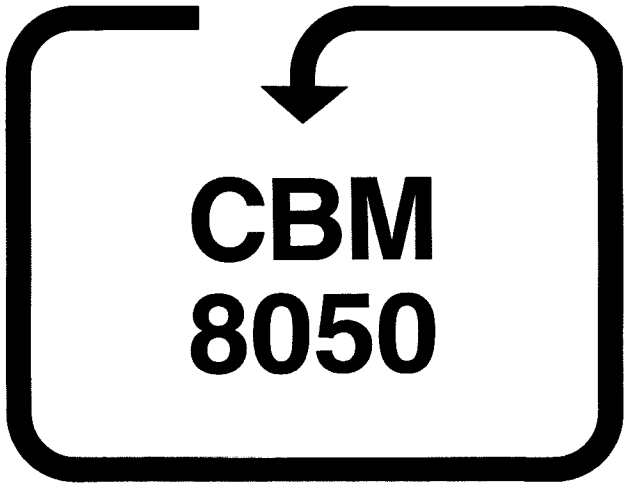


commodore - computer

Bedienungshandbuch

INHALTSÜBERSICHT

I. VORWORTE

1. Hinweise zur Typographie

Dieses Handbuch wurde vollständig mit Text-Bearbeitungs-Programmen erstellt, die auf dem Markt für den CBM 8032 erhältlich sind und auf einem Typenraddrucker der gehobenen Klasse ausgedruckt, der über den IEC-Bus ganz regulär an den CBM 8032 angeschlossen ist. Daran können Sie die mögliche Leistungsfähigkeit Ihres Computers erkennen.

Leider brachte es der deutsche Zeichensatz mit sich, daß ein Zeichen, das auf dem Bildschirm und der Tastatur Ihres Computers vorhanden ist, nicht auf dem verwendeten Typenrad enthalten ist.

Doppelkreuz ist das SHIFT-Zeichen der Ziffer 3 und wird für die Kennzeichnung der logischen Adresse verwendet. Dieses Zeichen wird als § (Paragraphenzeichen) dargestellt. Da andererseits das Paragraphenzeichen auf dem Computer nicht zu finden ist, kann es durch diese ´falsche´ Darstellung zu keinen Verwechslungen kommen.

Wir bitten Sie um Verständnis für diese kleine Unannehmlichkeit.

Beachten Sie auch folgenden Hinweis zur **Groß- und Kleinschreibung:**

Alle Kommandobestandteile, die groß geschrieben sind, müssen exakt so geschrieben werden, wie dargestellt.

Aber: Diese Großbuchstaben müssen **ohne SHIFT** eingegeben werden, ganz gleich, ob sie dann auf dem Bildschirm groß oder klein dargestellt werden!

Alle klein geschriebenen Kommandobestandteile sind Platzhalter und benötigen aktuelle Werte.

2. Vorwort

Dieses Handbuch soll kein Leitfaden zum Erlernen von allgemeinen Dateiverwaltungskonzepten sein, sondern ein Nachschlagewerk, in dem alle wissenswerten Fakten über die Programmierung Ihres COMMODORE - Floppy-Laufwerks zusammengestellt sind.

Lassen Sie sich von der Informationsfülle nicht abschrecken, Sie haben ja ein Nachschlagewerk vor sich. Die Informationen wurden sehr sorgfältig ausgewählt. Grundlage war die große Erfahrung des SOFTWAREVERBUND-MICROCOMPUTER in der Programmierung von Commodore-Floppys und die Erfahrung aus vielen Anwenderseminaren, die von den Autoren durchgeführt wurden.

Für Anfänger wollen wir noch zwei Hinweise geben:

(1) Haben Sie Mut zur Lücke. Das soll heißen, daß Sie sich nicht beim ersten Durchlesen an irgendwelchen Einzelheiten verbeißen sollen, die Ihnen unverständlich erscheinen.

Viele Details werden erst im Zusammenhang mit anderen Fakten verständlich, so daß Sie also mindestens einmal alles gelesen haben müssen, um beim nächsten Mal mehr verstehen zu können.

Einige Details und auch ganze Anweisungen werden Sie am Anfang nicht benötigen, vielleicht auch nie. Merken Sie sich bei solchen vernachlässigten Stellen aber an, daß Sie sie nicht gelesen oder nicht verstanden haben. Vielleicht schmökern Sie später nochmal alle Stellen durch, die Sie so gekennzeichnet haben und finden doch noch den einen oder anderen wertvollen Hinweis.

(2) Erkundigen Sie sich bei Ihrem Commodore-Händler nach Schulungskursen. In der Regel ist es leichter, sich nach oder während einer Einführung in Seminarform mit dem Floppy anzufreunden, als sofort ins kalte Wasser zu springen.

(3) Beachten Sie die Hinweise im nachfolgenden Kommentar. Daraus können Sie ersehen, welche Teile für wen wichtig sind.

Wir hoffen, daß dieses Handbuch Sie möglichst effektiv mit den außerordentlich vielfältigen Möglichkeiten Ihres Floppys vertraut macht und daß Sie an seiner Programmierung genausoviel Spaß und Befriedigung haben werden wie die Autoren und ihre Mitarbeiter.

Neu-Isenburg, im Januar 1981

COMMODORE GmbH

H. Schießl und J. Steiner

3. Kommentar zur Einteilung der Information

Dieses Handbuch ist in 8 Abschnitte unterteilt. Der besseren Übersicht wegen soll erklärt werden, was in diesen Abschnitten besprochen wird, für wen dies wissenswert ist und welche Voraussetzungen beim Lesen gegeben sein sollten.

zu I. Vorworte:

Inhalt

Vorworte

Für wen?

Für Sie

Voraussetzungen

Die Vorworte noch nicht gelesen haben

zu II. Auspacken, Inbetriebnahme, Gerätebeschreibung:

Inhalt

Auspacken, Inbetriebnahme und 'warum haben Sie ein Floppy gekauft'

Für wen?

Für jeden, der Floppys auspackt und anschließt und wissen will, warum.

Voraussetzungen

Karton mit Floppy und etwas Neugierde

zu III. Hinweise für den Bediener:

Inhalt

Die täglichen Handgriffe und die wichtigsten Kommandos

Für wen?

Anwender und Bediener, eventuell auch Programmierneulinge, die am Anfang ja mit Anwendern gleichzusetzen sind.

Ob Sie die Punkte 5.x benötigen, hängt von der Software ab, die Sie verwenden. Je besser die Software ist, desto mehr brauchen Sie nur Punkt 5.1 verstehen.

Voraussetzungen

Die Punkte 1. - 4. sollte jeder Anwender verstehen. Für Punkt 5. sollten Sie schon die ärgste Angst vor dem Computer abgebaut haben

zu IV. Kommunikationsmöglichkeiten und Parameter:

Inhalt

Je nachdem, ob Sie einen Rechner mit BASIC-4 (80XX oder 40XX) verwenden, oder noch mit BASIC-3 arbeiten (30XX), haben Sie verschiedene Möglichkeiten, Kommandos ans Floppy zu übertragen.

Nach einer Einführung in diese Problematik werden in den Punkten 2.3.x die Möglichkeiten beschrieben, die BASIC-3 bietet und in 2.4.x die von BASIC-4. Dabei werden auch Unterschiede bei der Parameterangabe und vor allem bei den Voreinstellungen erläutert.

Die prinzipielle Bedeutung aller Parameter und die normalerweise erlaubten Werte werden in Punkt 2.5 gemeinsam für beide Übermittlungstechniken beschrieben. Diese zusammengefasste Darstellung kommt jenen entgegen, die mit beiden Möglichkeiten arbeiten müssen.

Für wen?

Für jeden, der Floppy-Kommandos entweder im Direktmodus oder im Programm verwenden will. Interessierte Anwender können diesen Abschnitt lesen, Programmierer müssen ihn lesen.

Voraussetzungen

Grundkenntnisse in BASIC und im Programmieren. Speziell die Befehle OPEN und PRINT§ werden benötigt. Der Begriff 'String' und die Aneinanderreihung von Strings sollte Ihnen geläufig sein.

zu V. Kommandos:

Inhalt

Jedes Floppy-Kommando mit Ausnahme der Direktzugriffkommandos ist hier unabhängig von allen anderen erklärt. Beide Übermittlungsarten (soweit vorhanden) werden jeweils erläutert.

