)
220 DA$=NA$+CR$+SA$+CR$+CS$+CR$+ZP$
260 RH=INT(RE/256)
270 RL=RE-256*RH
280 PRINT#15,"P"+CHR$(96+3)+CHR$(RL)+CHR
$(RH)+CHR$(1)
290 GOSUB 1000:IF EN=50 THEN PRINT#3,CHR
$(255):GOSUB 1000:GOTO 280
300 GOSUB 1000
310 PRINT#3,DA$
320 GOSUB 1000
330 PRINT#15,"P"+CHR$(96+3)+CHR$(RL)+CHR
$(RH)+CHR$(1)
340 GOSUB 1000
350 CLOSE 3:CLOSE 15:END
1000 INPUT#15,EN,EM$,ET,ES
1002 IF EN<20 THEN RETURN
1010 PRINT EM$:CLOSE 3:CLOSE 15:END
```

Unterscheidet sich der Feldinhalt in der Länge, so sind variable Feldlängen häufig vorteilhaft. Sind die Feldlängen jedoch gleichmäßig, so sind Felder mit fester Länge vorzuziehen. Felder mit fester Länge sind auch erforderlich, wenn der wahlweise auszugebende Offset-Parameter des RECORD#-Befehls bei einem bestimmten

Byte innerhalb des Datensatzes benutzt werden soll. Hier muß jedoch vor dieser Benutzung des Offsets gewarnt werden. Wird nur ein Teil eines Datensatzes geschrieben, so überschreibt das DOS eventuell verbleibende Leerzeichen. Muß also die Offset-Option benutzt werden, so darf mit Ausnahme des letzten Feldes kein Feld in einem Datensatz aktualisiert werden, wenn nicht alle vorhergehenden Felder ebenfalls später aus dem Speicher aktualisiert werden.
Bei den obigen Programmen wird sorgfältig darauf geachtet, daß die Datensatzlänge genau mit dem zur Verfügung stehenden Platz übereinstimmt. Bei Programmen, bei denen dies nicht der Fall ist, füllt DOS kurze Einträge mit Füllzeichen auf die volle Größe auf und schneidet überlange Einträge ab, damit sie in den ihnen zugewiesenen Platz passen. Wird ein Eintrag abgeschnitten, so gibt das DOS Fehler 51, „RECORD OVERFLOW" (Datensatzüberlauf) an. Zu kurze Einträge werden ohne eine DOS-Fehlermeldung akzeptiert.

6.8 LESEN VON DATENSÄTZEN

Nachdem ein relativer Datensatz richtig auf die Diskette geschrieben worden ist, kann er auf recht einfache Weise wieder in den Speicher des Computers zurückgelesen werden. Auch hier unterscheidet sich jedoch das Verfahren, je nachdem, ob Felder mit fester oder variabler Länge benutzt wurden. Nachfolgend werden die Programmzeilen aufgeführt, mit denen die oben erstellten Felder mit variabler Länge wieder aus Datensatz Nummer RE der Datei über Kanal 3 zurückgelesen werden können

BASIC 7.0:

```
10 :
20 DOPEN#3,"MYRELFILE",L88
30 INPUT "NUMMER DES DATENSATZES";RE
40 :
50 :
60 RECORD#3,(RE),1
70 GOSUB 1000
80 INPUT#3,NA$,SA$,CS$,ZP$
90 GOSUB 1000
100 RECORD#3,(RE),1
110 GOSUB 1000
120 PRINT NA$:PRINTSA$
130 PRINT CS$:PRINT ZP$
140 DCLOSE#3:END
1000 IF DS<20 THEN RETURN
1002 :
1010 PRINT"FEHLER: ";DS$:DCLOSE#3:END
```

BASIC 2.0:

```
10 OPEN 15,8,15
20 OPEN 3,8,3,"MYRELFILE,L,"+CHR$(88)
30 INPUT "NUMMER DES DATENSATZES";RE
40 RH=INT(RE/256)
50 RL=RE-256*RH
60 PRINT#15,"P"+CHR$(96+3)+CHR$(RL)+CHR$
(RH)+CHR$(1)
70 GOSUB 1000
80 INPUT#3,NA$,SA$,CS$,ZP$
90 GOSUB 1000
100 PRINT#15,"P"+CHR$(96+3)+CHR$(RL)+CHR
$(RH)+CHR$(1)
110 GOSUB 1000
120 PRINT NA$:PRINT SA$
130 PRINT CS$:PRINT ZP$
140 CLOSE 3:CLOSE 15:END
1000 INPUT#15,EN,EM$,ET,ES
1002 IF EN<20 THEN RETURN
1010 PRINT"FEHLER: ";EM$:CLOSE 3:END
```

Nachfolgend die Programmzeilen, mit denen die Felder mit fester Länge wieder zurückgelesen werden:

BASIC 7.0:

```
10 :
20 DOPEN#3,"MEINERELDATEI",L88
30 INPUT "RECORD-NUMMER EINGEBEN";RE
40 :
50 :
60 RECORD#3,(RE),1
70 GOSUB 1000
80 INPUT#3,DA$
90 GOSUB 1000
100 RECORD#3,(RE),1
110 GOSUB 1000
112 NA$=LEFT$(DA$,27)
114 SA$=MID$(DA$,28,27)
```

```
116 CS$=MID$(DA$,55,23)
118 ZP$=RIGHT$(DA$,10)
120 PRINT NA$:PRINT SA$
130 PRINT CS$:PRINT ZP$
140 DCLOSE#3:END
1000 IF DS<20 THEN RETURN
1002 :
1010 PRINT"FEHLER:";DS$:DCLOSE#3:END
```

BASIC 2.0:

```
10 OPEN 15,8,15
20 OPEN 3,8,3,"MEINERELDATEI,L"+CHR$(88)

30 INPUT "RECORD-NUMMER EINGEBEN";RE
40 RH=INT(RE/256)
50 RL=RE-256*RH
60 PRINT#15,"P"+CHR$(96+3)+CHR$(RL)+CHR$
(RH)+CHR$(1)
70 GOSUB 1000
80 INPUT#3,DA$
90 GOSUB 1000
100 PRINT#15,"P"+CHR$(96+3)+CHR$(RL)+CHR
$(RH)+CHR$(1)
110 GOSUB 1000
112 NA$=LEFT$(DA$,27)
114 SA$=MID$(DA$,28,27)
116 CS$=MID$(DA$,55,23)
118 ZP$=RIGHT$(DA$,10)
120 PRINT NA$:PRINT SA$
130 PRINT CS$:PRINT ZP$
140 CLOSE 3:CLOSE 15:END
1000 INPUT#15,EN,EM$,ET,ES
1002 IF EN<20 THEN RETURN
1010 PRINT"FEHLER:";EM$:CLOSE 3:CLOSE 15
:END
```

6.9 VORTEILE DURCH DIE VERWENDUNG INDIZIERTER DATEIEN

In den letzten Kapiteln wurde beschrieben, wie sequentielle und relative Dateien separat benutzt werden. Sie werden jedoch auch häufig gleichzeitig verwendet, wobei in der sequentiellen Datei kurz festgehalten wird, welcher Name in der relativen Datei in jedem Datensatz gespeichert wird. Auf diese Weise kann der Inhalt der sequentiellen Datei in ein Feld von Strings gelesen und alphabetisch sortiert werden. Nach dem Sortieren kann mit einer Technik, die als Binärsuche bezeichnet wird, ein in das Feld eingegebener Name schnell gefunden und der zugehörige Datensatz in der relativen Datei gelesen oder geschrieben werden. Komplexere Programme arbeiten häufig auch mit zwei oder noch mehr solcher indizierten Dateien, die nach verschiedenen Kriterien sortiert worden sind.

KAPITEL 7

DIREKTZUGRIFFSDATEIEN

Befehle für den Direktzugriff beziehen sich auf einzelne Sektoren der Diskette, wobei Daten vollständig unter Anweisungen des Benutzers gelesen und geschrieben werden. Durch diese Befehle erhält man eine fast vollständige Flexibilität in Programmen zur Datenverarbeitung. Allerdings übernimmt der Programmierer eine ziemlich große Verantwortung, da er nun selbst für die Richtigkeit der Operationen verantwortlich ist. Demzufolge werden diese Befehle normalerweise nur in komplexen kommerziellen Programmen benutzt, die die Daten einwandfrei ohne Hilfe des DOS selbst organisieren können.
Befehle für den Direktzugriff werden wesentlich häufiger in Dienstprogrammen benutzt, mit denen Teile der Diskette angezeigt und geändert werden, die normalerweise nicht direkt sichtbar sind. Mit derartigen Befehlen kann beispielsweise der Name einer Diskette geändert werden, ohne die auf ihr stehenden Programme zu löschen, ein Programm geschützt werden, damit es nicht gelöscht werden kann, oder der Benutzername an einer Stelle abgespeichert werden, an der er normalerweise nicht erwartet wird.

7.1 DISKETTENORGANISATION

Auf einer Diskette sind insgesamt 683 Blöcke (im 1541- und im 1570-Modus) und 1366 Blöcke (im 1571-Modus) vorhanden, von denen 664 (1328) benutzt werden können, da einige Blöcke für die BAM (Block Availability Map) und das Inhaltsverzeichnis reserviert sind.
Die Diskettenoberfläche ist in Spuren unterteilt, die konzentrische Kreise auf der Diskettenoberfläche bilden. Pro Seite sind 35 verschiedene Spuren vorhanden, wobei am äußeren Rand der Diskette mit Spur 1 begonnen und bis zu Spur 35 am inneren Lochrand gegangen wird. Spur 18 (im 1541-Modus und bei der 1570) und Spur 53 (im 1571-Modus) werden für das Inhaltsverzeichnis und die BAM benutzt. Das DOS beschreibt die Diskette von der Mitte einer Diskettenseite nach außen und innen abwechselnd in beiden Richtungen.
Jede Spur ist in Sektoren, die auch als Blöcke bezeichnet werden, unterteilt. Da bei den äußeren Spuren mehr Platz als bei den inneren vorhanden ist, enthalten diese mehr Sektoren. Die äußersten Spuren umfassen jeweils 21 Sektoren, während die

innersten Spuren nur jeweils 17 Sektoren aufweisen. In der folgenden Tabelle wird die Anzahl der Sektoren pro Spur angegeben.

Tabelle 6.1 Format von Spur und Sektor

Spuren	Vorhandene Sektoren	Anzahl der Sektoren		
1 bis 17	0 bis 20	21	1541- bzw. 1570- Modus	1571- Modus
18 bis 24	0 bis 18	19		
25 bis 30	0 bis 17	18		
31 bis 35	0 bis 16	17		
36 bis 52	0 bis 20	21		
53 bis 59	0 bis 18	19		
60 bis 65	0 bis 17	18		
66 bis 70	0 bis 16	17		

In diesem Kapitel werden die DOS-Befehle beschrieben, mit denen jede Spur und jeder Block auf der Diskette direkt gelesen und geschrieben werden kann, sowie die Befehle, mit denen die Blöcke als belegt oder nicht belegt markiert werden.

7.2 ÖFFNEN EINES DATENKANALS FÜR DEN DIREKTZUGRIFF

Wird mit Direktzugriffsdateien gearbeitet, so müssen zwei Kanäle für die Diskette geöffnet sein: der Befehlskanal, der bereits erwähnt wurde, und ein weiterer Kanal für die Daten. Der Befehlskanal wird mit dem üblichen Befehl OPEN 15,8,15 oder einem entsprechenden Befehl geöffnet. Ein Datenkanal für den Direktzugriff wird ähnlich wie andere Dateien geöffnet, allerdings wird das „Nummernzeichen" (#), auf das wahlweise die Nummer des Puffers im Speicher der 1570/71 folgt, als Dateiname verwendet.

FORMAT DER OPEN-ANWEISUNGEN FÜR DIREKTZUGRIFFSDATEIEN:

OPEN datei, gerät, kanal, "#puffer"

wobei „datei" der Dateinummer und „gerät" der Geräteadresse der 1570/71 (normalerweise 8), entspricht. „kanal" entspricht der Kanalnummer, einer Zahl zwischen 2 und 14, die noch nicht von ebenfalls geöffneten Dateien benutzt wird. Falls vorhanden, ist „puffer" eine Zahl zwischen 0 und 3 und gibt die Nummer des

für diesen Kanal zu verwendenden Datenpuffers an. Die Lage des Puffers im Speicher geht aus folgender Tabelle hervor:

Nummer des Puffers	Adressbereich im Speicher der 1570/71
0	$0300 ... $03FF
1	$0400 ... $04FF
2	$0500 ... $05FF
3	$0680 ... $06FF

BEISPIELE:

Wird keine Puffernummer angegeben, so wählt das DOS der 1571 einen Puffer aus:

```
OPEN 5,8,5,"#"
```

Der Benutzer kann jedoch den Puffer auch selbst wählen:

```
OPEN 4,8,4,"#2"
```

7.3 DIREKTZUGRIFFSBEFEHLE

7.3.1 BLOCK-READ

Mit einem BLOCK-READ-Befehl wird der Inhalt eines Sektors in einen Puffer geladen. Auch wenn der BLOCK-READ-Befehl (B-R) noch Bestandteil des DOS-Befehlsvorrates ist, wird er nahezu stets durch den Befehl U1 ersetzt (siehe Kapitel 8).

FORMAT DES BLOCK-READ-BEFEHLS:

```
PRINT#15, "U1"; kanal;laufwerk,spur,sektor
```

„kanal" ist die Kanalnummer, die beim Öffnen der Datei, aus der der Block gelesen werden soll, als Sekundäradresse angegeben wurde. „laufwerk" ist die Laufwerksnummer, während „spur" und „sektor" Spur und Sektor des zu lesenden Blocks enthalten.

ALTERNATIVFORMATE:

PRINT#15, "U1:"kanal;laufwerk;spur;sektor
PRINT#15, "UA:"kanal;laufwerk;spur;sektor
PRINT#15, "U1 kanal laufwerk spur sektor"

BEISPIEL:

Nachfolgend ein vollständiges Programm, mit dem ein Sektor mit Hilfe von U1 in den Diskettenspeicher und von dort mit der GET#-Anweisung in den Speicher des Computers gelesen wird. (Ist mindestens einmal pro 88 Datenbytes ein RETURN-Zeichen [CHR$(13)] vorhanden, so kann INPUT# anstelle von GET# verwendet werden.)

```
110 MB=7936:REM $1F00
120 INPUT "WELCHE SPUR";T
130 INPUT "WELCHER SEKTOR";S
140 OPEN 15,8,15
150 OPEN 5,8,5,"#"

160 PRINT#15,"U1";5;0;T;S
170 FOR I=MB TO MB+255
180 GET#5,A$:IF A$=
""THEN A$=CHR$(0)
190 POKE I,ASC(A$)
200 NEXT I
210 CLOSE 5:CLOSE 15
220 END
```

Wahl eines Puffers
Wahl von Spur
und Sektor aus
Öffnen des Befehlskanals
Öffnen des Kanals für den Direktzugriff
Lesen des Sektors in den Diskettenpuffer
Verwendung einer Schleife, um den Diskettenpuffer in den Speicher des Computers zu kopieren
Schließen der Kanäle

Während die Schleife ausgeführt wird, wird der Inhalt des gewählten Blocks in den Speicher des Computers kopiert, wobei bei der Adresse begonnen wird, die mit der Variablen MB in Zeile 110 festgelegt wird.

7.3.2 BLOCK-WRITE

Mit einem BLOCK-WRITE-Befehl kann der Inhalt eines Dateienpuffers in einen bestimmten Sektor gesichert werden. Er ist somit das Gegenstück des BLOCK-READ-Befehls. Auch wenn der BLOCK-WRITE-Befehl (B-W) noch Bestandteil des DOS-Befehlsvorrates ist, wird er nahezu stets durch den Befehl U2 ersetzt.

FORMAT DES BLOCK-WRITE-BEFEHLS:
PRINT#15, "U2"; kanal;laufwerk;spur;sektor

„kanal" entspricht der Kanalnummer, die beim Öffnen der Datei, in die der Block gelesen wird, als Sekundäradresse angegeben wurde. „laufwerk" ist die Laufwerksnummer, während „spur" und „sektor" die Spur- bzw. Sektorennummern darstellen, die den aus dem Dateipuffer gesicherten Datenblock empfangen.

ALTERNATIVFORMATE:

```
PRINT#15, "U2:"kanal;laufwerk;spur;sektor
PRINT#15, "UB:"kanal;laufwerk;spur;sektor
PRINT#15, "U2 kanal laufwerk spur sektor"
```

BEISPIELE:
Um Spur 18, Sektor 1 aus dem Inhaltsverzeichnis vom Puffer auf die Diskette zurückzuschreiben, der mit Hilfe eines BLOCK-READ-Befehls beschrieben wurde, wird folgender Befehl benutzt:

```
PRINT#15, "U2";5;0;18;1
```

Nach der Beschreibung, wie das Inhaltsverzeichnis richtig geändert werden kann, wird noch einmal auf dieses Beispiel Bezug genommen.
Mit BLOCK-WRITE können Sie beispielsweise auch einen Namen in Spur 1, Sektor 1 schreiben; dies ist ein nur selten benutzter Sektor. Damit könnte eine Diskette als Ihnen gehörig markiert werden. Nachfolgend ein Programm, das diesen Zweck erfüllt, wobei die Alternativform des BLOCK-WRITE-Befehls verwendet wird:

```
110 INPUT "IHR NAME";NA$        Eingabe des Namens
120 OPEN 15,8,15                Öffnen des Befehlskanals
130 OPEN 4,8,4,"#"              Öffnen des Datenkanals für den
140 PRINT#4,NA$                 Direktzugriff
150 PRINT#15,"U2";4;0;1;1
160 CLOSE 4
170 CLOSE 15
180 END
```

7.3.3 BLOCK-READ UND -WRITE MIT PUFFERZEIGER

Auch wenn die Befehle BLOCK-READ und BLOCK-WRITE nahezu stets durch U1 bzw. U2 ersetzt werden, können sie ebenfalls sinnvoll angewendet werden, wenn ihre Arbeitsweise voll verstanden wird. Im Gegensatz zu U1 und U2 können Sie mit

B-R und B-W einen Sektor nicht vollständig lesen oder schreiben. Bei B-R wird mit dem ersten Byte des ausgewählten Sektors ein Pufferzeiger (siehe nächster Abschnitt) verwaltet, durch den festgelegt wird, wie viele Bytes dieses Sektors als für den Anwender vorhanden anzusehen sind. Mit einem Programm kann geprüft werden, ob nicht etwa doch versucht wird, über das Ende der tatsächlich in den Puffer geladenen Daten hinauszulesen. Hierbei überprüft das Programm, ob der Wert der Statusvariablen ST von 0 auf 64 geht. Wird der Pufferinhalt mit B-W wieder auf die Diskette zurückgeschrieben, so handelt es sich bei dem ersten geschriebenen Byte um den aktuellen Wert des Pufferzeigers. In den ausgewählten Sektor werden beim Schreiben nur so viele Bytes übernommen, wie durch den Pufferzeiger angegeben werden. Somit können B-R und B-W bei der Arbeit mit individuell aufgebauten Dateistrukturen überaus nützlich sein.

FORMAT DER BLOCK-READ- UND BLOCK-WRITE-BEFEHLE IN IHRER NORMALFORM:

PRINT#15,"BLOCK-READ";kanal;laufwerk;spur;sektor

In der abgekürzten Form:

PRINT#15,"B-R";kanal;laufwerk;spur;sektor

und

PRINT#15,"BLOCK-WRITE";kanal;laufwerk;spur;sektor

In der abgekürzten Form:

PRINT#15,"B-W";kanal;laufwerk;spur;sektor

„kanal" entspricht der Kanalnummer, die beim Öffnen der Datei, in die der Block gelesen wird, als Sekundäradresse angegeben wurde. „laufwerk" entspricht der Laufwerksnummer, während „spur" und „sektor" die Spur- bzw. Sektornummern darstellen, die den Datenblock enthalten, der teilweise in den Dateipuffer gelesen bzw. aus diesem geschrieben werden soll.

WICHTIGE HINWEISE:

1. In einem BLOCK-READ-BEFEHL in seiner Normalform wird mit dem ersten Byte des ausgewählten Sektors angegeben, wie viele Bytes dieses Sektors gültige Daten enthalten. Er kann somit nicht dazu verwendet werden, einen ganzen Sektor in den Puffer einzulesen, da das erste Datenbyte immer als die Anzahl der zu lesenden Zeichen und nicht als Bestandteil der Daten interpretiert wird.
2. In einem BLOCK-WRITE-Befehl in seiner Normalform entspricht das erste

geschriebene Byte ebenfalls dem aktuellen Wert des Pufferzeigers, wenn der Puffer wieder auf die Diskette geschrieben wird. Es werden nur so viele Bytes in den ausgewählten Sektor geschrieben, wie durch diesen Pufferzeiger angegeben werden. Dieser Befehl kann nicht dazu verwendet werden, einen ganzen Sektor unverändert auf die Diskette zu schreiben, da das erste Datenbyte von dem Pufferzeiger überschrieben wird.

7.3.4 SETZEN DES PUFFERZEIGERS

Der Pufferzeiger weist auf die Stelle, die vom nächsten Lese- oder Schreibbefehl des Rechners (GET#, PRINT#, ...) betroffen wird. Wird nämlich der Pufferzeiger verschoben, kann auf einzelne Bytes innerhalb eines Blocks in beliebiger Reihenfolge zugegriffen werden. Dadurch kann ein beliebiger Teil eines Sektors editiert oder in Feldern wie ein relativer Satz organisiert werden.

FORMAT DES BUFFER-POINTER-BEFEHLS:

PRINT#15,"BUFFER-POINTER";kanal;byte

Dieser Befehl wird i. a. folgendermaßen abgekürzt:

PRINT#15,"B-P";kanal;byte

„kanal" ist die Kanalnummer, die beim Öffnen der Datei als Sekundäradresse angegeben wurde und die den Puffer reserviert. Mit „byte" wird die Nummer des Zeichens innerhalb des Puffers angegeben, auf das gezeigt werden soll.

ALTERNATIVFORMATE:

PRINT#15,"B-P:"kanal;byte
PRINT#15,"B-P:kanal byte"

BEISPIEL:

Nachfolgend ein Programm, mit dem die erste Datei auf einer Diskette vor dem Löschen und Verändern geschützt wird. Es liest den Anfang des Inhaltsverzeichnisses (Spur 18, Sektor 1) in den Diskettenspeicher und setzt den Pufferzeiger auf das erste Byte des Dateityps. Danach wird die Datei geschützt, indem Bit 6 der Dateitypspezifikation gesetzt wird, und schließlich neu geschrieben.
(In Anhang 9.3 wird die Organisation des Inhaltsverzeichnisses im einzelnen beschrieben.)

```
110 OPEN 15,8,15
120 OPEN 5,8,5,"#"
130 PRINT#15,"U1";5;0;18;1
140 PRINT#15,"B-P";5;2
150 GET#5,A$: IF A$="" THEN A$=CHR$(0)
160 A=ASC(A$)OR 64
170 PRINT#15,"B-P";5;2
180 PRINT#5,CHR$(A);
190 PRINT#15,"U2";5;0;18;1
200 CLOSE 5
210 CLOSE 15
220 END
```

Öffnen des Befehlskanals
Öffnen des Datenkanals für den Direktzugriff
Lesen des Blocks von Spur 18, Sektor 1
Zeiger auf Byte 2 des Puffers setzen
Einlesen in den Speicher des Rechners
Bit 6 setzen zwecks Schutz
Zeiger wieder auf Dateityp (Byte 2) setzen
Alten Wert überschreiben
Geänderten Pufferinhalt auf Diskette zurückschreiben
Kanäle wieder schließen

Nachdem das obige Programm ausgeführt wurde, kann die erste Datei auf dieser Diskette nicht mehr gelöscht werden. Soll diese Datei später gelöscht werden, so führen Sie dasselbe Programm erneut aus, benutzen jedoch die geänderte Zeile 160, um die Datei wieder freizugeben:

```
160 A=ASC(A$)AND191
```

löscht Bit 6, um die Datei freizugeben.

7.3.5 BELEGUNG VON BLÖCKEN

Nachdem Daten mit Hilfe der Befehle für den Direktzugriff in einem bestimmten Sektor auf der Diskette geschrieben wurden, empfiehlt es sich, diesen Sektor als belegt zu markieren, damit keine anderen Dateien diesen Sektor überschreiben können. Auf diese Weise zugewiesene Blöcke sind sicher, bis auf der Diskette ein VALIDATE bzw. COLLECT durchgeführt wird.

FORMAT DES BLOCK-ALLOCATE-BEFEHLS:

PRINT#,"BLOCK-ALLOCATE";laufwerk;spur;sektor

Dieser Befehl wird i. a. folgendermaßen abgekürzt:

PRINT#15,"B-A";laufwerk;spur;sektor

„laufwerk" entspricht der Laufwerksnummer, während „spur" und „sektor" Spur und Sektor festlegen, der als belegt zu kennzeichnen ist.

ALTERNATIVFORMAT:

PRINT#15,"B-A:"laufwerk;spur;sektor

BEISPIEL:

Wird versucht, einen Block als belegt zu kennzeichnen, der nicht verfügbar ist, so gibt DOS die Fehlermeldung 65, „NO BLOCK" (Kein Block) aus. Die Werte für Spur und Sektor (s. Fehlermeldung) werden auf den nächsten verfügbaren Block gesetzt. Bevor ein Block zum Schreiben gewählt wird, muß also zuerst versucht werden, diesen Block als belegt zu kennzeichnen. Steht der Block nicht zur Verfügung, so lesen Sie den nächsten verfügbaren Block aus dem Fehlerkanal und weisen ihn stattdessen zu. In Spur 18 (Inhaltsverzeichnis) dürfen jedoch keine Datenblöcke abgelegt bzw. als belegt gekennzeichnet werden. Wird durch den Fehlerkanal Spur 0 als nächster freier Block zurückgegeben, so bedeutet dies, daß keine Blöcke mehr verfügbar sind und die Diskette voll ist.
Nachfolgend ein Programm, mit dem ein Text in einem freien Block auf der Diskette abgelegt und der entsprechende Block als belegt gekennzeichnet wird:

```
100 OPEN 15,8,15
110 OPEN 5,8,5,"#"

120 PRINT#5,
"ICH DENKE ALSO BIN ICH"
130 T=1:S=1


140 PRINT#15,"B-A";0;T;S
150 INPUT#15,EN,EM$,ET,ES
160 IF EN=0 THEN 210
170 IF EN<>65 THEN PRINT
EN,EM$,ET,ES:STOP

180 IF ET=0 THEN PRINT
"DISK VOLL":STOP
190 IF ET=18 THEN ET=19:
ES=0
```

Öffnen des Befehlskanals
Öffnen des Datenkanals für den Direktzugriff
Text in den Puffer der 1571 schreiben
Erster Block, bei dem die Suche nach einem freien Block beginnen soll
Versuch, den Block als belegt zu kennzeichnen
Auslesen des Fehlerkanals
Wenn Belegung erfolgreich, kein weiteres Suchen
Block bereits belegt
Bei Spur 0 Suche beenden, da Diskette voll
Inhaltsverzeichnis nicht verändern

```
200 T=ET:S=ES:GOTO 140
```
Neue Werte für Spur und Sektor aus Fehlerkanal übernehmen

```
210 PRINT#15,"U2";5;0;T;
S
```
Puffer in den als belegt gekennzeichneten Block schreiben

```
220 PRINT"ABGESPEICHERT
AUF SPUR" T ", SEKTOR"
S
```

```
230 CLOSE 5:CLOSE 15
```
Schließen aller Kanäle

```
240 END
```

7.3.6 FREIGABE VON BLÖCKEN

Der Befehl BLOCK-FREE ist das Gegenstück zu BLOCK-ALLOCATE. Mit ihm wird ein Block freigegeben, der nicht mehr benötigt wird, damit er wieder von DOS benutzt werden kann. Mit BLOCK-FREE wird der entsprechende Block in der BAM freigegeben und steht somit für andere Dateien wieder zur Verfügung. Die Daten selbst werden nicht gelöscht.

FORMAT DES BLOCK-FREE-BEFEHLS:

PRINT#15,"BLOCK-FREE";laufwerk;spur;sektor

In der abgekürzten Form:

PRINT#15,"B-F";laufwerk;spur;sektor

wobei „laufwerk" der Laufwerksnummer entspricht, während durch „spur" und „sektor" Spur und Sektor angegeben werden, die den in den Dateipuffer zu lesenden Datenblock enthalten.

ALTERNATIVFORMAT:

PRINT#15,"B-F:";laufwerk;spur;sektor

BEISPIEL:

Um den Sektor wieder freizugeben, in den der Benutzername in dem Beispiel für BLOCK-WRITE geschrieben und der in dem ersten Beispiel für BLOCK-ALLOCATE als belegt gekennzeichnet worden ist, könnte der folgende Befehl benutzt werden:

PRINT#15,"B-F";0;1;1

7.4 HINWEISE ZUR ARBEIT MIT DIREKTZUGRIFFSDATEIEN

Durch Kombination der Befehle in diesem Kapitel kann ein Programm zur Dateiverarbeitung entwickelt werden, das Direktzugriffsdateien benutzt. Hier müssen Sie nur wissen, wie die einzelnen Blöcke Ihrer Direktzugriffsdatei verfolgt werden können und welche Blöcke auf der Diskette von einer derartigen Datei benutzt werden. (Selbst wenn Sie wissen, daß ein Sektor nicht von der Direktzugriffsdatei belegt wird, müssen Sie sicher sein, daß er nicht von einer anderen, nicht dazugehörigen Datei auf der Diskette belegt wird.)
Im allgemeinen ist es üblich, in einer sequentiellen Datei festzuhalten, welche Sektoren von einer Direktzugriffsdatei belegt werden. In der sequentiellen Datei wird eine Liste mit Eintragsnummern, Spur, Sektor und Byte-Position jedes Eintrags festgehalten. Das hat zur Folge, daß für eine solche Direktzugriffsdatei drei Kanäle benötigt werden: einen für den Befehlskanal, einen für die wahlfreien Daten und schließlich noch einen Kanal für diese sog. sequentielle Indexdatei.

KAPITEL 8

FLOPPY-SYSTEMBEFEHLE

Erfahrene Programmierer können mit bestimmten Befehlen analog zu den BASIC-Befehlen PEEK, POKE, SYS, USR, die sich auf den Rechner beziehen, auch die Arbeitsweise der 1570/71 ändern. Weiterhin ist es möglich, Maschinenprogramme im Speicher der 1570/71 auszuführen, die entweder vom Computer in den Diskettenspeicher geschrieben oder direkt von der Diskette in den gewünschten Puffer des Diskettenspeichers geladen werden. Dies entspricht dem Laden und Ausführen von Programmen in Maschinensprache im Computer.
Wie bei der Benutzung von PEEK-, POKE- und SYS-Befehlen beim Computer muß auch bei der Benutzung der in diesem Kapitel beschriebenen Befehle mit äußerster Vorsicht vorgegangen werden. Hier handelt es sich im wesentlichen um Befehle in Maschinensprache, bei denen alle in BASIC vorhandenen Absicherungen wegfallen. Kommt es zu einer Fehlfunktion, so müssen Sie unter Umständen das Diskettenlaufwerk aus- und wieder einschalten (nachdem Sie die Diskette herausgenommen haben), um die Kontrolle wieder zu erhalten. Experimentieren Sie mit diesen Befehlen nicht bei wichtigen Disketten. Stattdessen wird empfohlen, eine zusätzliche Kopie anzufertigen und mit dieser Kopie zu arbeiten. Wissen Sie, wie ein 6502 in Maschinensprache programmiert wird, so ist dies überaus hilfreich. Außerdem benötigen Sie Informationen über die Speicherbelegung der 1570/71.

8.1 DIE SPEICHERAUFTEILUNG DER 1570/71

Adreßbereich	Zweck
0000–00FF	Zeropage-Arbeitsbereich, Jobwarteschlange, Variablen
0100–01FF	GCR-Überlaufbereich und Stack (BAM/Seite 1 der 1571 im 1571-Modus und BAM der 1570)
0200–02FF	Befehlspuffer, Parser, Tabellen und Variablen
0300–07FF	5 Datenpuffer (0 – 4); davon wird einer für die BAM benutzt
1800–180F	Ports für den seriellen Bus und den Controller
1C00–1C0F	Ports für den Controller
8000–FFE5	32 KByte ROM: DOS und Routinen für den Contoller
FFE6–FFFF	Sprungtabelle

8.2 MEMORY-READ

Die Diskettenstation 1570/71 verfügt über 32 K ROM (Nur-Lese-Speicher) sowie 4 K RAM (Schreib-/Lese-Speicher), von denen nur 2 K benutzt werden. Sie können mit den MEMORY-Befehlen direkt auf jede Position innerhalb des Speichers, auch auf die Puffer, die das DOS im RAM erstellt hat, zugreifen. Mit MEMORY-READ legen Sie die Adresse der Diskettenstation fest, die durch den angeschlossenen Computer gelesen werden soll. Der Befehl MEMORY-READ entspricht gewissermaßen der BASIC-Funktion PEEK, bezieht sich jedoch auf den Speicher der Diskettenstation.

HINWEIS: Im Gegensatz zu anderen Diskettenbefehlen müssen die in diesem Kapitel beschriebenen Befehle abgekürzt werden. So ist M-R richtig, MEMORY-READ jedoch keine zulässige Alternative.

FORMAT DES MEMORY-READ-BEFEHLS:

PRINT#15,"M-R"CHR$(adresse low)CHR$(adresse high)CHR$(anzahl)

wobei „adresse low" den niederwertigen und „adresse high" den höherwertigen Teil der Adresse des Speichers der Diskettenstation darstellt. Wird die Option „anzahl" mit angegeben, so wird damit ausgewählt, wie viele Bytes ab der angegebenen Adresse gelesen werden sollen. Wird der Parameter weggelassen, so wird nur ein Byte gelesen.
Wird nun der Kommandokanal (15) (Fehlerkanal) mit Hilfe der GET#-Anweisung gelesen, so erhält man das Byte von der im M-R-Befehl angegebenen Adresse. Der GET#-Befehl kann zum Auslesen der folgenden Speicheradressen verwendet werden, wenn der wahlweise anzugebende Parameter „anzahl" im „M-R"-Befehl beachtet wird.
Jede INPUT#-Anweisung auf den Fehlerkanal führt bei Benutzung zu unbrauchbaren Ergebnissen.

BEISPIELE:

Mit den folgenden Programmzeilen kann festgestellt werden, wie viele Versuche das DOS unternimmt, um einen bestimmten Sektor auf der Diskette zu lesen, ob eine halbe Spur auf jeder Seite durchsucht wird, wenn der Lesevorgang nicht erfolgreich ausgeführt werden kann, und ob zu Spur 1 und wieder zurück „gesprungen" wird, bevor der Sektor als unleserlich erklärt wird. Mit ihnen wird eine besondere Variable in die Zeropage des Diskettenspeichers gelesen, die als REVCNT bezeichnet wird. Sie besteht bei Adresse $006A (hexadezimal) im Speicher der Diskettenstation.