Für wen?

Einzelne Kommandos können für Anwender interessant sein. Der Programmierer sollte alle Kommandos gelesen und verstanden haben, um für seine Probleme das entsprechende Werkzeug zu finden.

Möglicherweise können die Kommandos COLLECT, Intialize, CONCAT, RENAME für manche uninteressant sein.

Bei 'Datenverkehr' kann APPEND und eventuell RECORD zurückgestellt werden.

Die Fehlermeldungen sind zum Nachschlagen von Fall zu Fall gedacht.

Voraussetzungen

Kapitel IV sollten Sie in etwa verstanden haben und bei Bedarf nochmal nachschlagen, wie die einzelnen Parameter definiert sind.

Für 2, 3, und 4 benötigen Sie relativ wenig Vorkenntnisse (etwa wie für Kapitel IV).

Für 'Datenverkehr' dürften mittlere bis gute BASIC und allgemeine Programmierkenntnisse erforderlich sein, weil Sie erst dann so weit sein können, diese Anweisungen zu benötigen. Vor allem für die Behandlung von REL-Dateien trifft dies zu. 'Datenverkehr' stellt nur das Werkzeug (die Kommandos) nicht aber Zusammenhänge dar. Dies wird in Kapitel VI. gemacht. Möglicherweise sollten Sie am Anfang zwischen diesen beiden Kapiteln hin- und herlesen.

zu VI. Dateibehandlung und Datenverkehr:

Inhalt

Hier wird sowohl generell in Dateikonzepte eingeführt, wenn auch nur kurz und dann an Hand von Beispielsprogrammen die beiden hier vorhandenen Dateien mit ihrer Behandlung erklärt.

Für wen?

Für jeden Programmierer, der serielle oder Satz-Dateien benötigt. Für Anwender ist ab diesem Kapitel nichts mehr zu holen.

Voraussetzungen

Siehe letzten Absatz über das vorhergehende Kapitel.

zu VII. DOS-Übersicht:

Inhalt

Beschreibung des DOS (Disk Operating System) mit guten Worten, Bild und Tabellen.

Für wen?

Für jeden interessierten Programmierer zum 'nur mal reinschauen'.

Voraussetzungen

Wer diese Information anwenden will, muß maschinennahe Programmierung beherrschen oder Assembler-Programmierer sein.

Diese Information ist nur in Zusammenhang mit Kapitel VIII. sinnvoll.

zu VIII. DOS-Hilfsroutinen:

Inhalt

Hier werden maschinennahe Routinen des DOS beschrieben, die den Aufbau eigener Dateiverwaltungen erlauben (etwa analog zu REL-Dateien).

Für wen?

Maschinennahe BASIC-Programmierer oder Assembler-Programmierer.

Voraussetzungen

Kapitel VII und gute Nerven.

Wir hoffen, daß diese Übersicht verhindert, daß Sie Information zu verstehen versuchen, die Ihnen nur Frustrationen einbringen kann. Den ernsten Leser bitten wir, die letzten Witzeleien dieses Handbuches zu entschuldigen und versichern ihm, daß es ab hier nichts mehr zu lachen gibt.

II. AUSPACKEN, INBETRIEBNAHME, GERÄTEBESCHREIBUNG

1. Auspacken des Gerätes

Ehe Sie Ihr Floppy auspacken, sollten Sie den Karton auf äußere **Beschädigungen** überprüfen. Wenn Sie welche feststellen, achten Sie beim Auspacken des Gerätes bitte besonders auf Beschädigungen des Inhalts. Bei Beschädigung benachrichtigen Sie bitte sofort die Spedition bzw. Ihren Fachhändler.

Werfen Sie die **Spezialverpackung** nicht weg. Beim Transport durch Spedition ist das Gerät nur durch diese Verpackung optimal geschützt.

Der Karton sollte außer dem **Handbuch** und dem **CBM dual drive floppy disk 8050** noch eine **Garantiekarte** enthalten. Wenn nicht, benachrichtigen Sie bitte Ihren Fachhändler.

Zusätzlich müssen Sie noch das richtige **Verbindungskabel** für den Rechneranschluß haben, das **nicht im Lieferumfang des Floppys enthalten** ist:

2. Verbindungkabel

Das Floppy wird mit dem Rechner über die IEEE-488-Schnittstelle verbunden, an die auch andere Peripheriegeräte für den cbm-Rechner angeschlossen werden können.

Durch das Prinzip dieser Schnittstelle können nicht alle Verbindungskabel direkt am Rechner eingesteckt werden. Sie müssen bei einem anderen Peripheriegerät mit eingesteckt werden und haben dadurch Verbindung mit dem Rechner.

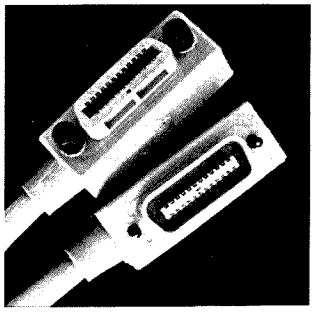
Diese Verbindung von Gerät zu Gerät wird durch das 'Huckepack'-Prinzip des Anschlußsteckers ermöglicht: Jeder Stecker kann einen weiteren Stecker aufnehmen. Ob Sie nun mehrere Geräte ketten- oder sternförmig zusammenschließen, hängt von Ihrer Geräteverteilung ab. Elektrisch ist die Anschlußart unwichtig. Beachten Sie aber, daß die mechanische Belastbarkeit bei sternförmigem Anschluß nicht sehr groß ist.

Es gibt **zwei verschiedene Anschlußkabel:**

Vom **Rechner zum ersten Peripheriegerät** benötigen Sie ein Verbindungskabel **Rechner-Peripherie** . Der **flache Stecker** des Kabels wird am Rechner angesteckt: Wenn Sie von hinten auf den Rechner schauen, müssen Sie den Stecker in den **rechten** der beiden breiten Anschlüsse stecken und zwar so, daß der 'COMMODORE'-Schriftzug nach oben zeigt!

Zwischen den einzelnen Peripheriegeräten benötigen Sie ein Kabel **Peripherie--Peripherie.** Dieses hat an **beiden Enden gleiche Stecker.** Die beiden **Rändelschrauben müssen angezogen werden,** ehe Sie den nächsten Stecker 'huckepack' aufstecken können.

Kabel Computer-Peripherie



Kabel Peripherie-Peripherie

3. Rechneranschluß

Schließen Sie das Floppy mit dem richtigen Verbindungskabel an Ihr bestehendes CBM-System an.

Achten Sie beim Anschluß darauf, daß alle Geräte ausgeschaltet sind.

4. Bedien- und Anzeigeelemente

Der **Netzschalter** befindet sich an der Rückseite links oben. Daneben ist die **Netzsicherung** angeordnet.

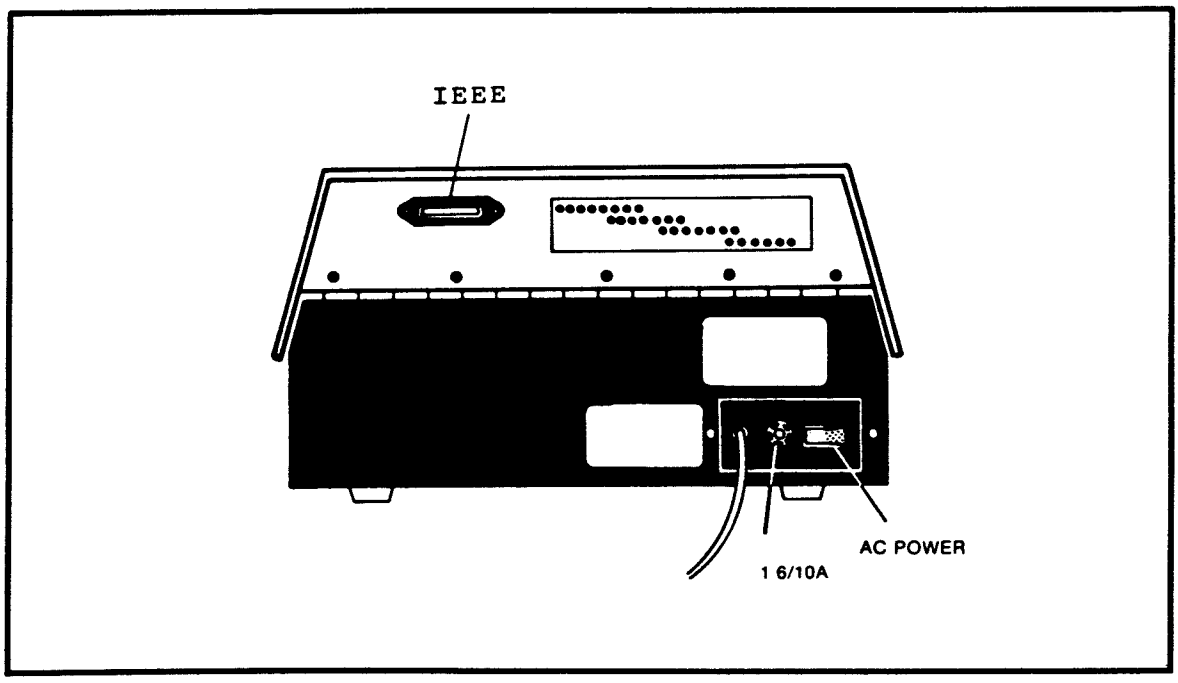
Der Stecker für das **Verbindungskabel** befindet sich an der Rückseite rechts oben. Von hinten betrachtet, muß das Kabel nach **links** vom Stecker weggehen!

Das Floppy 8050 enthält zwei Diskettenlaufwerke, an denen sich rechts oben je eine grüne Kontrolleuchte befindet. Diese Leuchte zeigt an, ob das Laufwerk in Betrieb ist, bzw. ob eine Datei darauf geöffnet ist.

Eine weitere Kontrollleuchte befindet sich in der Mitte zwischen den Laufwerken. Diese Leuchte hat zwei Funktionen:

1. Wenn das Gerät eingeschaltet wird, leuchtet sie **grün** und zeigt so an, daß das **Gerät in Betrieb** ist.

2. Wenn **Fehler** auftreten, leuchtet sie **rot** und zeigt damit an, daß eine Fehlermeldung ansteht, die vom Rechner abgefragt werden kann. (siehe Fehlermeldungen)



5. Inbetriebnahme des Systems

Schließen Sie die Netzstecker aller Geräte an das Stromnetz an. Achten Sie darauf, daß in dem betreffenden Netz keine starken Spannungsschwankungen auftreten, da dies die ordnungsgemäße Funktion des Systems beeinträchtigen kann.

Stellen Sie die Verbindungen zwischen den Geräten durch die entsprechenden Verbindungskabel her.

Beim Einschalten des Sytems **müssen immer die Laufwerksklappen geöffnet** (oben) sein.

Bei der ersten Inbetriebnahme sollten Sie zuerst das Floppy alleine einschalten, um die Reaktion der Kontrolleuchten beim Einschalten des Floppys und dann nochmal beim Einschalten des Rechners beobachten zu können.

6. Automatischer Selbsttest

Sie sollten bei jedem Einschalten des Floppys die Reaktion der Kontrolleuchten beobachten. Das Gerät führt nämlich beim Einschalten einen Speicher- und Betriebssystemtest durch. Ein Fehler wird dann durch die drei Kontrolleuchten dargestellt.

Die Lampen leuchten nach dem Einschalten zweimal in folgender Reihenfolge auf:

1. die mittlere rot, die äußeren grün
2. die mittlere grün, die äußeren gehen aus.

Der letzte Zustand bleibt bestehen, solange das Floppy nicht vom Rechner angesprochen wird.

Ist das Floppy an den Rechner angeschlossen und eingeschaltet, so reagiert das Floppy beim Ein- und Ausschalten des Rechners in derselben Weise, wie eben beschrieben.

Falls die Kontrolleuchten nicht in der beschriebenen Form reagieren, warten Sie bitte einige Sekunden und versuchen es nochmal. Funktioniert das Gerät dann immer noch nicht einwandfrei, sollten Sie es vom Commodore-Service überprüfen lassen.

7. Was ist ein 'dual drive floppy disk'?

In Deutschland hat sich die Kurz-Bezeichnung 'Floppy-(Doppel)-Laufwerk' eingebürgert. Gemeint ist ein Gerät, das gleichzeitig zwei 'Disketten' aufnehmen kann, um auf ihnen Daten abzuspeichern, bzw. Daten von ihnen zu lesen. In Zukunft werden wir das Gerät nur noch kurz als 'Floppy' bezeichnen und das Speichermedium als 'Disketten'.

Das CBM 8050 Floppy ist ein 'Massenspeicher', der über den IEEE-488 Bus mit einem CBM-Rechner zusammenarbeitet. Durch den Einsatz modernster Micro-Elektronik, wie z.B. zweier Microprozessoren, ist das Gerät so 'intelligent', daß es viele Funktionen unabhängig vom Rechner durchführen kann, nachdem es den Befehl dazu bekommen hat. Dies erspart dem Anwender Leerlaufzeiten, entlastet den Programmierer und bewirkt hohe Datenübertragungsraten.

8. Wofür benötigt man ein Floppy?

Ein Floppy wird dazu benutzt, größere Mengen von Informationen abzuspeichern. (Ein 8050 kann auf über eine Million Zeichen zugreifen.) Dabei sind folgende Anwendungsfälle zu unterscheiden:

Abspeichern von Information, die über **lange Zeit** verfügbar sein soll, auch wenn der Rechner zwischendurch ausgeschaltet worden ist. Dazu gehören Programme oder Daten, die der Anwender später wieder im Zugriff haben will.

Zwischenspeichern von Daten, die der Rechner zeitweise nicht in seinem Arbeitsspeicher halten kann, weil die Datenmenge im Moment zu groß wird und diese Daten erst in einem anderen Programmteil wieder benötigt werden.

Bestimmte Daten können innerhalb von wenigen Sekunden gefunden werden und pro Sekunde kann das Floppy etwa zweitausend Zeichen verarbeiten.

9. Wo werden die Daten gespeichert?

Das Floppy speichert die Daten auf sogenannten 'Floppy-Disks' ab. Die genaue deutsche Bezeichnung dafür könnte sein: 'flexible Magnetplatte'. Da diese Bezeichnung umständlich und ungebräuchlich ist, wird in diesem Handbuch das allgemein gebräuchliche Wort 'Diskette' verwendet.

Eine Diskette ist eine sehr dünne flexible Magnetscheibe, die genauso wie ein Tonband mit einer magnetischen Oxidschicht versehen ist. Die Information wird auf die Diskette aufgetragen, indem diese Schicht in bestimmter Weise magnetisiert wird.

Zur besseren Handhabung ist die Platte fest umschlossen von einer quadratischen Schutzhülle, in der sie auch während des Betriebes verbleibt. Diese Hülle hat eine runde Öffnung für den Plattenantrieb in der Mitte und eine lange ovale für den Schreib-Lese-Kopf.

Es gibt Disketten in verschiedenen Größen. Für dieses Gerät benötigen Sie sogenannte 'Mini-Disketten'. Diese sind 13 mal 13 cm groß (5 1/4 ")

Beim Kauf sollten Sie darauf achten, daß Sie nicht Disketten schlechter Qualität bekommen, die dann meist auch billiger sind. Wenn Sie sich nicht mit den verschiedenen Fabrikaten auskennen, sollten Sie sich auf jeden Fall beraten lassen.