```
110 OPEN 15,8,15
120 PRINT#15,"M-R";CHR$(106);CHR$(0)
130 GET#15,G$:IF G$="" THEN G$=CHR$(0)
140 G=ASC(G$)
150 B=G AND 128:B$="EINGESCHALTET":IF B
THEN B$="AUSGESCHALTET"
160 S=G AND 64:S$="":IF S THEN S$="NICHT
 "
170 T=G AND 63:PRINT "ANZAHL DER VERSUCH
E:";T
180 PRINT "BUMPS SIND ";B$;"."
190 PRINT "LESEVERSUCHE NEBEN DER SPUR W
ERDEN ";S$;"DURCHGEFUEHRT."
200 CLOSE 15
210 END
```

```
110 OPEN 15,8,15
120 INPUT "ANZAHL DER ZU LESENDEN BYTES
(0=ENDE)";NL
130 IF NL<1 THEN CLOSE 15:END
140 IF NL>255 THEN 120
150 INPUT "STARTADRESSE";AD
160 AH=INT(AD/256):AL=AD-AH*256
170 PRINT#15,"M-R";CHR$(AL);CHR$(AH);CHR
$(NL)
180 FOR I=1 TO NL
190 GET#15,A$:IF A$="" THEN A$=CHR$(0)
200 PRINT ASC(A$);
210 NEXT I
220 PRINT
230 GOTO 120
```

8.3 MEMORY-WRITE

Der MEMORY-WRITE-Befehl ist das Äquivalent des BASIC-Befehls POKE, wirkt sich jedoch auf den Diskettenspeicher und nicht auf den Computer aus. Mit M-W können maximal 34 Bytes gleichzeitig in den Diskettenspeicher geschrieben werden. Mit MEMORY-EXECUTE und einigen USER-Befehlen können auf diese Weise geschriebene Programme ausgeführt werden.

FORMAT DES MEMORY-WRITE-BEFEHLS:

PRINT#15,"M-W"CHR$(adresse low)CHR$(adresse high)CHR$(anzahl) CHR$(datenbyte(s))

„adresse low" entspricht dem niederwertigen und „adresse high" dem höherwertigen Teil der Adresse im Diskettenspeicher, bei der mit dem Schreiben begonnen werden muß. „anzahl" entspricht der Anzahl der Speicherpositionen, die geschrieben werden (von 1 bis 34). „datenbyte(s)" entspricht einem bzw. mehreren Bytes, die jeweils als einzelne CHR$-Werte im Befehl angegeben werden müssen.

BEISPIELE:

Mit diesem Befehl können „Bumps" abgeschaltet werden, wenn DOS-geschützte Programme geladen werden (d.h. Programme, die vor dem Kopieren geschützt wurden, indem bestimmte Diskettenfehler erstellt und geprüft wurden).

PRINT#15,"M-W"CHR$(106)CHR$(0)CHR$(1)CHR$(133)

Mit dem folgenden Befehl können fehlerhafte Sektoren wiederhergestellt werden, beispielsweise wenn eine umfangreiche Datei beschädigt wurde und nicht normal gelesen werden kann.

PRINT#15,"M-W"CHR$(106)CHR$(0)CHR$(1)CHR$(31)

Diese beiden Beispiele sind unter bestimmten Umständen besonders nützlich. Sie entsprechen gewissermaßen den BASIC-Befehlen POKE 106,133 bzw. POKE106,31, beziehen sich jedoch auf den Disketten- und nicht auf den Computerspeicher. Wie bereits aus dem ersten Beispiel des vorhergehenden Abschnitts hervorgeht, werden durch Adresse 106 (dez.) im Floppyspeicher drei Optionen für das Laufwerk aktiviert, die sich alle auf die Fehlerbehandlung beziehen. Ist Bit 7 (das höchstwertige Bit) gesetzt, so werden keine „Bumps" durchgeführt, d.h. im Fehlerfall fährt der Schreib-/Lesekopf nicht mehr gegen den Anschlag vor Spur 1 und versucht, den angewählten Block anschließend erneut zu lesen. Ist Bit 6 gesetzt, so hat dies zur Folge, daß im Fehlerfall nicht mehr versucht wird, die an die gewählte Spur ober- und unterhalb angrenzenden sog. Halbspuren zu lesen. Mit den unteren 6 Bits wird angegeben, wie oft das DOS versucht, jeden Sektor vor und nach der Suche und den „Bumps" zu lesen, bevor ein Fehler gemeldet wird. Da 63 die größte Zahl ist, die mit 6 Bits ausgedrückt werden kann, entspricht dies der Höchstzahl der möglichen Versuche. (Der Befehl PRINT#15,"U0>R"CHR$(wert) kann übrigens diesen o.g. M-W-Befehl ersetzen.)
Anhand dieses Beispiels wird deutlich, wie wichtig es ist, Erfahrungen mit den PEEK- und POKE-Befehlen und der Maschinensprache zu haben, bevor Diskettenbefehle für den Direktzugriff in ihrer ganzen Leistungsfähigkeit verwendet werden können.

8.4 MEMORY-EXECUTE

Jede Routine im Diskettenspeicher, d. h. im RAM oder im ROM, kann mit dem MEMORY-EXECUTE-Befehl ausgeführt werden. Dieser Befehl ist das Äquivalent zum BASIC-SYS-Aufruf eines Programms oder einer Subroutine in Maschinensprache, wird jedoch im Diskettenspeicher und nicht innerhalb des Computers ausgeführt.

FORMAT DES MEMORY-EXECUTE-BEFEHLS:

```
PRINT#15,"M-E"CHR$(adresse low)CHR$(adresse high)
```

wobei „adresse low" dem niederwertigen und „adresse high" dem höchstwertigen Teil der Adresse im Diskettenspeicher entspricht, bei der die Ausführung beginnen soll.

BEISPIEL:

Nachfolgend ein MEMORY-EXECUTE-BEFEHL, der absolut nichts bewirkt. Die erste ausgeführte Instruktion ist eine RTS-Instruktion, die den Befehl beendet:

```
PRINT#15,"M-E"CHR$(179)CHR$(242)
```

Ein plausiblerer Grund für die Benutzung dieses Befehls besteht in der künstlichen Auslösung einer Fehlermeldung. Vergessen Sie aber nicht, den Fehlerkanal abzufragen.

```
PRINT#15,"M-E"CHR$(201)CHR$(239)
```

In den meisten Fällen ist zur Benutzung dieses Befehls eine genaue Kenntnis der internen Arbeitsweise des DOS erforderlich. Außerdem ist dieser Befehl in der Regel zusammen mit anderen Befehlen, wie beispielsweise MEMORY-WRITE, zu verwenden.

8.5 BLOCK-EXECUTE

Dieser selten benutzte Befehl lädt einen Sektor mit einer Routine in Maschinensprache von der Diskette in den Puffer der 1570/71 und führt ihn ab der ersten Position innerhalb des Puffers aus, bis eine RETURN-FROM-SUBROUTINE (RTS-)Instruktion diesen Befehl beendet.

FORMAT DES BLOCK-EXECUTE-BEFEHLS:

PRINT#15,"B-E";kanal;laufwerk;spur;sektor

„kanal" entspricht der Kanalnummer, die beim Öffnen der Datei angegeben wurde, in die der Block geladen wird. „laufwerk" entspricht der Laufwerksnummer, während „spur" und „sektor" den Spur- bzw. Sektornummern entsprechen, die den Datenblock enthalten, der in den Dateipuffer geladen und dort als Programm ausgeführt werden soll.

ALTERNATIVFORMATE:

PRINT#15,"B-E:";kanal;laufwerk;spur;sektor
PRINT#15,"B-E kanal laufwerk spur sektor"

BEISPIELE:

Angenommen, Sie haben ein Programm in Maschinensprache auf Spur 1, Sektor 8 einer Diskette geschrieben und möchten es in dem Puffer 1 des Diskettenspeichers ab Adresse '$0400' ausführen. Hierzu könnten Sie folgendes Programm benutzen:

```
110 OPEN 15,8,15
120 OPEN 2,8,2,"#1"

130 PRINT#15,
"B-E:";2;0;1;8
140 CLOSE 2
150 CLOSE 15
160 END
```

Öffnet den Befehlskanal.
Öffnet den Direktzugriffskanal für Puffer 1.
Lädt Spur 1, Sektor 8 in den Puffer und führt den Inhalt als Programm aus.
Schließen der Kanäle.

8.6 USER-BEFEHLE

Die meisten USER-Befehle sollten als JMP- oder BASIC-SYS-Befehle in Maschinensprache für Programme in Maschinensprache benutzt werden, die im Speicher der 1570/71 stehen. Einige dieser Befehle werden jedoch auch noch zu anderen Zwecken benutzt. So „ersetzen" beispielsweise die Befehle U1 und U2 die Befehle BLOCK-READ und BLOCK-WRITE, UI startet die 1570/71 erneut, ohne die Variablen zu ändern („Warmstart"), und UJ nimmt einen Kaltstart der 1570/71 vor, als wäre sie aus- und danach wieder eingeschaltet worden.

USER-Befehl	Funktion
U0	Stellt die Standardsprungtabelle des Benutzers wieder her.
U1 oder UA	„Ersatz" für Block-Read.
U2 oder UB	„Ersatz" für Block-Write.
U3 oder UC	Sprung nach $0500.
U4 oder UD	Sprung nach $0503.
U5 oder UE	Sprung nach $0506.
U6 oder UF	Sprung nach $0509.
U7 oder UG	Sprung nach $050c.
U8 oder UH	Sprung nach $050f.
U9 oder UI	Sprung nach ($FFFA) Rücksetztabellen.
U: oder UJ	RESET-Vektor.
U0 > Mm	Modus: m: 0 für 1541, 1 für 1570 bzw. 1571
U0 > Hs	Diskettenseite (nur im 1541-Modus und nur bei der 1570 s: 0 = unten, 1 = oben)
U0 > dv	Geräteadresse, dv = CHR$(4) . . . CHR$(30)

Werden diese Speicherpositionen mit einem anderen JMP-Befehl in Maschinensprache geladen, wie beispielsweise JMP $0520, so können längere Routinen erstellt werden, die im Diskettenspeicher zusammen mit einfach zu benutzenden Sprungtabellen ausgeführt werden können.

FORMAT DER USER-BEFEHLE:

PRINT#15,"Uzeichen"

wobei durch „zeichen" einer der oben aufgeführten USER-Befehle definiert wird.

BEISPIELE:

PRINT#15,"U:"	**Form des DOS-Befehls RESET.**
PRINT#15,"U3"	**Führt das Programm am Anfang von Puffer 2 aus.**

8.7 SYSTEMPROGRAMME IN DATEIEN

Mit diesem Befehl wird eine Benutzerdatei in das RAM des Laufwerks geladen. Die beiden ersten Bytes der Datei müssen den höherwertigen bzw. niederwertigen Teil der Adresse enthalten. Mit dem dritten Byte wird angegeben, wie viele Zeichen folgen. Darüber hinaus ist ein abschließendes Prüfsummen-Byte anzugeben. Die Ladeadresse ist gleichzeitig die Startadresse.

BEFEHLSFORMAT ZUM LADEN VON SYSTEMPROGRAMMEN

```
PRINT#"&0:dateiname"
```

8.8 MASCHINENPROGRAMME ZUR BEDIENUNG DER 1571

Nachfolgend eine Liste der diskettenbezogenen ROM-residenten Kernal-Subroutinen und ein praktisches Beispiel für ihre Verwendung durch ein Programm, das eine sequentielle Datei von der Diskette des angeschlossenen Rechners einliest und auf dem Bildschirm ausgibt. Hier wird darauf hingewiesen, daß die meisten solcher Programme die Installation auf ein oder mehrere Prozessorregister bzw. Speicherpositionen voraussetzen und daß alle mit dem JSR-Befehl in der Assemblersprache aufgerufen werden.
Für eine genauere Erläuterung der Funktion jeder Routine und der Festlegung der Parameter für jede Routine wird auf das Programmierhandbuch für den jeweiligen Computer verwiesen.

DISKETTENBEZOGENE KERNAL-UNTERPROGRAMME

Label		Adresse	Funktion
SETLFS	=	$FFBA	;Dateiparameter festlegen
SETNAM	=	$FFBD	;Dateinamen und -länge festlegen
OPEN	=	$FFCO	;Datei öffnen
CLOSE	=	$FFC3	;Datei schließen
CHKIN	=	$FFC6	;Kanal für Eingabe öffnen
CHKOUT	=	$FFC9	;Kanal für Ausgabe öffnen
CLRCHN	=	$FFCC	;Kanäle schließen und Standard einst.
CHRIN	=	$FFCF	;Byte vom definierten Kanal einlesen
CHROUT	=	$FFD2	;Byte auf definierten Kanal ausgeben