10. Technische Daten

Speicherkapazität

Gesamtkapazität	533248 Bytes pro Diskette
Serielle Dateien	521208 Bytes pro Diskette
Relative Dateien	464312 bis 517398 Bytes pro Diskette abhängig von der Satzgröße 182880 Bytes pro Datei 65535 Sätze pro Datei
Directory-Einträge	224 pro Diskette
Sektoren pro Spur	23 bis 29
Bytes pro Sektor	256 (nutzbar: 254 bei PRG-, SEQ-, REL-Dateien bzw. 255 bei Direkt-Zugriff)
Spuren	77
Blöcke	2083 (davon 2052 für den Benutzer verfügbar)

IC's

Controller	6505	Microprozessor
	6530	I/O, RAM, ROM
	6522	I/O, Timer
Interface	6502	Microprozessor
	6532 (2)	I/O, RAM, Timer
	6564 (2)	ROM
Gemeinsam	6114 (8)	4*1K RAM

Physikalische Eigenschaften

Material	Stahlgehäuse
Abmessungen	18 (H) * 38 (B) * 40 (T)
Gewicht	12 kg

Elektrische Eigenschaften

Spannung	220 V
Frequenz	50 or 60 Hz
Leistungsaufnahme	50 W

Speichermedium

Mini - Disketten	5 1/4 " einseitig, single density softsektoriert (siehe III.5.5)

III. HINWEISE FÜR DEN BEDIENER

1. Einschalten des Systems

Prinzipiell ist die Reihenfolge, in der die einzelnen Geräte eingeschaltet werden, beliebig. Beachten Sie aber, daß mehr als die Hälfte aller Geräte, die am IEC-Bus hängen, eingeschaltet sind, ehe Sie Daten über den Bus senden. Anderenfalls wird die Datenübertragung über den Bus gestört.

Wenn Sie die ganze Computeranlage an eine Steckerleiste mit Zentralschalter anschließen, ersparen Sie sich die Arbeit, die einzelnen Geräte einzuschalten.

Wenn allerdings der Rechner kurz nach den Peripheriegeräten eingeschaltet wird, erhalten die Geräte vom Rechner ein 'Rücksetzsignal' (Reset) und machen nochmal die gleiche Initialisierung durch, wie wenn sie alleine eingeschaltet werden. Dies hat den Vorteil, daß Sie an dieser Reaktion sehen können, ob die Verbindungen zwischen Rechner und Peripheriegeräten in Ordnung sind.

In welcher Reihenfolge Sie auch einschalten, achten Sie immer darauf, daß die Klappen der Laufwerke offen sind, da das Floppy bei einem Reset 'nicht zurechnungsfähig ist' und deshalb möglicherweise Information löschen kann.

Wegen der automatischen Initialisierung dürfen die Disketten nicht im Schlitten **eingerastet** sein!

2. Klappenbehandlung

Leere Klappe

Wenn keine Diskette im Laufwerk ist, ist die Klappe immer oben. Versuchen Sie nicht, die leere Klappe nach unten zu drücken, da sie mechanisch gesperrt ist.

Diskette einlegen

Wenn Sie eine Diskette einschieben, achten Sie auf mögliche Widerstände durch Verkanten der Diskette. Wenn die Diskette gerade im Laufwerk verschwunden ist, stoßen Sie auf Widerstand. Jetzt müssen Sie die Diskette gegen diesen Widerstand ganz ins Laufwerk schieben. Kurz bevor Sie mit den Fingern an der zurückgesetzten Wand anstossen, hören Sie ein Klicken. Jetzt ist die Diskette richtig in den Schlitten eingerastet.

Schlitten senken

Um die Diskette endgültig in ihre Arbeitsposition zu bringen, drücken Sie den geriffelten Klappengriff bis zum Anschlag nach unten. Sie hören, daß dabei kurz der Motor anläuft. Dadurch wird die Diskette sauber zentriert. Achten Sie darauf, daß die Klappe zu ist, wenn der Motor zu laufen aufhört, sonst ist möglicherweise die Diskette nicht richtig zentriert. Andererseits dürfen Sie die Klappe nicht ruckartig nach unten ziehen, weil sonst die Diskette keine Zeit hat, sich richtig zu zentrieren.

Schlitten heben

Wenn der Schlitten wieder nach oben kommen soll, drücken Sie den Griff nochmals nach unten und lassen ihn dann nach oben gleiten (nicht schnappen lassen!). Die Diskette ist dann immer noch richtig im Schlitten, der Schreib/Lesekopf sitzt aber nicht mehr auf.

Diskette entnehmen

Zum Entnehmen der Diskette schieben Sie den Griff nach oben. Dadurch kommt die Diskette so weit aus dem Schlitten, daß sie bequem herausgenommen werden kann.

Eine Diskette darf nur entnommen werden, wenn die Leuchte des entsprechenden Laufwerks erloschen ist!

3. Diskettenbehandlung

Disketten sind relativ robuste Datenträger. Dennoch können sie bei unsachgemäßer Behandlung Schaden nehmen. Hier sollen deshalb einige wichtige Hinweise gegeben werden:

Die Diskette steckt in einer Schutzhülle. Diese Hülle hat oben und unten je ein ovales Loch. ('Oben' ist die Seite mit dem Etikett!) Unter allen Umständen ist die Berührung der Diskettenoberfläche durch diese Öffnungen zu vermeiden! Da die Diskette auf der Unterseite beschrieben wird, darf diese auf **keinen Fall** mit den **Fingern berührt** werden und ist auch vor Staub zu schützen. Staub kann nicht nur die Disketten, sondern auch die Schreib- / Leseköpfe angreifen oder zerstören.

Um solchen Berührungen vorzubeugen, sollte man sich angewöhnen, Disketten nicht ohne Tasche herumliegen zu lassen. Eine Diskette sollte entweder in der Schutztasche oder im Laufwerk stecken! Klebeetiketten klebt oder entfernt man am besten, während die Diskette in der Tasche steckt.

Während des **Ein- oder Ausschaltens** des Floppys oder des Rechners sollten die **Klappen geöffnet** sein, damit die Schreib- / Leseköpfe nicht auf den Disketten aufliegen!

Natürlich dürfen Disketten nicht starken magnetischen Feldern oder direkter Sonneneinstrahlung ausgesetzt sein.

Vor dem Einlegen einer Diskette sollte man sie von Hand in ihrer Schutzhülle **zentrieren**, weil sonst u.U. der Rand des Mittelloches verbogen wird und dann die Diskette kaum noch zu zentrieren ist. Dies ist nach unserer Erfahrung die häufigste Ursache für unbrauchbare Disketten!

4. Datensicherung

Sowohl durch Hardwarefehler (Schmutz usw.), als auch durch Bedienungsfehler im Softwarebereich kann der Inhalt einer Diskette ganz oder teilweise zerstört bzw. unleserlich werden.

Deshalb gilt bei Disketten die gleiche Regel wie bei allen Datenträgern:

Jede Datei muß mindestens eine Kopie haben!

Dies kann durch verschiedene Maßnahmen erreicht werden:

Man kann ganze Disketten duplizieren oder einzelne Dateien kopieren. Dabei stellt sich die Frage, wann diese Kopien erstellt werden sollen: sofort, stündlich, täglich oder wöchentlich?

Vielleicht hilft bei der Entscheidung folgende Regel: Man sollte immer davon ausgehen, daß **jederzeit** ein Fehler auftreten kann. Deshalb sollte man nur solange auf eine Kopie verzichten, solange der Aufwand für die Wiederherstellung der Daten nicht wesentlich größer ist, als der Aufwand für die Erstellung der Kopie.

Nach der Ersterfassung kann man nach dem Vater / Sohn / Enkel Prinzip vorgehen: Wenn in einer Datei Änderungen vorgenommen werden, ist die neue Datei der 'Sohn' der alten Datei, des 'Vaters'. Wird der 'Sohn' geändert, entsteht daraus ein 'Enkel'. Nun lautet die Regel, daß der 'Vater' erst gelöscht werden darf, wenn der 'Enkel' erfolgreich abgespeichert wurde. Das bedeutet, er mußte mindestens einmal gelesen werden können. Dadurch hat man immer mindestens zwei 'Versionen' einer Datei, die sich nur durch Änderungen unterscheiden sollten, die relativ geringfügig sind.

Je nachdem, wieviel Platz vorhanden ist, kann man auch mehr als zwei Generationen zulassen. Natürlich ist es gut, wenn die verschiedenen Generationen auf verschiedenen Disketten liegen. Aber auch auf der gleichen Diskette ist es sinnvoll, da bei den meisten Fehlern nicht die ganze Diskette unleserlich wird.

5. Anwender-Befehle

Auch als Nur-Anwender müssen Sie einige wenige BASIC-Befehle beherrschen, sofern Sie nicht sehr ausgefeilte Software verwenden. Wir führen deshalb die wichtigsten Befehle auch hier im Floppy-Handbuch nochmal auf.