```
;
START   LDA   #4          ;Länge, ...
        LDX   #<FNADR     ;... Adresse (low) und
        LDY   #>FNADR     ;... Adresse (high) für SETNAM
        JSR   SETNAM      ;Dateinamen definieren
        LDA   #3          ;logische Dateinummer ...
        LDX   #8          ;... Primäradresse und ...
        LDY   #0          ;... Sekundäradresse ...
        JSR   SETLFS      ;... festlegen
        JSR   OPEN        ;OPEN 3,8,0, "TEST"
        LDX   #3          ;Wahl der logischen Datei Nr. 3 ...
        JSR   CHKIN       ;... als Eingabedatei
NEXT    JSR   CHRIN       ;nächstes Byte aus der Datei lesen
        BEQ   END         ;Schleife bis Dateiende oder -fehler
        JSR   CHROUT      ;ein Byte auf dem Bildschirm ausgeben
        JMP   NEXT        ;Schleife
;
END     LDA   #3          ;logische Datei Nr. 3 ...
        JSR   CLOSE       ;... schließen
        JSR   CLRCHN      ;Standard-I/O wiederherstellen
        RTS               ;Rückkehr zu BASIC
;
FNADR   .BYTE "TEST"      ;Dateiname
```

8.9 DIE BURSTBEFEHLE

Die neuen Diskettenlaufwerke 1570 und 1571 besitzen gegenüber den herkömmlichen Laufwerken einen erweiterten Befehlssatz, der nicht nur die Handhabung vereinfacht, sondern auch beispielsweise Kommandos zum schnellen Laden oder speziellen Formatieren bereitstellt.

8.9.1 DIE EINZELNEN BEFEHLE

8.9.1.1 READ

Das Burstkommando READ ermöglicht das Lesen eines oder mehrerer Sektoren mit schneller Datenübertragung zum Rechner.

READ

BYTE	BIT 7	6	5	4	3	2	1	0
00	0	1	0	1	0	1	0	1
01	0	0	1	1	0	0	0	0
02	T	E	B	S	0	0	0	N
03	DESTINATION TRACK							
04	DESTINATION SECTOR							
05	NUMBER OF SECTORS							
06	NEXT TRACK (OPTIONAL)							

Bereich: Alle Werte werden durch das jeweilige Diskettenformat bestimmt.

Flags:

T – Daten senden	(1 = keine Übertragung)
E – Fehler ignorieren	(1 = ignorieren)
B – nur Pufferübertragung	(1 = nur Pufferübertragung)
S – Diskettenseite	(nur bei MFM, immer 0 bei 1570)
N – Laufwerksnummer	(immer 0)

Protokoll: Burst Handshake

Hinweis: Bevor das BURST-READ- oder -WRITE-Kommando angewendet werden kann, muß mit dem Befehl INQUIRE DISK oder QUERY DISK FORMAT eine Anmeldung erfolgen. (Beide Befehle werden später beschrieben.) Dies braucht nur einmal, wenn die Diskette gewechselt wird, durchgeführt werden.

Ergebnis: Es wird ein Burst-Statusbyte, auf das die Daten folgen, für jeden Sektor übertragen. Tritt ein Fehler auf, wird die Datenübertragung abgebrochen, wenn das E-Flag nicht gesetzt wurde.

8.9.1.2 WRITE

Das Burstkommando WRITE dient dem schnellen Abspeichern von Daten auf mehrere Sektoren der Diskette.

WRITE

BYTE	BIT 7	6	5	4	3	2	1	0
00	0	1	0	1	0	1	0	1
01	0	0	1	1	0	0	0	0
02	T	E	B	S	0	0	1	N
03	DESTINATION TRACK							
04	DESTINATION SECTOR							
05	NUMBER OF SECTORS							
06	NEXT TRACK (OPTIONAL)							

Bereich: Alle Werte sind von dem entsprechenden Diskettenformat abhängig.

Flags:

T – Daten übertragen	(1 = keine Übertragung)
E – Fehler ignorieren	(1 = ignorieren)
B – nur Pufferübertragung	(1 = nur Pufferübertragung)
S – Diskettenseite	(nur bei MFM)
N – Laufwerksnummer	(immer 0)

Protokoll: Besonderes Verfahren, siehe Beispiel.

Hinweis: Bevor die Burstkommandos READ oder WRITE ausgeführt werden können, ist mit Hilfe der Befehle INQUIRE DISK oder QUERY DISK FORMAT (wird später beschrieben) die Diskette anzumelden. Dies ist nach jedem Wechsel der Diskette nur ein Mal erforderlich.

Eingabe: Der Rechner hat die Burstdaten übertragen.

Ausgabe: Auf jede WRITE-Operation folgt ein Burst-Statusbyte.

8.9.1.3 INQUIRE DISK

Das INQUIRE-DISK-Kommando wird nach dem Wechsel einer Diskette vor der Anwendung von READ und WRITE benötigt.

INQUIRE DISK

BYTE	BIT 7	6	5	4	3	2	1	0
00	0	1	0	1	0	1	0	1
01	0	0	1	1	0	0	0	0
02	X	X	X	S	0	1	0	N

Flags: S – Diskettenseite (nur bei MFM, immer 0 bei 1570)
N – Laufwerksnummer (immer 0)

8.9.1.4 FORMAT MFM

Die Funktion FORMAT MFM ermöglicht das Formatieren von Disketten in verschiedenen MFM-Formaten.

FORMAT MFM

BYTE	BIT 7	6	5	4	3	2	1	0
00	0	1	0	1	0	1	0	1
01	0	0	1	1	0	0	0	0
02	P	I	D	S	0	1	1	N
03	M = 1	T						
04								
05								
06								
07								
08								
09								
0A								
0B								

Flags:

P – partielles Format	(1 =	partiell)
I – Indexadreßmarke	(1 =	Schreiben)
D – Zwei-Seiten-Flag	(1 =	doppelseitig formatieren)
S – Diskettenseite		
T – Sektortabelle vorhanden	(1 =	Tabelle vorhanden, alle anderen Parameter sind beizufügen)
N – Laufwerknummer	(immer 0)	

Protokoll: normal

Hinweise: Diesem Kommando muß entweder das INQUIRE-DISK- oder der QUERY-DISK-FORMAT-Befehl folgen.

Die Parameter D und S können von der 1570 nicht verwendet werden.

Ergebnis: Kein Burst-Statusbyte – der Status wird mit der Floppystatusmeldung bereitgestellt.

8.9.1.5 FORMAT GCR (ohne Inhaltsverzeichnis)

Das Kommando FORMAT GCR führt eine GCR-Formatierung der Diskette durch. Das GCR-Format (GCR = group code recording) ist das Standardaufzeichnungsformat bei der Commodore-Diskettenstation 1570 bzw. 1571. Bei Verwendung des hier beschriebenen Befehls wird die in das entsprechende Laufwerk eingelegte Diskette nur formatiert und nicht zusätzlich noch – wie bei dem NEW-Befehl – für den Normalgebrauch durch das Schreiben der BAM und des Inhaltsverzeichnisses vorbereitet.

FORMAT GCR (NO DIRECTORY)

BYTE	BIT 7	6	5	4	3	2	1	0
00	0	1	0	1	0	1	0	1
01	0	0	1	1	0	0	0	0
02	X	X	X	X	0	1	1	N
03	M=0							
04			ID LOW					
05			ID HIGH					

Flags: N – Laufwerksnummer
X – ohne Bedeutung (kann 0 oder 1 sein)

Protokoll: Normal

Hinweis: Dieser Befehl darf nur nach Durchführung von INQUIRE DISK oder QUERY DISK verwendet werden.

Rückmeldung: Keine – der Status wird mit der Floppystatusmeldung bereitgestellt.

8.9.1.6 SECTOR INTERLEAVE

Der Befehl SECTOR INTERLEAVE dient zur Festlegung des Sektorversatzes beim Burst-Read und -Write.

SECTOR INTERLEAVE

BYTE	BIT 7	6	5	4	3	2	1	0
00	0	1	0	1	0	1	0	1
01	0	0	1	1	0	0	0	0
02	W	X	X	0	1	0	0	N
04	INTERLEAVE							

Flags: W – Flag für Schreiben
N – Laufwerksnummer (immer 0)
X – ohne Bedeutung

Protokoll: Burst-Handshake

Hinweis: Dieser Befehl dient nur zur Unterstützung der Burst-Befehle READ und WRITE bei Verarbeitung mehrerer Sektoren.

Rückmeldung: Keine bei W = 0, Interleave-Burst-Byte bei W = 1.

8.9.1.7 QUERY DISK FORMAT

Das Kommando "QUERY DISK FORMAT" dient der Untersuchung von Diskettenformaten oder zum Aktivieren von Fremdformaten.

QUERY DISK FORMAT

BYTE	BIT 7	6	5	4	3	2	1	0
00	0	1	0	1	0	1	0	1
01	0	0	1	1	0	0	0	0
02	F	X	X	S	1	0	1	N
03	OFFSET				(OPTIONAL F-BIT SET)			

Flags:	F – einzelne Spur	(bei F = 1 wird der in Byte 3 angegebene Offset verwendet; F = 0 entspricht Offset = 0)
	N – Laufwerksnummer	(immer 0)
	X – ohne Bedeutung	(darf 0 oder 1 sein)

Protokoll: Burst-Handshake

Hinweis: Neben der Untersuchung von Diskettenformaten kann dieses Kommando auch vor dem Lesen bzw. Beschreiben von Fremdformaten zur Einstellung des Disk-Controlers verwendet werden.

Rückmeldung: Burst-Status Byte, keine weiteren Informationen bei GCR-Format oder Lesefehlern.
Bei MFM erhält man folgende weiteren Informationen:
- Anzahl der Sektoren pro Spur
- Logische Nummer der Spur (aus dem Header)
- Minimale Sektornummer
  (die kleinste vorhandene logische Sektornummer auf der Spur)
- Maximale Sektornummer
  (die größte logische Nummer der Sektoren auf der Spur)

8.9.1.8 INQUIRE STATUS

Mit Hilfe der Funktion INQUIRE STATUS kann der Status abgefragt oder neu vorgegeben werden. Weiterhin kann die 1570 bzw. mit dieser Funktion auf eine Diskette vorbereitet werden (sog. Anmeldung), dies ist insbesondere nach dem Wechsel einer Diskette in MFM-Formaten erforderlich.

INQUIRE STATUS

BYTE	BIT 7	6	5	4	3	2	1	0
00	0	1	0	1	0	1	0	1
01	0	0	1	1	0	0	0	0
02	W	C	X	0	1	1	0	N
03	NEW STATUS (W-BIT CLEAR)							

Flags: W – Flag für Schreiben
C – Flag für Diskettenwechsel
N – Laufwerknummer (immer 0)
X – Ohne Bedeutung

Hinweis: C = 1 u. W = 0: Diskette anmelden
C = 1 u. W = 1: Abfrage, ob Diskette angemeldet ist

Protokoll: Burst-Handshake bei W = 1

Rückmeldung: Keine bei W = 0, Burst-Status-Byte bei W = 1.

8.9.1.9 BACKUP DISK

Dieser Befehl kann von den Einzellaufwerken 1570 und 1571 nicht verarbeitet werden.

8.9.1.10 CHGUTL UTILITY

Es handelt sich hierbei um ein "UTILITY-Paket" nützlicher DOS-Routinen, die auch von BASIC aus einfach verwendet werden können:

CHGUTL UTILITY

BYTE	BIT 7	6	5	4	3	2	1	0
00	0	1	0	1	0	1	0	1
01	0	0	1	1	0	0	0	0
02	X	X	X	1	1	1	1	0
03	UTILITY COMMANDS: 'S', 'R', 'T', 'M', 'H', #DEV							
04	COMMAND PARAMETER							

Die in der obigen Tabelle mit "X" gekennzeichneten Flags sind ohne Bedeutung und können beliebig gesetzt werden. Daher lassen sich alle CHGUTL-UTILITY-Befehle beispielsweise mit folgenden BASIC-Kommandos zum Laufwerk übertragen:

```
OPEN15,8,15: PRINT#15,"U0>" cm$; pa$
```

Hierbei ist 'cm$' einer der in der Tabelle aufgeführten Befehle mit folgender Bedeutung:

"S"	– DOS-Sektorversatz festlegen
"R"	– Anzahl der Disketten-Leserversuche durch das DOS
"T"	– ROM-Test
"M"	– Wahl der Betriebsart (0 = 1541-Modus, 1 = 1570- bzw. 1571-Modus)
"H"	– Wahl der Diskettenseite (0 = Seite 0, 1 = Seite 1 (nur bei der 1571))
CHR$(device)	– Festlegung der Geräteadresse ('device' von 4 bis 30)

8.9.1.11 FASTLOAD UTILITY

Diese Routine ermöglicht das schnelle Laden von Dateien. Hierzu ist folgender Befehlscode an die 1570 bzw. 1571 zu übermitteln:

FASTLOAD UTILITY

BYTE	BIT 7	6	5	4	3	2	1	0
00	0	1	0	1	0	1	0	1
01	0	0	1	1	0	0	0	0
02	P	X	X	1	1	1	1	1
03-??	FILE NAME							

Flags: P – Flag für Dateityp
(0 = Es werden nur PRG-Dateien übertragen)
(1 = Es werden auch andere Dateitypen übertragen)
X – ohne Bedeutung

Protokoll: Burst

Ergebnis: Jeweils ein Burststatusbyte vor jedem übertragenen Sektor

Status: %0000000X – ok
%00000010 – Datei nicht gefunden
%00011111 – Ende der Übertragung

Jedes andere Burststatusbyte ist als Lesefehler auf der Diskette zu interpretieren.

8.9.2 Die Anwendung der Burstbefehle

Bei der Anwendung der Burstbefehle sind bestimmte Regeln zu beachten, die in den nächsten Abschnitten beschrieben werden. Es gibt zwei Arten von Burstbefehlen: einfache, bei denen kein Burstprotokoll erforderlich ist (z.B. die 'CHGUTL UTILITIES', s. Kap. 8.9.1.10) und solche, bei denen ein Burstprotokoll erforderlich ist (z.B. READ und WRITE). Bei dieser Gruppe erhält man zusätzlich das sog. Burststatusbyte, welches Informationen über die Durchführung des Befehls beinhaltet.

8.9.2.1 Das Burstübertragungsprotokoll

Bevor die Burstübertragungsroutinen verwendet werden können, ist sicherzustellen, daß die zu dem Rechner angeschlossene Diskettenstation über die Burstübertragungsmöglichkeiten verfügt. Diese Unterscheidung wird vom schnellen seriellen Übertragungsprotokoll (Byte-Modus) intern bewirkt, wenn eine Überprüfungsroutine (send-cmd-string) verwendet wird. Diese Routine spricht die Diskettenstation an und stellt ihre Übertragungsgeschwindigkeit ein.

Folgende Abläufe sind bei Burst-Read und -Write einzuhalten:

Burst-Read: Burst-Write: Erläuterung der Einzelnen Prozeduren

BURST READ

```
send-cmd-string;                    (*determine speed*)
if device-fast then
    serial-in;                      (*turn 6526 to input*)
    repeat                          (*repeat for all sectors*)
        read-error;                 (*retrieve error byte*)
        toggle-clock;               (*no error*)
            repeat                  (*repeat for all sectors*)
                wait-byte;          (*poll 6526 for byte*)
                toggle-clock;       (*toggle clock*)
                store-data;         (*save data*)
        until last-byte;            (*last byte ?*)
    until last-sector;              (*any more sectors ?*)
    set-clock-high;                 (*release clock line*)
else
    read-1541;                      (*send unit read*)
```

BURST WRITE

```
send-cmd-string;                        (*determine speed*)
if device-fast then
    repeat                              (*repeat for multi-sector*)
        serial-out;                     (*serial port out*)
            repeat                      (*repeat for sector-size*)
                check-clock;            (*clock toggle ?*)
                send-byte;              (*send byte*)
            until last-byte;            (*last byte ?*)
        serial-in;                      (*serial port in*)
        clock-low;                      (*ready for status*)
        read-err;                       (*controller error ?*)
        clock-high;                     (*restore clock*)
    until last-sector;                  (*until last sector*)
else
    write-1541;                         (*unit write*)
```

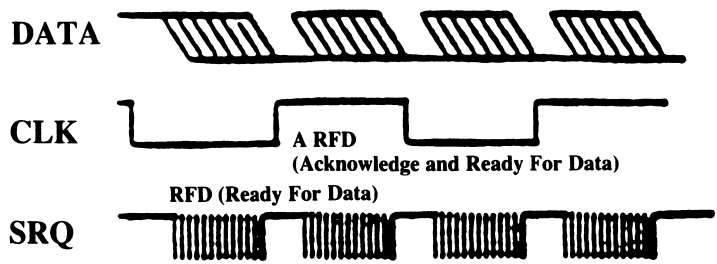
EXPLANATION OF PROCEDURES

send-cmd-string	sends one byte of the command to determine whether the drive is fast or slow.
toggle-clock	changes the state of the clock line.
clock-hi	changes the state of the clock to logic 1.
clock-lo	changes the state of the clock to logic 0.
wait-byte	polls the 6526 for a byte ready.
read-error	calls toggle-clock and wait-byte, then returns to the main if there are no errors.
store-data	stores the data in a particular memory location.
last-byte	depending on sector size, will increment and compare value to sector size.
last-sector	decrements the number of sector transfers requested and stops when done.
serial-in	sets the 6526 serial port and driver circuit to input mode.
read-err	calls wait-byte and evaluates the status of the previous controller job.

serial-out	sets the 6526 serial port and driver circuit to output mode.
check-clock	checks the status of the clock line and returns upon toggling of the clock line.
send-byte	sends a byte of data to the 1571.
read-1541	sends a typical unit read to a 1541.
write-1541	sends a typical unit write to a 1541.

8.9.2.2 Burst-Handshake

Die folgende Abbildung stellt das Burstübertragungsprotokoll dar. Es handelt sich dabei um ein statusabhängiges Protokoll (einfach und schnell). Bei jeder Flanke des Taktes (clock) kann ein Datenbyte gesendet werden. Das Burstübertragungsprotokoll kann in 3 Phasen aufgeteilt werden:



1. Kommando senden. Eine Zeichenkette wird unter Verwendung von KERNAL-Programmen gesendet.

2. Überprüfung, ob die Diskettenstation für die Burstübertragung geeignet ist.

3. Handshake. Datenübertragung unter Beachtung besonderer Regeln.

8.9.2.3 Das Burststatusbyte

Die einzelnen Bits im Statusbyte haben folgende Bedeutung:

Bit	7	6	5	4	3	2	1	0
	Modus	Laufwerks-nummer	Sektorgröße		Status des Disk-Controllers			

Beim Modus (Bit 7) bedeutet: 0 = MFM, 1 = GCR

Die Laufwerksnummer (Bit 6) ist bei den Einzellaufwerken 1570 und 1571 immer Null.

Die Sektorgröße (Bits 5 und 4) – sie ist nur bei MFM von Bedeutung – ist der folgenden Tabelle zu entnehmen:

00 – 128 Bytes/Sektor
01 – 256 Bytes/Sektor
10 – 512 Bytes/Sektor
11 – 1024 Bytes/Sektor

Die Bits 3 bis 0 geben den Disk-Controller-Status wieder. Hierbei ist zwischen GCR und MFM zu unterscheiden:

Bit	Bedeutung des Disk-Controller-Status bei	
3210	GCR	MFM
000X	ok	ok
0010	Sektor nicht gefunden	Sektor nicht gefunden
0011	kein SYNC	keine Adreßmarke
0100	Datenblock nicht gefunden	(reserviert)
0101	Prüfsummenfelder im Datenblock	CRC-Datenfehler
0110	Formatfehler	Formatfehler
1000	Schreibschutz	Schreibschutz
1001	Prüfsummenfehler im HEADER	Prüfsummenfehler im HEADER
1010	Daten reichen in den nächsten Block	(reserviert)
1011	Disketten-ID-Fehler, Diskettenwechs.	Diskettenwechsel
1100	reserviert	(reserviert)
1101	reserviert	(reserviert)
1110	Syntaxfehler	Syntaxfehler
1111	Laufwerk nicht vorhanden	Laufwerk nicht vorhanden

8.9.2.4 Beispiele zur Verwendung der Burstroutinen

```
READY.

1000 ;DER BURSTBEFEHL 'READ'
1010 ;
1020 ;BEISPIELPROGRAMM ZUM LESEN VON N DATENBLOECKEN
1030 ;
1040 N       = 3              ;ANZAHL DER ZU LESENDEN SEKTOREN
1050 ;
1060 LOCAT   = $2000          ;BEGINN
1070 ;
1080 SETLFS  = $FFBA          ;DATEIPARAMETER FESTLEGEN
1090 SETNAM  = $FFBD          ;DATEINAMEN DEFINIEREN
1100 OPEN    = $FFC0          ;DATEI OEFFNEN
1110 CLOSE   = $FFC3          ;DATEI SCHLIESSEN
1120 CHKOUT  = $FFC9          ;AUSGABEKANAL WAEHLEN
1130 CLRCHN  = $FFCC          ;ALLE KANAELE FREIGEBEN
1140 BSOUT   = $FFD2          ;EIN BYTE AUSGEBEN
1150 ;
1160 DLSDR   = $DC0C          ;DATENREGISTER FUER SERIELLE UEBERTRAGUNG
1170 DLICR   = $DC0D          ;INTERRUPT-KONTROLLREGISTER
1180 D2PRA   = $DD00          ;DATENREGISTER VON CIA 2
1190 ;
1200 SERIAL  = $0A1C          ;FLAG FUER SCHNELLE SERIELLE UEBERTRAGUNG
1210 ;
1220 STAT    = $FA            ;ZWISCHENSPEICHER FUER BURSTSTATUS
1230 BUFFAD  = $0400
1240 BUFFER  = $FB            ;PUFFER ZUM SCHREIBEN ($FB,$FC)
1250 ;
1260 ;
1270 SCSIZE  = $01            ;SEKTORGROESSE (256 BYTES/SEKTOR)
1280 ;
1290 *       = LOCAT
1300 CMDBUF  .BYTE 'U0',$00,1,1,N,2
1310 CMDLG   = 7
1320 ;
1330         LDA #<BUFFAD     ;PUFFER
1340         STA BUFFER       ;BEREITSTELLEN
1350         LDA #>BUFFAD     ;
1360         STA BUFFER+1     ;
1370 ;
1380         LDA #15          ;LOGISCHE DATEINUMMER
1390         LDX #8           ;GERAETEADRESSE
1400         LDY #15          ;SEKUNDAERADRESSE
1410         JSR SETLFS       ;DATEIPARAMETER DEFINIEREN
1420 ;
1430         LDA #0           ;KEIN DATEINAME
1440         JSR SETNAM       ;DATEINAMEN DEFINIEREN
1450 ;
1460         JSR OPEN         ;DATEI OEFFNEN
1470 ;
1480 READ    LDA #$00
1490         STA STAT         ;ALTEN STATUS LOESCHEN
1500         LDA SERIAL
1510         AND #%10111111   ;FLAG FUER SCHNELLE UEBERTRAGUNG LOESCHEN
1520         STA SERIAL
1530 ;
1540         LDX  #15
1550         JSR  CHKOUT      ;AUSGABEKANAL OEFFNEN
1560 ;
1570         LDX #0
1580         LDY CMDLG        ;LAENGE DES BEFEHLS
1590 SENDIT  LDA CMDBUF,X     ;BEFEHL HOLEN
1600         JSR BSOUT        ;BEFEHL SENDEN
1610         INX
```

```
1620          DEY
1630          BNE SENDIT
1640 ;
1650          JSR CLRCHN        ;ALLE KANAELE FREIGEBEN
1660          BIT SERIAL        ;UEBERTRAGUNGSGESCHWINDIGKEIT ERMITTELN
1670          BVC ERROR         ;NORMALE UEBERTRAGUNGSRATE
1680 ;
1690          SEI
1700          BIT DLICR
1710          LDX CMDBUF+5      ;ANZAHL DER SEKTOREN HOLEN
1720          LDA D2PRA         ;DATENBYTE VOM SERIELLEN PORT HOLEN
1730          EOR #%00010000    ;TAKTPEGEL WECHSELN
1740          STA D2PRA         ;ZURUECKSPEICHERN
1750 ;
1760 READIT   LDA #8
1770 WAIT1    BIT DLICR         ;BURSTSTATUSBYTE ERWARTEN
1780          BEQ WAIT1         ;WARTEN, BIS EMPFANGEN
1790 ;
1800          LDA D2PRA         ;SERIELLEN PORT LESEN
1810          EOR #%00010000    ;TAKTPEGEL WECHSELN
1820          STA D2PRA         ;ZURUECKSCHREIBEN
1830 ;
1840          LDA DLSDR         ;STATUSBYTE LESEN
1850          STA STAT          ;UND ABSPEICHERN
1860          AND #%00001111    ;DISK-CONTROLLER-STATUS HERAUSMASKIEREN
1870          CMP #2            ;FEHLERKONTROLLE: SEKTOR GEFUNDEN?
1880          BCS ERROR
1890 ;
1900          LDY #0
1910          LDA #8
1920 WAIT2    BIT DLICR         ;AUF ZU EMPFANGENDES BYTE WARTEN
1930          BEQ WAIT2
1940 ;
1950 TOPRD    LDA D2PRA         ;SERIELLEN PORT LESEN
1960          EOR #%00010000    ;TAKTPEGEL WECHSELN
1970          STA D2PRA         ;ZURUECKSCHREIBEN
1980 ;
1990          LDA DLSDR         ;DATEN HOLEN
2000 OPTION   STA (BUFFER),Y    ;UND IN DEN PUFFER SCHREIBEN
2010          INY
2020          BNE TOPRD
2030          DEX
2040          BEQ DONERD        ;FERTIG?
2050 ;
2060          INC BUFFER+1      ;NAECHSTEN PUFFER
2070          JMP READIT
2080 ;
2090 DONERD   CLI
2100          CLC               ;RUECKMELDUNG 'OK': C=0
2110          LDA #15
2120          JSR CLOSE         ;KOMMANDOKANAL SCHLIESSEN
2130          RTS
2140 ;
2150 ERROR    CLI
2160          SEC               ;RUECKMELDUNG 'FEHLER': C=1
2170          LDA #15
2180          JSR CLOSE         ;KOMMANDOKANAL SCHLIESSEN
2190          RTS
2200 ;
2210          .END

READY.
```

```
READY.

1000 ;DER BURSTBEFEHL 'WRITE'
1010 ;
1020 ;BEISPIELPROGRAMM ZUM SCHREIBEN VON N DATENBLOECKEN
1030 ;
1040 N       = 3              ;ANZAHL DER ZU LESENDEN SEKTOREN
1050 ;
1060 LOCAT   = $2000          ;BEGINN
1070 ;
1080 SETLFS  = $FFBA          ;DATEIPARAMETER FESTLEGEN
1090 SETNAM  = $FFBD          ;DATEINAMEN DEFINIEREN
1100 OPEN    = $FFC0          ;DATEI OEFFNEN
1110 CLOSE   = $FFC3          ;DATEI SCHLIESSEN
1120 CHKOUT  = $FFC9          ;AUSGABEKANAL DEFINIEREN
1130 CLRCHN  = $FFCC          ;ALLE KANAELE FREIGEBEN, STANDARD EINSTELLEN
1140 BSOUT   = $FFD2          ;EIN BYTE AUSGEBEN
1150 ;
1160 MMUREG  = $D505          ;MMU-REGISTER
1170 DLTIML  = $DC04          ;TIMER A, LOW
1180 DLTIMH  = $DC05          ;TIMER A, HIGH
1190 DLSDR   = $DC0C          ;DATENREGISTER FUER SERIELLE UEBERTRAGUNG
1200 DLICR   = $DC0D          ;INTERRUPT-KONTROLLREGISTER
1210 DLCRA   = $DC0E          ;KONTROLLREGISTER A
1220 ;
1230 D2PRA   = $DD00          ;DATENREGISTER VON CIA 2
1240 ;
1250 SERIAL  = $0A1C          ;FLAG FUER SCHNELLE SERIELLE UEBERTRAGUNG
1260 ;
1270 CLKIN   = %01000000
1280 CKINL   = %10111111
1290 CLKOUT  = %00010000
1300 ;
1310 SCSIZE  = $01            ;SEKTORGROESSE (256 BYTES/SEKTOR)
1320 ;
1330 STAT    = $FA            ;ZWISCHENSPEICHER FUER BURSTSTATUS
1340 BUFFER  = $FB            ;PUFFER ZUM AUSLESEN ($FB, $FC)
1350 OLDCLK  = $FD            ;ZWISCHENSPEICHER FUER TAKTZUSTAND
1360 ;
1370 BUFFAD  = $0400          ;BILDSCHIRMSPEICHER SOLL ALS PUFFER
1380 ;
1390 *       = LOCAT
1400 CMDLG   = 7
1410 CMDBUF  .BYTE 'U0',$02,1,1,N,2
1420 ;
1430         LDA #<BUFFAD     ;PUFFER
1440         STA BUFFER       ;BEREITSTELLEN
1450         LDA #>BUFFAD     ;
1460         STA BUFFER+1     ;
1470 ;
1480         LDA #15          ;LOGISCHE DATEINUMMER
1490         LDX #8           ;GERAETEADRESSE
1500         LDY #15          ;SEKUNDAERADRESSE
1510         JSR SETLFS       ;DATEIPARAMETER DEFINIEREN
1520 ;
1530         LDA #0           ;KEINE DATEINAME
1540         JSR SETNAM       ;DATEINAMEN ("") DEFINIEREN
1550 ;
1560         JSR OPEN         ;DATEI OEFFNEN
1570 ;
1580 WRITE   LDA #0
1590         STA STAT         ;ALTEN STATUS LOESCHEN
1600         LDA SERIAL
1610         AND #%10111111   ;FLAG FUER SCHNELLE SERIELLE UEBERTRAGUNG LOESCHEN
```

```
1620         STA SERIAL
1630 ;
1640         LDX #15
1650         JSR CHKOUT       ;AUSGABEKANAL OEFFNEN
1660 ;
1670         LDX #0
1680         LDY #CMDLG       ;BEFEHLSLAENGE
1690 SENDIT  LDA CMDBUF,X     ;BEFEHL HOLEN
1700         JSR BSOUT        ;BEFEHL SENDEN
1710         INX
1720         DEY              ;BYTEWEISE UEBERTRAGEN
1730         BNE SENDIT
1740 ;
1750         JSR CLRCHN       ;KANAELE FREIGEBEN UND STANDARD EINSTELLEN
1760 ;                        ;
1770         BIT SERIAL       ;UEBERTRAGUNGSGESCHWINDIGKEIT ERMITTELN
1780         BVC ERROR        ;NORMALE UEBERTRAGUNGSRATE, KEIN BURST-WRITE
1790 ;
1800         SEI
1810         LDA #CLKIN       ;HIGH-STATUS
1820         STA OLDCLK       ;FUER TAKT
1830 ;
1840         LDY #0
1850         LDX CMDBUF+5     ;ANZAHL DER SEKTOREN HOLEN
1860 WRITIT  JSR SPOUT        ;SERIELLE AUSGABE AUF PORT
1870 ;
1880 TOPWR   LDA D2PRA        ;TAKT ABFRAGEN
1890         CMP D2PRA        ;SICHERSTELLEN, DASS KEIN WECHSEL
1900         BNE TOPWR
1910 ;
1920         EOR OLDCLK       ;ALTER TAKTSTATUS
1930         AND #CLKIN       ;PEGELSTATUSBIT HERAUSMASKIEREN
1940         BEQ TOPWR        ;WENN TAKTPEGEL GEAENDERT, WIEDERHOLEN
1950 ;
1960         LDA OLDCLK       ;TAKTPEGEL-
1970         EOR #CLKIN       ;STATUS
1980         STA OLDCLK       ;AENDERN
1990 ;
2000         LDA (BUFFER),Y   ;JEWEILS EIN BYTE AUS PUFFER HOLEN
2010         STA DLSDR        ;UND SENDEN
2020 ;
2030 OPTION  LDA #8
2040 WAIT1   BIT DLICR        ;AUF UEBERTRAGUNG WARTEN
2050         BEQ WAIT1
2060 ;
2070         INY
2080         BNE TOPWR        ;WIEDERHOLEN, BIS 1 SEKTOR UEBERTRAGEN
2090 ;
2100         JSR SPINP        ;LESEN VOM SERIELLEN PORT
2110         BIT DLICR        ;RUECKMELDUNG LOESEN
2120         JSR CLKLO        ;TAKTPEGELSTATUS AUF LOW SETZEN, D. H. FERTIG
2130 ;
2140         LDA #$08
2150 WAIT2   BIT DLICR        ;AUF BURSTSTATUS-
2160         BEQ WAIT2        ;BYTE WARTEN
2170 ;
2180         LDA DLSDR        ;BURSTSTATUSBYTE LESEN
2190         STA STAT         ;UDN ABSPEICHERN
2200         JSR CLKHI        ;TAKTPEGELSTATUS AUF HIGH SETZEN
2210 ;
2220         LDA STAT         ;BURSTSTATUSBYTE LESEN
2230         AND #$0F         ;DISK-CONTROLLER-STATUS HERAUSMASKIEREN
2240         CMP #$02         ;FEHLERKONTROLLE: SEKTOR GEFUNDEN?
2250         BCS ERROR        ;WENN JA, ABBRUCH
2260 ;
2270         DEX
```