Eine kurze Übersicht soll Ihnen zeigen, welche Befehle im folgenden erklärt werden:

BASIC-4	BASIC-3	
SHIFT/RUN	LOAD"*",8 / RUN	erstes Programm laden und starten
DLOAD	LOAD	beliebiges Programm laden
RUN	RUN	Programm starten
DIRECTORY	LOAD"$...",8	Inhaltsverzeichnis der Diskette lesen
BACKUP	Duplicate	Diskette duplizieren (Sicherungsduplikat)
HEADER	New	Neue Diskette anlegen
DSAVE	SAVE	Programm abspeichern
COPY	Copy	Datei kopieren
SCRATCH	Scratch	Datei löschen
?DS$	10 INPUT§15,F,F$	Fehlermeldung abfragen

Die Befehle sind in der Reihenfolge erklärt, wie Sie von Ihnen am wahrscheinlichsten benötigt werden. D.h. die weiter hinten beschriebenen Befehle benötigen Sie eventuell am Anfang gar nicht.

Da Sie eventuell Ihr Floppy 8050 an einen Rechner der 3000-er Serie angeschlossen haben, führen wir die Befehle beider BASIC-Versionen an. Die erste Möglichkeit ist für BASIC-4, die zweite für BASIC-3.

5.1 Erstes Programm laden und starten

BASIC-4:

Wenn Sie die Tasten **SHIFT und RUN** drücken, nachdem Sie die Anlage eingeschaltet haben, wird das erste Programm der Diskette im rechten Laufwerk geladen und automatisch gestartet.

BASIC-3:

sofort nach dem Einschalten:
```
LOAD"*",8
RUN
```

sonst:
```
LOAD"0:*",8
RUN
```

Bei qualitativ hochwertiger Software ist dieses erste Programm ein sogenannter Systemstart, der erstens alle benötigten Programme automatisch lädt und zweitens ab sofort mit Ihnen in deutschem Dialog in Verbindung tritt. In der Regel können Sie dann durch Drücken einer Taste ein bestimmtes Programm aus einer ganzen Anzahl von angebotenen Programmen laden und starten.

5.2 Inhaltsverzeichnis der Diskette lesen

Wenn Sie wissen wollen, welche Programme oder Dateien auf einer Diskette sind, können Sie durch

DIRECTORY oder
LOAD"$",8 und **LIST**

die Inhaltsverzeichnisse beider Disketten nacheinander auf den Bildschirm bringen. Denken Sie dabei an die Funktion der beiden Tasten ':/*' bzw. 'Pfeil nach links' beim 8032.

Durch Anfügen von 'D0' oder 'D1' bei 'DIRECTORY' können Sie gezielt das Inhaltsverzeichnis der rechten oder linken Diskette lesen (BASIC-4)

```
DIr D0
DIr D1
```

Wenn Sie den dritten Buchstaben von DIRECTORY **mit SHIFT** eingeben, versteht dies der Rechner als Abkürzung und Sie müssen das furchtbar lange Wort nicht immer ausschreiben.

BASIC-3:

```
LOAD"$0",8 und LIST
LOAD"$1",8 und LIST
```

Hierbei wird ein im Speicher stehendes BASIC-Programm überschrieben.

5.3 Beliebiges Programm laden

Falls Sie unabhängig von einem 'Systemstart-Programm' ein bestimmtes Programm einer Diskette laden wollen, können Sie dies durch

```
DLOAD "NAME"              oder
LOAD  "0:NAME" , 8
```

tun. Dieser Befehl lädt das Programm 'NAME' vom **rechten Laufwerk** in den Arbeitsspeicher. Er teilt den Ladevorgang durch 'LOADING' mit und meldet sich mit 'READY.' zurück.

Falls das gewünschte Programm auf der eingelegten Diskette nicht vorhanden ist oder irgendein Diskettenfehler auftritt, wird

FILE NOT FOUND ERROR

gemeldet. Überprüfen Sie in diesem Fall die richtige Schreibweise des Namens und ob die richtige Diskette eingelegt ist.

Wollen Sie ein Programm vom **linken Laufwerk** laden, so fügen Sie an den oben beschriebenen Befehl noch durch Komma getrennt 'D1' an:

```
DLOAD "NAME" , D1         oder
LOAD "1:NAME" , 8
```

5.4 Programm starten

Ein Programm, das Sie durch (D)LOAD in den Speicher geladen haben, müssen Sie noch durch

RUN

starten, damit es für Sie tätig werden kann.

5.5 Neue Diskette anlegen

Fabrikneue Disketten müssen erst durch einen speziellen Befehl 'sektoriert' werden, ehe sie Programme bzw. Daten aufnehmen können.

```
HEADER "NAME" , IXX , D0          oder
OPEN1,8,15,"N0:NAME,XX"
```

Durch HEADER (nicht bei OPEN) wird die Frage

ARE YOU SHURE?

ausgelöst, die sicherstellen soll, daß Sie nicht aus Versehen eine Diskette löschen, die bereits Daten enthält. Diese Frage ist mit 'Y' oder 'YES' zu beantworten, sonst wird die HEADER-Anweisung ignoriert.

'NAME' ist der Diskettenname, er darf maximal 16 Zeichen lang sein. Die beiden XX hinter I stehen stellvertretend für die 'Identität' der Diskette. Diese Kennzeichnung ist zwei beliebige Zeichen lang.

Lassen Sie den Zusatz 'IXX' weg, so wird die Diskette nur gelöscht, aber nicht sektoriert. Dies geht nur bei Disketten, die bereits in Gebrauch waren.

Durch die oben angeführte Anweisung wird die Diskette im rechten Laufwerk sektoriert. Bei Angabe von 'D1' statt 'D0' wird die Diskette im linken Laufwerk sektoriert. Die Reihenfolge der einzelnen Angaben NAME, I oder D ist unwichtig, wichtig ist nur, daß sie jeweils durch Komma getrennt werden.

5.6 Diskette duplizieren

Von Datendisketten kann nicht oft genug ein Sicherungsduplikat erstellt werden, um bösen Überraschungen durch Verlust wertvoller Daten vorzubeugen. Sie können Disketten vollständig duplizieren mit

BASIC-4:

BACKUP D0 TO D1	**rechte Diskette auf linke übertragen**
(BACKUP D1 TO D0	linke Diskette auf rechte übertragen)

Die Form des BACKUP ist:

BACKUP (dupliziere von) D (Laufwerk) 0 TO (nach) D (Laufwerk) 1

Links von 'TO' steht also das Quellaufwerk, rechts das Ziellaufwerk.

Wenn Sie immer die gleiche Richtung (D0 TO D1) verwenden, laufen Sie weniger Gefahr, irgendwann die Richtung zu verwechseln.

Die Diskette in dem Laufwerk, das rechts von TO aufgeführt ist, wird völlig gelöscht!

BASIC-2

OPEN1,8,15,	**"D1=0"**	**rechte Diskette auf linke übertragen**
(	"D0=1"	linke Diskette auf rechte übertragen)

Links von '=' steht das Ziellaufwerk, rechts das Quelllaufwerk.

Wenn Sie immer die gleiche Richtung (1=0) verwenden, laufen Sie weniger Gefahr, irgendwann die Richtung zu verwechseln.

Die Diskette in dem Laufwerk, das links von = aufgeführt ist, wird völlig gelöscht!

Da eine falsche Anwendung dieses Befehls die Vernichtung der Daten zur Folge haben kann, die Sie eigentlich sichern wollten, beachten Sie bitte die folgenden Hinweise genau:

Schützen Sie die 'Quelldiskette' immer mit einem 'Schreibschutzaufkleber'!

Vergewissern Sie sich, daß die 'Zieldiskette' wirklich leer ist bzw. gelöscht werden darf!

5.7 Programm abspeichern

Ein im Arbeitsspeicher stehendes BASIC-Programm können Sie mit

```
DSAVE "NAME"                    oder
SAVE "0:NAME" , 8
```

auf die Diskette im rechten Laufwerk abspeichern. Auch hier kann durch Anfügen von D1 auf das linke Laufwerk abgespeichert werden.

5.8 Datei kopieren

Einzelne Dateien können von einem Laufwerk auf das andere durch

```
COPY "NAME" , D0 TO "NAME" , D1    oder
OPEN1,8,15,"C1:NAME=0:NAME"
```

kopiert werden. Wie schon bei BACKUP steht links von TO das Quellaufwerk und rechts das Ziellaufwerk. Neu sind die beiden Namen. Auch hier steht links der Name der Datei, die kopiert werden soll und rechts der Name der neuen Datei. Sie können - müssen aber nicht - gleich sein.