```
2280          BEQ DONEWR        ;FERTIG?
2290 ;
2300          INC BUFFER+1      ;NAECHSTEN PUFFER AUSLESEN
2310          JMP WRITIT        ;NAECHSTEN SEKTOR LESEN
2320 ;
2330 DONEWR CLI
2340          CLC
2350          LDA #15
2360          JSR CLOSE         ;KOMMANDOKANAL SCHLIESSEN
2370          RTS
2380 ;
2390 ERROR    CLI
2400          LDA #15
2410          JSR CLOSE         ;KOMMANDOKANAL SCHLIESSEN
2420          SEC
2430          RTS
2440 ;
2450 ;
2460 ;UNTERPROGRAMME:
2470 SPOUT    LDA MMUREG        ;DATENUEBER-
2480          ORA #$08          ;TRAGUNGSRICHTUNG AUF
2490          STA MMUREG        ;AUSGABE
2500          LDA #$7F
2510          STA DLICR         ;KEIN IRQ (INTERRUPT)
2520          LDA #0
2530          STA DLTIMH        ;
2540          LDA #$03
2550          STA DLTIML        ;TIMER SETZEN
2560          LDA DLCRA
2570          AND #$80          ;TOD
2580          ORA #$55
2590          STA DLCRA         ;CRA AUF AUSGABE SETZEN
2600          BIT DLICR         ;RUECKMELDUNG LOESCHEN
2610          RTS
2620 ;
2630 SPINP    LDA DLCRA         ;EINGABE, 6526
2640          AND #$80
2650          ORA #$08
2660          STA DLCRA
2670          LDA MMUREG
2680          AND #$F7
2690          STA MMUREG        ;RICHTUNG DER SERIELLEN UEBERTRAGUNG AUF EINGABE
2700          RTS
2710 ;
2720 CLKLO    LDA D2PRA         ;TAKTPEGELSTATUS AUF LOW SETZEN
2730          ORA #CLKOUT
2740          STA D2PRA
2750          RTS
2760 ;
2770 CLKHI    LDA D2PRA         ;TAKTPEGELSTATUS AUF HIGH SETZEN
2780          AND #$FF-CLKOUT
2790          STA D2PRA
2800          RTS
2810 ;
2820          .END

READY.
```

KAPITEL 9 ANHANG

9.1 ÄNDERUNG DER GERÄTEADRESSE

Hardware-Methode

Mit zwei DIP-Switches auf der Rückseite der 1571 kann die Geräteadresse der 1571 geändert werden. Mit einem Schraubenzieher, Bleistift oder einem anderen kleinen Werkzeug können die Schalter entsprechend eingestellt werden. In der folgenden Tabelle werden die für jede Geräteadresse erforderlichen Einstellungen angegeben.

Links („1")	Rechts („2")	Geräteadresse
Oben (ON)	Oben (ON)	8
Unten (OFF)	Oben (ON)	9
Oben (ON)	Unten (OFF)	10
Unten (OFF)	Unten (OFF)	11

Bei der 1570 sind ebenfalls zwei DIP-Switches zur Änderung der Geräteadresse vorhanden; diese sind jedoch nicht frei zugänglich. Zur Einstellung der Geräteadresse ist der Deckel der Diskettenstation durch Lösen der vier Schrauben am Gehäuseboden zu entfernen. Die beiden DIP-Switches befinden sich – von vorn gesehen – an der rechten Seite der Leiterplatte („SW1").

Hinweis: Um Schwierigkeiten wegen Garantieverlusts zu vermeiden, sollte das Öffnen der 1570 nur durch einen Commodore-Fachhändler vorgenommen werden.

Softwaremäßiges Umstellen der Geräteadresse

Wird das Diskettenlaufwerk eingeschaltet, so wird die hardwaremäßig eingestellte Geräteadresse eingelesen und entsprechend aufbereitet in die Speicherpositionen 119 und 120 geschrieben. Diese Geräteadresse kann danach jederzeit durch eine neue Adresse per Programm oder Direktmodus überschrieben werden und bleibt solange gültig, bis sie wieder geändert oder die 1570/71 zurückgesetzt (mit RESET) wird.

FORMATE FÜR DIE VORÜBERGEHENDE ÄNDERUNG DER GERÄTEADRESSE:

```
OPEN 15,8,15
PRINT# 15,"U0>" + CHR$(n)    mit n = 8 . . . 30
CLOSE 15
```

oder

```
PRINT# 15,"M-W"CHR$(119)CHR$(0)CHR$(2)CHR$
  (gerät + 32)CHR$(gerät + 64)
```

BEISPIELE:

```
5 INPUT "ALTE GERAETENUMMER";OD
10 INPUT "NEUE GERAETENUMMER";DV
20 IF DV<8 OR DV>30 THEN 10
30 OPEN 15,OD,15,"U0>"+CHR$(DV):CLOSE 15
```

oder

```
10 INPUT "NEUE GERAETENUMMER";DV
20 IF DV<8 OR DV>11 THEN 10
30 OPEN 15,8,15
40 PRINT#15,"M-W";CHR$(119);CHR$(0);CHR$
(2);CHR$(DV+32);CHR$(DV+64)
50 CLOSE 15
```

HINWEIS: Werden zwei Diskettenlaufwerke verwendet und soll die Geräteadresse eines Diskettenlaufwerks vorübergehend geändert werden, so müssen Sie das obige Programm ausführen, während das Diskettenlaufwerk, dessen Geräteadresse nicht geändert werden soll, ausgeschaltet ist. Nachdem das Programm ausgeführt worden ist, kann dieses Laufwerk wieder eingeschaltet werden. Sollen mehr als zwei Laufwerke gleichzeitig angeschlossen werden, so empfiehlt es sich, die Geräteadresse hardwaremäßig zu ändern.

9.2 DOS-FEHLERMELDUNGEN UND MÖGLICHE URSACHEN

HINWEIS: Viele kommerzielle Programmdisketten werden absichtlich mit einem oder mehreren der folgenden Fehler hergestellt, damit Programme nicht unbefugt dupliziert werden können. Kommt es zu einem Diskettenfehler, während eine Sicherheitskopie einer kommerziellen Programmdiskete angefertigt wird, so schlagen Sie in dem Handbuch für das entsprechende Programm nach. Wird in dem Copyright-Vermerk angegeben, daß Benutzer das Programm nicht für ihre eigenen Zwecke kopieren dürfen, so können Sie die Diskette nicht duplizieren. In derartigen Fällen kann in der Regel von dem Händler oder direkt von dem Unternehmen eine zusätzliche Sicherheitskopie erworben werden.

Die DOS-Fehlermeldungen haben folgendes Format:

nr,meldung,spur,sektor

Hierbei ist nr die Nummer, meldung der Text der Fehlermeldung und spur,sektor

Spur und Sektor der Stelle, an der der Fehler aufgetreten ist. In der Regel kommt man mit dem einfachen Text der Fehlermeldung aus; hier folgt nun eine ausführlichere Beschreibung der einzelnen Fehler.

00: OK (kein Fehler)
Diese Meldung wird im allgemeinen angezeigt, wenn der Fehlerkanal überprüft wird. Sie bedeutet, daß bei der letzten Diskettenoperation kein Fehler aufgetreten ist.

01: FILES SCRATCHED (kein Fehler)
(Dateien gelöscht)
Diese Meldung wird angezeigt, wenn der Fehlerkanal nach Benutzung des SCRATCH-Befehls überprüft wird. Mit der Spurnummer wird hierbei angegeben, wie viele Dateien gelöscht wurden.

HINWEIS: Werden Fehlermeldungen mit Nummern angezeigt, die kleiner sind als 20, so können diese ignoriert werden. Echte Fehler weisen Fehlernummern von 20 oder höher auf.

20: READ ERROR (block header not found)
(Lesefehler – Blockkennsatz nicht gefunden)
Die Diskettensteuereinheit kann den Kennsatz des angeforderten Datenblocks nicht finden. Diese Fehlermeldung wird durch einen unzulässigen Block oder einen zerstörten Kennsatz verursacht. Dieser Fehler kann im allgemeinen nicht behoben werden.

21: READ ERROR (no sync character)
(Lesefehler – kein Synchronisationszeichen)
Die Diskettensteuereinheit kann kein Synchronisierungszeichen in der gewünschten Spur finden. Dieser Fehler wird durch eine falsche Ausrichtung oder eine Diskette verursacht, die nicht vorhanden, nicht formatiert oder falsch eingelegt ist. Dieser Fehlermeldung kann auch ein Hardwarefehler zugrunde liegen. Ist diese Fehlermeldung auf eine der o. g. Ursachen zurückzuführen, so kann er i. a. nicht behoben werden.

22: READ ERROR (data block not present)
(Lesefehler – Datenblock nicht verfügbar)
Die Diskettensteuereinheit wurde aufgefordert, einen nicht richtig geschriebenen Datenblock zu lesen oder zu überprüfen. Diese Fehlermeldung wird in Verbindung mit den BLOCK-Befehlen angezeigt und gibt eine unzulässige Spur- und/oder Sektor-Anforderung an.

23: READ ERROR (checksum error in data block)
(Lesefehler – Prüfsummenfehler im Datenblock)
Hier handelt es sich um einen Fehler in den Daten. Der Sektor wurde in den Diskettenspeicher eingelesen, die Prüfsumme ist jedoch falsch. Dieser Fehlermeldung könnten Erdungsprobleme zugrunde liegen. Dieser relativ geringfügige Fehler kann häufig behoben werden, indem der Sektor einfach mit den Befehlen für den Direktzugriff gelesen und danach erneut geschrieben wird.

24: READ ERROR (byte decoding error)
(Lesefehler – Fehler beim Decodieren der Bytes)
Die Daten oder der Kennsatz wurden in den Diskettenspeicher eingelesen, durch ein ungültiges Bitmuster in einem Datenbyte wurde jedoch ein Hardwarefehler erkannt. Diese Fehlermeldung kann auf Erdungsprobleme hinweisen.

25: WRITE ERROR (write-verify error)
(Schreibfehler – Schreib-/Prüffehler)
Die Steuereinheit hat festgestellt, daß die auf die Diskette geschriebenen Daten und die entsprechenden Daten im Diskettenspeicher nicht übereinstimmen. Diese Fehlermeldung kann auf eine fehlerhafte Diskette hinweisen. Ist dies die Ursache, so kann die Operation mit einer anderen Diskette wiederholt werden. Benutzen Sie nur hochwertige Disketten von bekannten Markenherstellern.

26: WRITE PROTECT ON
(Schreibschutz eingeschaltet)
Die Steuereinheit wollte einen Datenblock schreiben, während die Schreibschutzkerbe abgedeckt war. Dieser Fehler wird im allgemeinen durch einen Schreibvorgang auf einer Diskette verursacht, während die Schreibschutzkerbe auf der Diskette mit einem Streifen abgedeckt ist, mit dem verhindert werden soll, daß der Inhalt der Diskette geändert wird.

27: READ ERROR (checksum error in header)
(Lesefehler – Prüfsummenfehler Kennsatz)
Die Steuereinheit hat einen Fehler in den Kennsatzbytes des angeforderten Datenblocks entdeckt. Der Block wurde nicht in den Diskettenspeicher gelesen. Diese Fehlermeldung kann auf Erdungsprobleme hinweisen. Der Fehler kann im allgemeinen nicht behoben werden.

28: WRITE ERROR (long data block)
(Schreibfehler – Langer Datenblock)
Die Steuereinheit versucht, das Synchronisierungszeichen des nächsten Kennsatzes zu finden, nachdem ein Datenblock geschrieben wurde. Wird das Synchronisierungszeichen nicht rechtzeitig gelesen, so wird diese Fehlermeldung generiert. Sie wird durch ein falsches Diskettenformat (die Daten erstrecken sich in den nächsten Block) oder durch einen Hardwarefehler verursacht.

29: DISK ID MISMATCH
(Nichtübereinstimmung von Disketten-IDs)
Die Diskettensteuereinheit sollte auf eine Diskette zugreifen, die nicht initialisiert worden ist. Diese Fehlermeldung kann auch ausgegeben werden, wenn eine Diskette über einen falschen Kennsatz verfügt.

30: SYNTAX ERROR (general syntax)
(Syntaxfehler – allgemeine Syntax)
Das DOS kann den über den Kommandokanal gesendeten Befehl nicht interpretieren. Häufig wird diese Fehlermeldung durch eine unzulässige Anzahl von Dateinamen oder unzulässiger Joker verursacht. Prüfen Sie die Eingabe und wiederholen Sie die Operation.

31: SYNTAX ERROR (invalid command)
(Syntaxfehler – ungültiger Befehl)
Das DOS erkennt den Befehl nicht. Er muß mit dem ersten gesendeten Zeichen beginnen. Prüfen Sie die Eingabe und wiederholen Sie die Operation.

32: SYNTAX ERROR (long line)
(Syntaxfehler – lange Zeile)
Der gesendete Befehl umfaßt mehr als 58 Zeichen. Benutzen Sie abgekürzte Diskettenbefehle.

33: SYNTAX ERROR (invalid file name)
(Syntaxfehler – ungültiger Dateiname)
In dem SAVE-Befehl oder beim Öffnen von Dateien für das Schreiben neuer Daten können keine Joker benutzt werden. Der Dateiname muß voll ausgeschrieben werden.

34: SYNTAX ERROR (no file given)
(Syntaxfehler – keine Datei angegeben)

Der Dateiname wurde bei einem Befehl weggelassen oder das DOS erkennt ihn nicht als solchen. Häufig wurde ein Doppelpunkt (:) weggelassen. Wiederholen Sie die Operation.

39: SYNTAX ERROR (invalid command)
(Syntaxfehler – ungültiger Befehl)
Das DOS erkennt einen über den Kommandokanal gesendeten Befehl nicht (Sekundäradresse 15). Prüfen Sie die Eingabe und wiederholen Sie die Operation.

50: RECORD NOT PRESENT
(Datensatz nicht vorhanden)
Der angeforderte Datensatz wurde noch nicht erstellt. Dies ist bei einer neuen relativen Datei oder bei einer Datei, die absichtlich erweitert wird, kein Fehler. Diese Fehlermeldung wird angezeigt, wenn über den letzten bestehenden Eintrag hinaus gelesen oder mit dem RECORD-Befehl auf einen nicht vorhandenen Datensatz Bezug genommen wird.

51: OVERFLOW IN RECORD
(Überlauf im Datensatz)
Die in den aktuellen Eintrag zu schreibenden Daten überschreiten die Länge des Datensatzes. Die überzähligen Zeichen wurden abgeschnitten. Prüfen Sie, ob sämtliche Sonderzeichen (wie beispielsweise Zeilenschaltungszeichen) enthalten sind, indem Sie die Größe des Datensatzes nachrechnen.

52: FILE TOO LARGE
(Datei zu groß)
Auf der Diskette ist nicht mehr genügend Platz vorhanden, um den angeforderten relativen Datensatz unterzubringen. Um diesen Fehler zu vermeiden, geben Sie die Größe der Datei beim Erstellen an. Ist die Datei zu groß für die Diskette, so teilen Sie sie entweder in zwei Dateien auf zwei Disketten auf oder benutzen Abkürzungen in den Daten, um kürzere Einträge zu ermöglichen.

60: WRITE FILE OPEN
(Schreibdatei geöffnet)
Eine Schreibdatei, die nicht geschlossen wurde, wird zum Lesen erneut geöffnet. Diese Datei muß sofort gesichert werden, wie unter BASIC-Hinweis 2 auf Seite 25 beschrieben. Ansonsten wird sie zu einer nicht richtig geschlossenen Datei und geht wahrscheinlich verloren.

61: FILE NOT OPEN
(Datei nicht geöffnet)
Es wird versucht, auf eine Datei, die nicht vom DOS geöffnet wurde, zuzugreifen. In einigen derartigen Fällen wird keine Fehlermeldung generiert. Statt dessen wird die Anforderung einfach ignoriert.

62: FILE NOT FOUND
(Datei nicht gefunden)
Die angeforderte Datei ist auf der Diskette im angegebenen Laufwerk nicht vorhanden. Prüfen Sie die Eingabe und wiederholen Sie die Operation. Initialisieren Sie ggf. die Diskette.

63: FILE EXISTS
(Datei vorhanden)
Eine Datei, die denselben Namen wie den für eine neue Datei angeforderten Namen aufweist, steht schon auf der Diskette. Die mehrfache Verwendung gleicher Dateinamen ist nicht zulässig. Wählen Sie einen anderen Namen.

64: FILE TYPE MISMATCH
(Nichtübereinstimmender Dateityp)
Die angeforderte Art des Dateizugriffs kann mit Dateien des angegebenen Typs nicht vorgenommen werden. Lesen Sie das Kapitel über diesen Dateityp noch einmal durch.

65: NO BLOCK
(Kein Block)
Diese Fehlermeldung wird in Verbindung mit dem Befehl B-A ausgegeben. Der Sektor, der als belegt gekennzeichnet werden sollte, ist bereits belegt. Bei den zurückgemeldeten Spur- und Sektornummern handelt es sich um die nächsthöhere verfügbare Spur und den nächsthöheren verfügbaren Sektor. Wird die Spurnummer 0 zurückgegeben, so sind alle weiteren Sektoren ebenfalls belegt. Ist die Diskette noch nicht voll, so wird ein erneuter Versuch mit einer niedrigeren Spur- und Sektornummer vorgenommen.

66: ILLEGAL TRACK OR SECTOR
(Spur oder Sektor unzulässig)
DOS hat versucht, auf eine nicht verfügbare Spur oder einen nicht vorhandenen Sektor zuzugreifen. Diese Fehlermeldung beruht meistens auf einem fehlerhaften Verkettungspointer in einem Datenblock, kann jedoch auch durch falsche Angaben bei den Direktzugriffsbefehlen hervorgerufen werden.

67: ILLEGAL SYSTEM T OR S
(Systemspur oder -sektor unzulässig)
Diese besondere Fehlermeldung gibt eine unzulässige Systemspur bzw. -block an.

70: NO CHANNEL
(Kein Kanal verfügbar)
Der angeforderte Kanal ist nicht verfügbar oder sämtliche Kanäle werden gerade benutzt. Maximal drei sequentielle Dateien oder eine relative Datei plus einer sequentiellen Datei können gleichzeitig geöffnet sein, plus dem Befehlskanal. Die Laufwerksnummer in einem OPEN-Befehl für eine sequentielle Datei sollte nicht weggelassen werden, oder es können nur zwei sequentielle Dateien benutzt werden. Schließen Sie sämtliche Dateien, sobald sie nicht mehr benötigt werden.

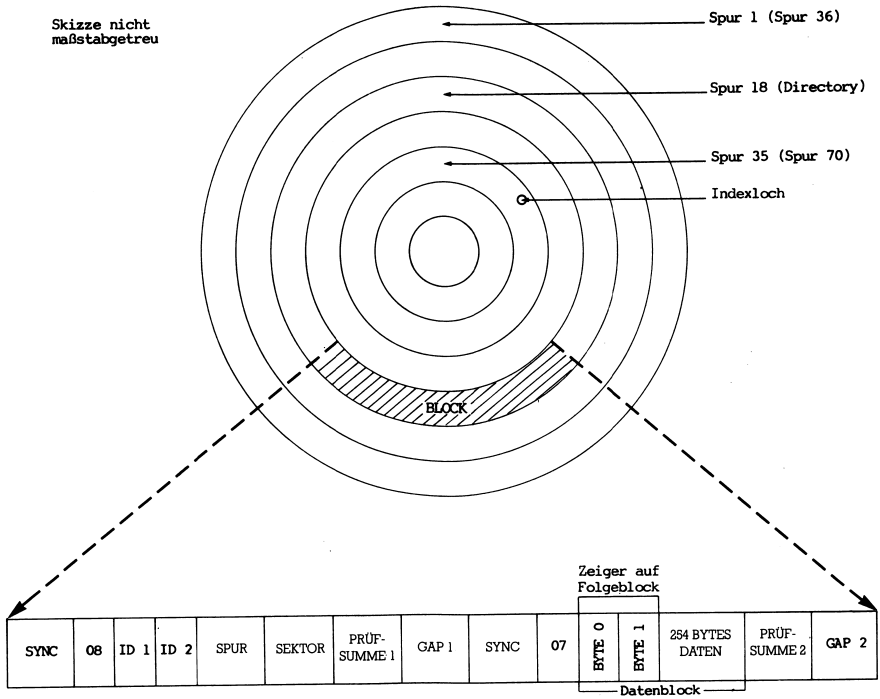
71: DIRECTORY ERROR
(Fehlerhaftes Inhaltsverzeichnis)
Die BAM (Block Availability Map) auf der Diskette stimmt nicht mit der Kopie im Speicher der 1571 überein. Zur Behebung der Fehlermeldung ist die Diskette zu initialisieren.

72: DISK FULL
(Diskette voll)
Die Diskette oder ihr Inhaltsverzeichnis ist voll. Die Fehlermeldung DISK FULL wird gesendet, wenn nur noch zwei Blöcke verfügbar sind, so daß die aktuelle Datei abgeschlossen werden kann. Wird diese Fehlermeldung empfangen und enthält das Inhaltsverzeichnis noch freie Blöcke, so haben Sie unter Umständen zu viele einzelne Dateien im Inhaltsverzeichnis und müssen einige Dateien kombinieren, nicht mehr benötigte Dateien löschen oder einige Dateien auf eine andere Diskette kopieren.

73: CBM DOS V3.0 1570 oder CBM DOS V3.0 1571
(Nichtübereinstimmung des DOS (CBM DOS V3.0 1570 bzw. 1571)
Wird der Fehlerstatus beim ersten Einschalten des Laufwerks überprüft, bevor ein Inhaltsverzeichnis erstellt oder ein anderer Befehl ausgegeben worden ist, so wird diese Meldung angezeigt. In diesem Fall handelt es sich nicht um eine Fehlermeldung, sondern um eine einfache Möglichkeit, festzustellen, welche DOS-Version gerade benutzt wird. Wird die Meldung später angezeigt, so wurde versucht, auf eine Diskette mit einem nichtkompatiblen Format zu schreiben, wie beispielsweise der alten DOS-1-Diskette vom Commodore-2040-Diskettenlaufwerk. Kopieren Sie die gewünschte(n) Datei(n) mit einem der Kopierprogramme auf der Test-/Demo-Diskette auf eine 1571-Diskette.

74: DRIVE NOT READY
(Laufwerk nicht bereit)
Es wurde versucht, auf das 1570/71-Einzellaufwerk zuzugreifen, ohne daß eine formatierte Diskette eingelegt war. Leere Disketten können erst benutzt werden, nachdem sie formatiert wurden.

9.3 DISKETTENFORMATE



Die Spuren 36 bis 70 beziehen sich auf zweiseitige Disketten (nur bei 1571)

Abbildung 4 GCR-formatierte Diskette

SYNC	40 Einer-Bits (nur Einsen entsprechen der höchsten Schreibfrequenz
08	Marke zur Kennzeichnung des Blockkennsatzes
PRÜFSUMME 1	Prüfsumme von ID1, ID2, SEKTOR, SPUR (ID1 und ID2 bilden zusammen die ID der Diskette)
SEKTOR	Die Nummer dieses Sektors
SPUR	Spur, in der dieser Sektor steht

ID1,ID2	2 Bytes (16 Bits), die die Disketten-ID angeben (die ID wird vom Benutzer angegeben, wenn die Diskette formatiert wird)
ZWISCHENRAUM 1 (GAP 1)	72 Bits mit GCR0 oder 01010101 (8 Bit umfassendes Byte × 9)
SYNC	40 Einer-Bits
07	Marke zur Kennzeichnung des Datenfeldes auf der Diskette
BYTE0,BYTE1	Spur und Sektor des nächsten zugehörigen Datenblockes
DATEN	254 Datenbytes (2540 Bits)
PRÜFSUMME 2	Prüfsumme von BYTE0, BYTE1 und DATEN
ZWISCHENRAUM 2 (GAP 2)	Variabler Zwischenraum mit GCR0. Dieser Zwischenraum ist variabel, je nach Geschwindigkeit der Diskette beim Formatieren. Zwischenraum 2 ist für sämtliche Sektoren konstant, mit Ausnahme des letzten Zwischenraums (der Zwischenraum zwischen dem ersten und letzten Sektor).

AUFTEILUNG DER BLÖCKE NACH SPUREN BEI DER 1570 UND 1571

Spurnummer	Sektorbereich	Gesamtzahl der Sektoren
1 bis 17	0 bis 20	21
18 bis 24	0 bis 18	19
25 bis 30	0 bis 17	18
31 bis 35	0 bis 16	17
36 bis 52	0 bis 20	21
53 bis 59	0 bis 18	19
60 bis 65	0 bis 17	18
66 bis 70	0 bis 16	17

Die Spuren 36 bis 70 sind nur bei der 1571 verfügbar.

BAM-FROMAT/HEADER DES INHALTSVERZEICHNISSES BEI DER 1570 UND BEI DER 1571 IM 1541-MODUS

BYTE	INHALT	DEFINITION
0	18	Spur des nächsten Verzeichnisblockes (immer 18)
1	1	Sektor des nächsten Verzeichnisblockes (immer 1)
2	65	ASCII-Zeichen A, gibt 1541-/1551-/1571-/4040-Format an

3	0	Flag für doppelseitige Disketten (wird im 1541-Modus ignoriert)
4		Anzahl der nicht belegten Sektoren in Spur 1
5		Spur 1, Sektor 0-7, Belegungstabelle
6		Spur 1, Sektor 8-16, Belegungstabelle
7		Spur 1, Sektor 17-23, Belegungstabelle
8		Anzahl der nicht belegten Sektoren in Spur 2
9		Spur 2, Sektor 0-7, Belegungstabelle
10		Spur 2, Sektor 8-16, Belegungstabelle
11		Spur 2, Sektor 17-23, Belegungstabelle
...		
140		Anzahl der nicht belegten Sektoren in Spur 35
141		Spur 35, Sektor 0-7, Belegungstabelle
142		Spur 35, Sektor 8-16, Belegungstabelle
143		Spur 35, Sektor 17-23, Belegungstabelle
144-159		Diskettenname, aufgefüllt mit geshifteten Leerzeichen [CHR$(160)]
160-161	160	Geshiftete Leerzeichen [CHR$(160)]
162-163		Disketten-ID
164	160	Geshiftete Leerzeichen [CHR$(160)]
165-166		ASCII-Darstellung von 2A, wobei es sich um die DOS-Version (2) bzw. den Formattyp (A für 1540/1541/1551/1570/1571/4040/2030) handelt.
167-170		Geshiftete Leerzeichen [CHR$(160)]
171-225		Nullen [CHR$(0)], nicht benutzt.

BAM-FORMAT/HEADES DES INHALTSVERZEICHNISSES BEI DER 1571 IM 1571-MODUS)

BYTE	INHALT	DEFINITION
0	18	Spur des nächsten Verzeichnisblockes (immer 18)
1	1	Sektor des nächsten Verzeichnisblockes (immer 1)
2	65	ASCII-Zeichen A, mit dem das 1541-/1551-/1570-/1571-/4040-Format gekennzeichnet wird
3		Flag für doppelseitige Disketten: $80 = doppelseitig, $00 = einseitig
4		Anzahl der verfügbaren Sektoren in Spur 1
5		Spur 1, Sektor 0-7, Belegungstabelle
6		Spur 1, Sektor 8-16, Belegungstabelle
7		Spur 1, Sektor 17-23, Belegungstabelle

8		Anzahl der verfügbaren Sektoren in Spur 2
9		Spur 2, Sektor 0-7, Belegungstabelle
10		Spur 2, Sektor 8-16, Belegungstabelle
11		Spur 2, Sektor 17-23, Belegungstabelle
...		
140		Anzahl der verfügbaren Sektoren in Spur 35
141		Spur 35, Sektor 0-7, Belegungstabelle
142		Spur 35, Sektor 8-16, Belegungstabelle
143		Spur 35, Sektor 17-23, Belegungstabelle
144-159		Diskettenname, mit geshifteten Leerzeichen aufgefüllt [CHR$(160)]
160-161	160	Geshiftete Leerzeichen [CHR$(160)]
162-163		Disketten-ID
	160	Geshiftete Leerzeichen [CHR$(160)]
165-166		ASCII-Darstellung von 2A, wobei es sich um die DOS-Version (2) bzw. den Formattyp (A für 1540/1541/1551/1570/1571/4040/2030) handelt.
167-179		Geshiftete Leerzeichen [CHR$(160)]
171-220		Nullen [CHR$(0)[, nicht benutzt.
221-237		Anzahl der verfügbaren Sektoren von Spur 36 bis 52 (ein Byte pro Spur)
238	0	Anzahl der verfügbaren Sektoren in Spur 53 (immer 0, in der Regel sind jedoch alle Spuren frei, obwohl das DOS diese Spur nicht unterstützt)
239-244		Anzahl der verfügbaren Sektoren in Spur 54 bis 59 (ein Byte pro Spur)
245-250		Anzahl der verfügbaren Sektoren in Spur 60 bis 65 (ein Byte pro Spur)
251-255		Anzahl der verfügbaren Sektoren in Spur 66 bis 70 (ein Byte pro Spur)

* %1 = Sektor frei (% bedeutet binär)
%0 = Sektor belegt (jedes Bit stellt einen Block dar)

Spur 53, Sektor 0 (nur bei der 1571)

BYTE	INHALT	DEFINITION
1		Spur 36, Sektor 0-7, Belegungstabelle
2		Spur 36, Sektor 8-16, Belegungstabelle

3		Spur 36, Sektor 17-23, Belegungstabelle
...		
102		Spur 70, Sektor 0-7, Belegungstabelle
103		Spur 70, Sektor 8-16, Belegungstabelle
104		Spur 70, Sektor 17-23, Belegungstabelle
105-255	0	Nullen ($00), nicht benutzt

%1 = Block frei (% bedeutet binär)
%0 = Block belegt (jedes Bit stellt einen Block dar)

FORMAT EINER PROGRAMMDATEI

BYTE	DEFINITION
ERSTER SEKTOR DER PRG-DATEI	
0,1	Spur und Sektor des nächsten Programmblocks.
2,3	Ladeadresse des Programms.
4-255	Die nächsten 252 Bytes der Programminformation, wie sie in den Speicher des Computers übertragen werden (wobei die BASIC-Schlüsselwörter in Form von Tokens angegeben werden).
WEITERE SEKTOREN DER PRG-DATEI	
0,1	Spur und Sektor des nächsten Programmblocks.
2-255	254 Bytes mit zu ladenden Daten.
LETZTER SEKTOR DER PRG-DATEI	
0,1	Null ($00), gefolgt von der Anzahl gültiger Bytes in diesem Sektor; hierzu gehören auch die Bytes 0 und 1.
ab 2	Die letzten Bytes der Programminformation. Das Ende einer BASIC-Datei wird durch drei Null-Bytes in einer Reihe markiert.

FORMAT EINER SEQUENTIELLEN DATEI

ALLE SEKTOREN (AUSSER DEM LETZTEN)	
0-1	Spur und Sektor des nächsten Datenblocks.
0-255	254 Datenbytes.
LETZTER SEKTOR DER DATEI	
0,1	Null ($00), gefolgt von der Anzahl gülter Bytes in diesem Sektor einschließlich der Bytes 0 und 1.
ab 2	Letzte Datenbytes.

FORMAT EINER RELATIVEN DATEI

BYTE	DEFINITION
DATENBLOCK	
0,1	Spur und Sektor des nächsten Datenblockes.
2-255	254 Datenbytes. Leere Einträge enthalten $FF im ersten Byte, gefolgt von $00 bis zum Ende des Datensatzes (Record). Teilweise beschriebene Datensätze werden mit Nullen aufgefüllt ($00).
SIDE-SEKTOR-BLOCK	
0-1	Spur und Sektor des nächsten Side-Sektor-Blocks.
2	Nummer des Side-Sektors
3	Länge des Datensatzes
4-5	Spur und Sektor des ersten Side-Sektors (Nummer 0).
6-7	Spur und Sektor des zweiten Side-Sektors (Nummer 1).
8-9	Spur und Sektor des dritten Side-Sektors (Nummer 2).
10-11	Spur und Sektor des vierten Side-Sektors (Nummer 3).
12-13	Spur und Sektor des fünften Side-Sektors (Nummer 4).
14-15	Spur und Sektor des sechsten Side-Sektors (Nummer 5).
16-255	Zeiger (Spur und Sektor) auf 120 Datenblöcke.

AUFBAU DES INHALTSVERZEICHNISSES
Spur 18, Sektor 1 bis 19

BYTE	DEFINITION
0,1	Spur und Sektor des nächsten Blocks des Inhaltsverzeichnisses
2-31	Dateieintrag 1*
34-63	Dateieintrag 2*
66-95	Dateieintrag 3*
98-127	Dateieintrag 4*
130-159	Dateieintrag 5*
162-191	Dateieintrag 6*
194-223	Dateieintrag 7*
226-255	Dateieintrag 8*
* STRUKTUR JEDES EINZELNEN VERZEICHNISEINTRAGS	

BYTE	INHALT	DEFINITION
0	128 + Type	Dateityp wird mit OR mit $80 verknüpft, um eine richtig abgeschlossene Datei anzugeben (wird die OR-Verknüpfung stattdessen mit $C0 vorgenommen, so ist die Datei gesperrt). TYPEN: 0 = DELeted (gelöscht) 1 = SEQential (sequentiell) 2 = PRogram (Programm) 3 = USeR (sequentiell, USER-Datei) 4 = RELative (relativ)
1-2		Spur und Sektor des ersten Datenblockes.
3-18		Dateiname, mit geshifteten Leerzeichen aufgefüllt.
19-20		Nur bei relativen Dateien: Spur und Sektor des ersten Side-Sektor-Blocks.
21		Nur bei relativen Dateien: Satzlänge.
22-25		Nicht benutzt.
26-27		Spur und Sektor der Ersatzdatei während eines @SAVE oder @OPEN.
28-29		Anzahl der Blöcke in der Datei: wird als 2-Byte-INTEGER-Zahl mit dem niederwertigen Byte zuerst und dem höherwertigen Byte darauffolgend gespeichert.

9.4 TABELLE DER DISKETTENBEFEHLE

BASIC 2.0

Allgemeines Format: OPEN 15,8,15:PRINT#15,befehl:CLOSE 15

VERWALTUNGSBEFEHLE

BASIC 2.0	NEW	"N0:diskettenname,id"
	COPY	"C0:neue datei=0:alte datei"
	RENAME	"R0:neuer name=alter name"
	SCRATCH	"S0:dateiname"
	VALIDATE	"V0"
BASIC 7.0	NEW	HEADER "diskettenname",lid,D0
(BASIC 3.5)	COPY	COPY "alte datei" TO "neue datei"
	RENAME	RENAME "alter name" TO "neuer name"
	SCRATCH	SCRATCH "dateiname"
	VALIDATE	COLLECT
BASIC 2.0, 3.5, 7.0	INITIALIZE	"I0"

DATEIBEFEHLE

BASIC 2.0	LOAD	LOAD "dateiname",8
	SAVE	SAVE "dateiname",8
	VERIFY	VERIFY "dateiname",8
BASIC 7.0/3.5	LOAD	DLOAD "dateiname"
	SAVE	SAVE "dateiname"
	VERIFY	DVERIFY "dateiname" (nur BASIC 7.0)
Binärdatei (nur BASIC 7.0)	BLOAD	BLOAD "dateiname",Bbank,Pstartadresse
	BSAVE	BSAVE "dateiname",Bbank,Pstartadresse TO endadresse+1
	BOOT	BOOT "dateiname"
	OPEN	DOPEN#datei#,"dateiname" [,Leintragslänge] [,W]
	CLOSE	DCLOSE#datei
	RECORD#	RECORD#,datei,eintragsnummer [,orfset]

BASIC 2.0, 3.5, 7.0	OPEN	OPENdatei,gerät,kanal,"0:dateiname,dateityp, richtung"
	CLOSE	CLOSEdatei
	RECORD#	"P" + CHR$(kanal)+CHR$(<eintrag) + CHR$(>eintrag)+CHR$(offset)
	PRINT#	PRINT#datei,datenliste
	GET#	GET#datei,variablenliste
	INPUT#	INPUT#datei,variablenliste

BEFEHLE FÜR DEN DIREKTZUGRIFF

BLOCK-ALLOCATE	"B-A";0;spur#;sektor#
BLOCK-EXECUTE	"B-E";kanal#;0;spur#;sektor#
BLOCK-FREE	"B-F";0;spur#;sektor#
BUFFER-POINTER	"B-P";kanal#;byte
BLOCK-READ	"U1";kanal#;0;spur;sektor#
BLOCK-WRITE	"U2";kanal#;0;spur#;sektor#
MEMORY-EXECUTE	"M-E"CHR$(<adresse)CHR$(>adresse)
MEMORY-READ	"M-R"CHR$(<adresse)CHR$(>adresse)CHR$(anzahl)
MEMORY-WRITE	"M-W"CHR$(<adresse)CHR$(>adresse)CHR$(anzahl)CHR$ (datenbyte)CHR$(datenbyte)...
USER	"Uzeichen"
UTILITY LOADER	"&0:dateiname"

9.5 TECHNISCHE DATEN

GCR-Format	Einseitig	Doppelseitig (nur 1571)
	252019 Bytes	252019*2 Bytes
Kapazität (unformatiert)	174848 Bytes	349696 Bytes
Kapazität (formatiert)	168656 Bytes	337312 Bytes
Maximale Größe einer sequentiellen Datei	167132 Bytes	167132 Bytes
Maximale Größe einer relativen Datei	65535	65535
Einträge pro Datei	144	144
Dateien pro Diskette	35	70
Spuren pro Diskette	17-21	17-21
Sektoren pro Spur	683 insgesamt	1366 insgesamt
Sektoren pro Diskette	664 frei	1328 frei
	256	256
Bytes pro Sektor		

MFM-Format	
Kapazität (unformatiert):	500000 Bytes pro Seite
Kapazität (formatiert):	
Sektorgröße 128	133120 Bytes pro Seite
Sektorgröße 256	163840 Bytes pro Seite
Sektorgröße 512	184320 Bytes pro Seite
Sektorgröße 1024	204800 Bytes pro Seite
Maximale Anzahl der Spuren	40 pro Seite
Sektoren pro Spur	
Sektorgröße 128	26
Sektorgröße 256	16
Sektorgröße 512	9
Sektorgröße 1024	5

BENUTZTE CHIPS	
6502A	Mikroprozessor
6522	
6526	
23256	ROM mit 32K Bytes
6116	RAM mit 2K Bytes
64H156/64H157	Gate Array
R/W-Hybridschaltkreis	Analogschaltung (MFM, GCR)
WD 1770	Disk-Controller

ABMESSUNGEN	1570	1571
Höhe	97 mm	76 mm
Breite	200 mm	216 mm
Tiefe	374 mm	346 mm
Gewicht	3,5 kg	2,8 kg

BETRIEBSDATEN	
Spannung	220 V Wechselspannung
Frequenz	50 Hz
Leistungsaufnahme	25 W

DATENTRÄGER

Jede hochwertige 5 1/4-"-Diskette kann benutzt werden. (Die Verwendung von Commodore-Disketten wird empfohlen.)

Technische Daten

Mikroprozessor 6502
2 K RAM
32 K ROM (integriertes DOS)
Serieller Bus, kompatibel zu allen Commodore-Homecomputern.

Diskettenformate:
Commodore Standard (GCR, Single Sided, Single Density)

Speicherkapazität:	170 KB (formatiert)
Spuren pro Diskette:	35
Sektoren pro Spur:	17 bis 21, je nach Lage
Sektoren pro Diskette:	insgesamt 683, 664 frei für den Anwender
Bytes pro Sektor:	256

MFM-Formate:
Speicherkapazität:

– bei 128 Bytes/Sektor:	130 KB
– bei 256 Bytes/Sektor:	160 KB
– bei 512 Bytes/Sektor:	180 KB
– bei 1024 Bytes/Sektor:	200 KB
Spuren pro Diskette:	max. 40

Sektoren/Spur:

– bei 128 Bytes/Sektor:	26
– bei 256 Bytes/Sektor:	16
– bei 512 Bytes/Sektor:	9
– bei 1024 Bytes/Sektor:	5

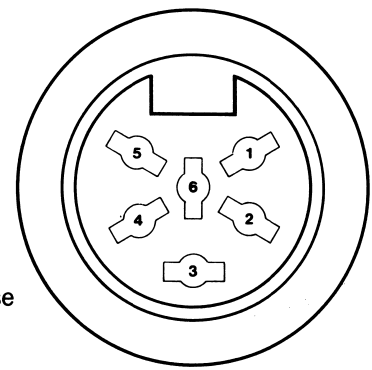
Datenträger:
Jede hochwertige 5-1/4-Zoll-Diskette kann verwendet werden. Commodore-Disketten werden empfohlen.

Maße: 97 mm x 200 mm x 374 mm (H x B x T)

Spannungsversorgung: 220 V, 50 Hz
Leistungsaufnahme: 25 W

9.6 DER SERIELLE BUS

Die serielle Schnittstelle besteht aus zwei 6poligen DIN-Buchsen bei jedem Laufwerk. Die zweite Buchse ist für die Verbindung mit anderen Laufwerken bzw. weiteren Peripheriegeräten vorgesehen. Bei dieser Schnittstelle handelt es sich um eine serielle Schnittstelle auf TTL-Ebene.
Über den seriellen Bus werden drei Arten von Operationen ausgeführt – Kontrolle, Senden, Empfangen. Der Rechner stellt die Steuereinheit dar und leitet die Protokolle auf dem seriellen Bus ein. Der Rechner fordert das Peripheriegerät zum Senden oder Empfangen auf. Sämtliche Einheiten, die an den seriellen Bus angeschlossen sind, empfangen die über den Bus gesendeten Daten. Damit der Rechner ein bestimmtes einzelnes Gerät ansprechen kann, verfügt jedes Gerät über eine Geräteadresse. Die Geräteadressen des Diskettenlaufwerks liegen zwischen 8 und 11 (normalerweise wird 8 verwendet).



Die 6polige DIN-Buchse (von außen)

Im einzelnen unterstützt der serielle Bus der 1570/71 die neue schnelle serielle Datenübertragung ebenso wie die bislang verwendete Stadarddatenübertragung. Der Hauptunterschied zwischen dem schnellen seriellen Bus und dem seriellem CBM-Standardbus liegt in der Verwendung von hardwaremäßig kontrollierten Leitungen für die Signale CLOCK und DATA. Dies wird über Shift-Register erzielt, die in dem 6526 resident vorhanden sind. Die schnelle serielle Kommunikation ist über jedes Peripheriegerät transparent, das an den seriellen Bus angeschlossen ist und nicht für die erforderliche Hard- bzw. Software verfügt, um Daten in hoher Geschwindigkeit zu sprechen.
Um mit dem seriellen CBM-Standardbus kompatibel zu bleiben, werden sämtliche Bytes, die unter ATN gesendet werden, langsam gesendet. Das Laufwerk wird im Standardmodus aktiviert. Der Rechner (C128) muß das Laufwerk in den schnellen Modus umschalten. Das Laufwerk bleibt im schnellen Modus, bis der gesendete Befehl abgearbeitet worden ist.

Die Pin-Belegung der Buchse für den seriellen Bus geht aus folgender Tabelle hervor:

Pin-Nummer	Signal	Richtung	Beschreibung
Pin 1	SRQ (Serviceanforderung)	I/O	Wird vom schnellen seriellen Bus als schnelle Taktleitung in beiden Richtungen benutzt. Wird von dem langsamen seriellen Bus nicht verwendet.
Pin 2	GND (Erde)	I	Masse
Pin 3	ATN	I/O	Der Rechner setzt dieses Signal auf den Logikpegel Low und löst damit einen Interrupt auf der Steuerplatine aus. Daraufhin wird die Geräteadresse auf der Datenleitung gesendet. Antwortet keines der angeschlossenen Peripheriegeräte daraufhin innerhalb einer bestimmten vorgegebenen Zeit, so geht der Sender (Rechner) davon aus, daß das adressierte Gerät nicht am Bus angeschlossen ist.
Pin 4	CLK (Takt)	I/O	Dieses Signal wird für die zeitliche Steuerung der Daten benutzt, die auf dem langsamen seriellen Bus gesendet werden (Softwaretakt).
Pin 5	DATA	I/O	Die Daten auf dem seriellen Bus werden softwaremäßig getaktet Bit für Bit übertragen.
Pin 6	RESET		Dieses Signal bewirkt einen RESET des Peripheriegerätes nach einem RESET des Rechners.

9.7 DIE TEST-/DEMODISKETTE

Die Test-/Demodiskette liegt den COMMODORE-Diskettenlaufwerk 1570 und 1571 bei. Sie enthält eine Beschreibung ihrer Programme (HOW TO USE), zwei Programmpakete zur vereinfachten Handhabung (DOS 5,1, DOS SHELL) und zahlreiche weitere nützliche Hilfsprogramme. Im folgenden Inhaltsverzeichnis sind diese Programme einzeln aufgeführt.