Wenn kein Name angegeben wird, werden alle Dateien der Quelldiskette auf die Zieldiskette kopiert. Der Unterschied zu BACKUP besteht darin, daß bei COPY die Zieldiskette nicht gelöscht wird, sie kann also bereits Dateien enthalten, zu denen noch die Dateien der Quelldiskette hinzukopiert werden.

5.9 Datei löschen

Einzelne Dateien können Sie durch die Anweisung

```
SCRATCH "NAME"                  oder
OPEN1,8,15,"S0:NAME"
```

löschen. Wenn der Zusatz 'D1' fehlt, wird auf dem rechten Laufwerk gelöscht. Wie bei HEADER wird bei SCRATCH mit

```
ARE YOU SHURE?
```

nachgefragt, ob Sie dies ernst meinen. Antworten Sie auch hier durch 'Y'. Nach erfolgreichem Löschen wird gemeldet:

```
01 FILES SCRATCHED 01 00
```

Die zweite Zahl von rechts sagt Ihnen, daß 1 Datei gelöscht wurde.

Eine gelöschte Datei kann nicht wieder zurückgeholt werden, wenden Sie diese Anweisung also ähnlich sorgfältig an wie BACKUP und HEADER.

5.10 Fehlermeldung abfragen

Wenn die mittlere Kontrollleuchte rot brennt, ist ein Fehler aufgetreten. Die Fehlermeldung können Sie abfragen durch

```
?DS$
```

Bei BASIC-3 ist dies etwas komplizierter, da Sie ein kleines Programm schreiben müssen:

```
10 OPEN1,8,15 : INPUT§1,F,F$,SP,SE : ?F,F$,SP,SE
```

Nach GOTO 10 erhalten Sie die Fehlermeldung auf den Bildschirm gedruckt.

IV. DIE KOMMUNIKATIONSMÖGLICHKEITEN ZWISCHEN RECHNER UND FLOPPY UND IHRE PARAMETER

1. Einführung

Das Floppy 8050 ist ein 'hochintelligentes' Peripheriegerät, das komplexe Operationen auf Grund eines einzigen Kommandos vom Rechner selbständig und unabhängig vom Rechner durchführen kann.

Die meisten Kommandos können entweder unmittelbar über Tastatur und Bildschirm (Direktmodus) oder von einem Programm aus an das Floppy übermittelt werden.

Das Floppy erwartet die Kommandos des Rechners immer in derselben Art. Dagegen gibt es im Rechner verschiedene Darstellungsmöglichkeiten für identische Floppy-Aufträge. Dazu kommt, daß abhängig von der Darstellungsart verschiedene Parameter erforderlich sind, bzw. wegfallen können, obwohl das Floppy immer dieselben Daten erhalten muß.

Die verschiedenen Darstellungsarten gepaart mit den gleichen Parametern in verschiedenen Formen wirken auf den ersten Blick verwirrend. Um den Zugang zu erleichtern, haben wir folgenden Weg gewählt: Wir beschreiben jeweils zuerst den Floppy-Auftrag mit seiner grundsätzlichen Wirkung im Floppy und unmittelbar anschließend die verschiedenen Übermittlungsmöglichkeiten des Auftrags.

Ehe auf Einzelheiten eingegangen werden kann, wird ein Überblick über die einzelnen Aufträge aus der Sicht des Floppys gegeben:

Drei verschiedene Tätigkeiten des Floppys sind zu unterscheiden:

(1) Daten vom Rechner übernehmen und auf Diskette schreiben oder
Daten von Diskette lesen und an den Rechner übergeben

(2) Aufträge empfangen und quittieren

(3) Verwaltungsaufträge unabhängig vom Rechner ausführen

Punkt 1 beschreibt die eigentliche Aufgabe des Floppys: Daten abspeichern und später wieder zurückgeben. Wir werden darauf gleich unter 'Programm- und Datenbehandlung' etwas näher eingehen.

Punkt 3 unterscheidet sich von 1 in zweierlei Hinsicht: Erstens ist das Floppy durch solche Verwaltungsaufträge u.U. bis zu Minuten mit sich selbst beschäftigt, ohne den Rechner aufzuhalten, aber auch ohne selbst ansprechbar zu sein. Zweitens unterscheidet sich die Art der Übermittlung solcher Aufträge von den Kommandos der Datenbehandlung.

Womit wir bei Punkt 2 wären: Ehe das Floppy etwas arbeiten kann, muß es vom Rechner erfahren, was. Und wenn der Auftrag abgeschlossen ist, muß der Rechner erfahren können, ob alles gutgegangen ist.

Nach diesem groben Überblick aus der Sicht des Floppys wollen wir das Ganze aus der Sicht des Rechners betrachten und können dabei die Betrachtung etwas verfeinern:

1.1 Programm- und Datenbehandlung

Obwohl für das Floppy die Unterscheidung zwischen Programmen und Daten eigentlich unwichtig ist, hat sie beim Rechner durchaus ihre Berechtigung: Der Rechner bzw. die BASIC-Befehle unterscheiden zwischen dem Abspeichern und Lesen von Programmen einerseits und von Daten, die im Programm verarbeitet werden, andererseits.

Die Befehle für Programme laden und speichern sind: (D)LOAD, (D)SAVE und VERIFY.

Für lesen und abspeichern von Daten sind zuständig (D)OPEN, (APPEND), (RECORD), PRINT§, INPUT§, GET§ und (D)CLOSE.

Die eingeklammerten Befehle sind nur ab BASIC-4 vorhanden. Alle Funktionen können auch mit dem Befehlsvorrat von BASIC-3 abgerufen werden. Die 'Disk--BASIC-Befehle' bieten lediglich mehr Komfort, vor allem im Direktmodus.

Die Programmbehandlung ist wesentlich einfacher als die Datenbehandlung, da erstens Programme eine sehr einfache Übermittlungs-Struktur haben und zweitens immer nur 1 Programm gleichzeitig gelesen oder abgespeichert wird.

Dagegen gibt es zwei grundsätzlich verschiedene Datenstrukturen (SEQ/REL) und es können mehrere 'Datenkanäle' oder 'Dateien' gleichzeitig offen (in Betrieb) sein. Aus diesem Grund gibt es für diese Aufgabenstellung mehr Befehle und mehr Parameter.

1.2 Kommandoübermittlung / Quittung

Die oben angesprochenen Verwaltungsaufträge (z.B. Datei kopieren) werden prinzipiell über den sogenannten Kommandokanal übertragen. Bei den Disk-Befehlen des BASIC-4 geschieht die Übersetzung des BASIC-Kommandos in das Kommandostringformat, sowie die Übermittlung ans Floppy automatisch. Dagegen muß bei der anderen Möglichkeit, die bei BASIC-3 und -4 funktioniert, durch OPEN der Kommandokanal geöffnet, und durch PRINT§ der Kommandostring übertragen werden.

Die Quittung des Floppy wird bei BASIC-4 durch die Variablen-Funktion DS oder DS$ abgefragt, bei BASIC-3 etwas umständlicher durch INPUT§ von maximal vier Variablenwerten vom Kommandokanal.

2. Übermittlungsmöglichkeiten

2.1 Einführung

Die Tabelle 'Kommando-Übersicht' zeigt, daß es - von BASIC aus betrachtet - drei Arten von Kommandos gibt. Diese drei Arten unterscheiden sich in den Übermittlungsmöglichkeiten:

(1) Kommandostring oder Disk-BASIC (z.B. HEADER)

(2) Nur Kommandostring (z.B. M-W)

(3) Normales BASIC oder Disk-BASIC (z.B. LOAD)

Ehe erklärt wird, welche Übermittlungsmöglichkeit jeweils günstiger ist, soll kurz der Grund für die verschiedenen Möglichkeiten aufgezeigt werden:

Die Methode, den Kommandostring direkt im Programm zu erzeugen und durch PRINT§ über den Kommandokanal zu übermitteln, verursacht im Programm einen erträglichen Aufwand, wenngleich die Disk-BASIC-Befehle weniger Aufwand verursachen. Aber im Direktmodus ist diese Art der Kommandoübermittlung sehr schwerfällig und vor allem Nur-Anwendern kaum zumutbar. Die Fehlerabfrage ist sogar im Direktmodus überhaupt nicht möglich (BASIC-3).

Deshalb wurden ab BASIC-4 weitere Befehle implementiert, die die Eingabe der meisten Floppy-Befehle wesentlich komfortabler gestalten, als dies bei der direkten Kommandostring-Methode der Fall ist. Im Direktmodus ist auf jeden Fall die Verwendung der Disk-BASIC-Befehle zu empfehlen.

Dagegen sollten Sie vor der Verwendung dieser Befehle innerhalb von Programmen überlegen, ob die Programme nur ab BASIC-4 aufwärtskompatibel, oder auch für BASIC-3 gültig sein sollen. Im letzteren Fall müßten Sie auf die Verwendung der Disk-BASIC-Befehle innerhalb des Programmes verzichten.

Falls Sie aber weder für BASIC-3 schreiben, noch die maschinennahen Befehle (Direktzugriff, DOS-Manipulationen) benötigen, für die keine D-BASIC-Befehle existieren, können Sie in allen Kapiteln die Anmerkungen zum Kommandostring übergehen.

Um ein mögliches Mißverständnis auszuschließen, sei noch erwähnt, daß die Kompatibilität zwischen den beiden BASIC-Versionen zu unterscheiden ist von der Kompatibilität zwischen den einzelnen DOS-Versionen. Für das Floppy besteht kein Unterschied, ob es Kommandos über den Umweg eines D-BASIC-Befehls oder direkt durch PRINT§ über den Kommandokanal erhält. Da andererseits die drei DOS-Versionen 1, 2a und 2c (auch als 2.1 und 2.5 bezeichnet) (3040, 4040, 8050) aufwärtskompatibel sind, ist es bei der Auswahl der BASIC-Befehle unerheblich, welches Floppy angeschlossen ist. Allerdings dürfen bei DOS 1 die Befehle APPEND und RECORD nicht verwendet werden, da DOS 1 einmal geschlossene Dateien nicht mehr zum Schreiben öffnen kann und keine REL-Dateien kennt. Außerdem ist COPY aller Dateien nicht möglich (COPY ohne Namensangabe).

Nach dieser kurzen Übersicht sollen nun zuerst die beiden angesprochenen Übermittlungsmöglichkeiten ausführlich behandelt werden und anschließend die einzelnen Parameter.

Wenn Sie das Format der Übermittlungsmöglichkeit, die Sie verwenden wollen, sowie die Bedeutung und Grenzen der einzelnen Parameter verstanden haben, brauchen Sie bei den einzelnen Kommandos nur noch die Wirkung zu verstehen. Das Format und die benötigten Parameter ergeben sich dann aus der Wirkung.

2.2 Kommando-Übersicht

BASIC-4	BASIC-3	Kommandostring	Bedeutung
Disketten-Kommandos			
HEADER		New	formatieren
BACKUP		Duplicate	duplizieren
COLLECT		Validate	überprüfen, bereinigen
DIRECTORY	LOAD"$..		Inhaltsverzeichnis
CATALOG	LOAD"$		Inhaltsverzeichnis
		Initialize	Initialisieren
Datei-Kommandos			
COPY		Copy	kopieren
CONCAT		Copy	zusammenhängen
RENAME		Rename	umbenennen
SCRATCH		Scratch	löschen
Programmbehandlung			
DLOAD	LOAD		laden
DSAVE	SAVE		abspeichern
	VERIFY		überprüfen
Datenverkehr einleiten / beenden			
DOPEN	OPEN		Datenkanal öffnen
APPEND	OPEN		Datei weiterbescheiben
DCLOSE	CLOSE		Datenkanal schließen
Datenverkehr			
RECORD		Position	Record-Zeiger posit.
	PRINT§		Daten schreiben
	INPUT§		Daten einlesen
	GET§		ein Zeichen einlesen
Fehlermeldung lesen			
DS, DS$	OPEN / INPUT§		
Direktzugriff, DOS-Operationen			
		B-R	Block lesen
		B-W	Block schreiben
		B-E	Block ausführen
		B-A	Block belegen
		B-F	Block freigeben
		B-P	Puffer freigeben
		M-W	Speicher beschreiben
		M-R	Speicher lesen
		M-E	Speicher-Routine ausführen
		U	Routine über Sprungleiste

2.3 Kommandos über den Kommandokanal senden

Jedes der Kommandos C D I N P R S V (siehe Kommandozeichen 2.3.3) kann direkt durch PRINT§ über den Kommandokanal übertragen werden.

Im folgenden wird erklärt, was es mit dem Kommandokanal auf sich hat, wie das Format des Kommandostrings generell aussieht und welche Parameter welche Wirkung haben.

2.3.1 Der Kommandokanal

Bei OPEN und in den Kapiteln über Direktzugriff können Sie allgemeine Informationen über die Floppykanäle finden.

Der Kommandokanal ist ein spezieller Kanal mit der Nummer 15 (Sekundäradresse). Über diesen Kanal werden keine Daten zur Aufzeichnung übertragen, sondern nur Kommandos ans Floppy geschickt. Außerdem meldet das Floppy über diesen Kanal, ob Fehler bei einer Operation aufgetreten sind (s. Fehlerbehandlung).

(Die Disk-BASIC-Befehle übertragen die Befehle ans Floppy ebenfalls über den Kommandokanal.)

Der Kommandokanal wird wie jeder andere Kanal **durch OPEN geöffnet:**

OPEN 15,8,15 (s. OPEN)

Durch DOPEN kann der Kommandokanal nicht geöffnet werden, weil DOPEN unbedingt einen Dateinamen braucht, in diesem Fall aber kein Name angegeben werden kann.

Während OPEN nur die Voraussetzung ist, daß PRINT§ funktionieren kann, hat CLOSE des Kommandokanals eine sehr wichtige Neben-Funktion - es schließt alle Dateien dieses Floppys:

CLOSE des Kommandokanals schließt alle Dateien (Kanäle) des Floppys

Der Kommandokanal belegt im Floppy keinen der allgemeinen Puffer, die für die anderen Kanäle (Datenkanäle) belegt werden. Vom Floppy aus gesehen ist es also gleichgültig, ob der Kommandokanal geöffnet ist oder nicht. Dagegen belegt ein geöffneter Kommandokanal im Rechner eine von den zehn möglichen Dateien.

2.3.2 Aufbau des Kommandos

Um das Verständnis der Kommandosyntax zu erleichtern, sollen erst die verschiedenen Informationen besprochen werden, die ans Floppy übermittelt werden müssen. Die Erklärung erfolgt dabei in der Reihenfolge, wie die Informationen im Kommandostring enthalten sind.

Allgemein hat ein Kommando den Aufbau:

kommandozeichen laufwerknummer : dateiname = laufwerknummer2 : dateiname2

Der Kommandostring kann insgesamt **maximal 40 Zeichen** lang sein. (wichtig bei CONCAT = COPY von mehreren Dateien!)

Bei Operationen mit Quelle und Ziel können Sie sich die Richtung leicht merken, wenn Sie als Eselsbrücke die normale BASIC-Wertzuweisung nehmen: Die Variable, die den **neuen** Wert erhält, steht immer **links** vom 'Gleichheits'zeichen. Auch bei den Floppy-Kommandos gilt:

NEU = ALT

oder

ZIEL = QUELLE

Wenn Sie sich außerdem noch merken, daß in der Regel eine Datei erst dann eindeutig gekennzeichnet ist, wenn Sie Laufwerk und Dateinamen angegeben haben, daß also

laufwerk : dateiname

immer eine Einheit bilden, können Sie mit dem Format des Kommandostrings keine Schwierigkeiten mehr haben.

Im Kommandostring dürfen **keine Blanks** enthalten sein, es sei denn, sie sind Bestandteil eines Dateinamens.

2.3.3 Kommandozeichen

Alle Kommandos sind durch 1 Zeichen eindeutig gekennzeichnet. In einem Kommandostring darf nur 1 Kommandozeichen enthalten sein.