```
0 [redacted]
42    "HOW TO USE"          PRG
36    "HOW TO PART 2"       PRG
1     "CBM"                 USR
68    "DOS SHELL"           PRG
1     "C-64 WEDGE"          PRG
4     "VIC-20 WEDGE"        PRG
4     "DOS 5.1"             PRG
7     "SD.BACKUP.C16"       PRG
7     "SD.BACKUP.+4"        PRG
13    "SD.BACKUP.C64"       PRG
8     "SD.COPY.C64"         PRG
14    "UNI-COPY"            PRG
8     "PRINT.+4.UTIL"       PRG
8     "PRINT.C16.UTIL"      PRG
8     "PRINT.64.UTIL"       PRG
8     "DIR PRINT"           PRG
9     "DIR SQUEEZE"         PRG
14    "DISK-DUMP"           PRG
8     "DISPLAY BAM"         PRG
5     "CHANGE UNIT"         PRG
8     "UNSCRATCH"           PRG
8     "LOAD ADDRESS"        PRG
19    "REL.FILE.DEMO"       PRG
5     "SEQ.FILE.DEMO"       PRG
5     "SEQ LISTER"          PRG
3     "DATAMAKER.C64"       PRG
3     "DATAMAKER.TED"       PRG
5     "HEADER CHANGE"       PRG
9     "C64 AUTOSTART"       PRG
23    "C64 DISKPATCH"       PRG
9     "PRINTER TEST"        PRG
4     "FORMAT DISK"         PRG
8     "AUTOBOOT MAKER"      PRG
281 BLOCKS FREE.
```

BESCHEINIGUNG DES HERSTELLERS

Hiermit wird bestätigt, dass das Disk Drive

COMMODORE 1570 / 1571

in Übereinstimmung mit den Bestimmungen der

Amtsblattverfügung Nr. 1046/1984

funk-entstört ist.

Der Deutschen Bundespost wurde das Inverkehrbringen dieses Gerätes angezeigt und die Berechtigung zur Überprüfung der Serie auf Einhaltung der Bestimmungen eingeräumt.

COMMODORE BÜROMASCHINEN GMBH

CERTIFICATE OF THE MANUFACTURER

Herewith we certify that our device disk drive

COMMODORE 1570 / 1571

corresponds to the regulations

Amtsblattverfügung Nr. 1046/1984

is eliminated of radio interference.

The German Bundespost has been informed that this unit is on the market and has got the right to check on the mass production if the limits are kept.

COMMODORE BUSINESS MACHINES LIMITED

Commodore

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

Commodore AG
Langenhagstr. 1
CH-4147 Aesch

Commodore Büromaschinen GmbH
Kinskygasse 40–44
A-1232 Wien

P/N 325 135/01