C - Datei(en) kopieren
D - Diskette duplizieren
I - Diskette(n) initialisieren
N - Diskette löschen / formatieren
R - Datei umbenennen
S - Datei(en) löschen
V - Serielle Dateien überprüfen / bereinigen
P - Positionieren auf Record und Byte

2.3.4 Laufwerk

Da das Floppy zwei Laufwerke hat, muß man ihm mitteilen, auf welchem die Operation ausgeführt werden soll.

Hinter der Laufwerknummer steht **immer ein Doppelpunkt,** wenn noch ein Dateiname folgt!

Beim **Lesen** kann die Angabe des Laufwerks entfallen, weil das Floppy-DOS zuerst auf dem zuletzt benutzten Laufwerk nachsieht, ob die gewünschte Datei vorhanden ist und wenn nicht, auf dem anderen Laufwerk sucht.

Beim **Schreiben** muß allerdings **unbedingt das Laufwerk** angegeben werden.

2.3.5 Dateiname

Der Dateiname steht **immer nach dem Doppelpunkt der Laufwerksnummer.**

Bei D, I und V wird kein Dateiname gebraucht.
Bei C und S kann ein oder mehrere Dateinamen angegeben werden.
Bei N wird genau ein Name (Diskettenname) angegeben.

2.3.6 Übermittlung des Kommandostrings hinter PRINT§

Dies setzt voraus, daß der **Kommandokanal** geöffnet ist.

Der Aufbau des Kommandostrings hinter PRINT§ kann sehr verschieden sein. Sie können also wie gewohnt alle Eigenschaften der String-Verknüpfungsbefehle, sowie die speziellen PRINT§-Eigenschaften ausnützen. Einige Beispiele sollen dies zeigen:

```
OPENkk,8,15

PRINT§kk,"C1:DATEINEU=0:DATEIALT"
oder
L1$="1":L2$="0":D1$="NEU":D2$="ALT":K$="C"
PRINT§kk,K$;L1$;"DATEI";D1$;"="L2$;"DATEI";D2$
```

Beide Anweisungen kopieren die Datei DATEIALT vom Laufwerk 0 auf das Laufwerk 1 unter dem neuen Namen DATEINEU. Sie können also in gewohnter Weise Variablen einsetzen.

2.3.7 Übermittlung des Kommandostrings hinter OPEN la , gn , 15

Das Kommando kann direkt als dn an das ´OPEN´ des Kommandokanals angehängt werden.

Zwischen dem Senden des Kommandos direkt am ´OPEN´ des **Kommandokanals** bzw. durch ´PRINT§´ ist folgender Unterschied zu beachten:

Bei ´PRINT§´ kehrt der Rechner **sofort** in den Direktmodus bzw. ins Programm zurück und das Floppy bearbeitet unabhängig vom Rechner das Kommando. Das gilt aber nur, wenn das Floppy nicht noch mit der Ausführung eines Kommandos beschäftigt ist, das vorher gegeben wurde.

Bei ´OPEN...´ wartet der Rechner, bis das Floppy den Befehl ausgeführt hat. Da dies beim Kopieren einer Diskette bis zu einigen Minuten dauern kann, ist es ratsam, die Kommandos nicht direkt mit dem OPEN zu senden.

2.3.8 Formatübersicht

Im folgenden wird mit **name** der Ausdruck **laufwerknummer : dateiname** bezeichnet, weil erst Laufwerknummer und Dateiname zusammen einen eindeutigen Zugriff zu einer bestimmten Datei ermöglichen. Die in **runden Klammern** stehende Bezeichnung kann unter bestimmten Bedingungen entfallen. Blanks sind nur wegen der größeren Übersicht eingestreut und dürfen **nicht** mit eingegeben werden.

```
PRINT§ la,  "I   laufwerk"                        Initialisieren
PRINT§ la,  "V   laufwerk"                        Bereinigen
PRINT§ la,  "D   ziellaufwerk = quellaufwerk"     Duplizieren
PRINT§ la,  "N   name (,disk-identität)"          Formatieren / Löschen
PRINT§ la,  "C   name = name (,name,...)"         Kopieren / Zusammenhängen
PRINT§ la,  "R   name = name"                     Umbenennen
PRINT§ la,  "S   name (,name,...)"                Löschen

PRINT§ la,  "P   sa rl rh bp"                     Record auswählen
```

2.4 Disk-BASIC-Kommandos

2.4.1 Übersicht

HEADER	Diskette löschen
BACKUP	Diskette duplizieren
DIRECTORY	Inhaltsverzeichnis auf Bildschirm oder Drucker anzeigen
CATALOG	wie DIRECTORY
COLLECT	Diskettenbelegung in Ordnung bringen
COPY	Datei(en) kopieren
CONCAT	Dateien zusammenhängen
RENAME	Datei umbenennen
SCRATCH	Datei(en) löschen
DOPEN	Datenkanal zum Floppy öffnen
APPEND	Datei ab Ende weiterbeschreiben
DCLOSE	Datenkanal schließen
RECORD	Auf einen Record positionieren

2.4.2 Formatbeschreibung

Die Disk-BASIC-Kommandos unterscheiden sich in der Art der Parameterangabe von den anderen BASIC-Kommandos. Bei den normalen BASIC-Kommandos werden die Parameter stellungsabhängig angegeben. Dies hat den Vorteil, daß keine weitere Kennzeichnung erforderlich ist, aber den Nachteil, daß erstens die Reihenfolge genau eingehalten werden muß und zweitens nur nachfolgende Parameter weggelassen werden können. Bei den Disk-BASIC-Kommandos werden alle Parameter mit Ausnahme des Dateinamens durch Schlüsselzeichen eingeleitet, wodurch die Stellung (Reihenfolge) unwichtig ist.

Bezeichnung	Zeichen	Bedeutung	Ersatzparameter
Logische Adresse LA	§	Nummer	nicht möglich
Gerätenummer GN	U oder ON U	Unit	Gerät Nr. 8
Laufwerk LW	D	Drive	Laufwerk 0
Recordlänge	L	Length	nicht möglich
ID	I	ID	keine ID

Außer der ID können alle Parameter als beliebige numerische Ausdrücke angegeben werden. Diese Ausdrücke müssen hinter dem Schlüsselzeichen **in Klammern** stehen. Das gleiche gilt für Variablen (Sonderfall eines numerischen Ausdrucks). Anders ausgedrückt muß alles in Klammern stehen, was nicht eine einfache Zahl ist (Zahlenkonstante).

2.4.3 Logische Adresse

Die logische Adresse steht hinter dem Zeichen '§' (Doppelkreuz). Das entspricht auch den üblichen Konventionen, da bei PRINT§, INPUT§ und GET§ die logische Adresse auch durch ein führendes '§' angegeben wird.

Bei den Kommandos 'DOPEN', 'APPEND' und 'DCLOSE' ist dieser Parameter stellungsunabhängig.

Der Wertebereich von LA reicht zwar von 0 bis 255, sie sollten aber bei BASIC-4 in Zusammenhang mit Floppy-Schreib-Dateien grundsätzlich nur Werte kleiner als 128 verwenden! Bei PRINT im Rechnerhandbuch ist beschrieben, daß sonst an jedes implizit gesendete CR ein LF angehängt wird, was zu Fehlern beim Lesen der Daten führt.

Beispiele:

```
DOPEN §1,"DATEINAME"          Öffne Datei mit Logischer Adresse 1

LA=2
DOPEN §(LA),"DATEINAME"       Öffne Datei mit Logischer Adresse 2

LA=1
DOPEN §(LA+1),"DATEINAME"     Öffne Datei mit Logischer Adresse 2
```

Falsch

```
LA=2
DOPEN §LA,"DATEINAME"         Die Variable LA muß in Klammern stehen
```

2.4.4 Gerätenummer (unit)

Die Gerätenummer steht hinter der Zeichenfolge 'ON U', 'ONU' oder dem Zeichen 'U'. Bei allen Disk-BASIC-Befehlen ist die Angabe der Gerätenummer optional.

Als **Ersatzparameter** wird die Gerätenummer 8 genommen.

Die Gerätenummer kann Werte zwischen 4 und 31 annehmen. Liegt der Wert nicht in diesem Bereich, meldet der Rechner 'ILLEGAL QUANTITY ERROR'.

Der Rechner meldet keinen Fehler, wenn mehrere Parameter 'U' in einem Befehl angegeben werden. Er nimmt dann den als letzten stehenden. Beachten Sie bitte, daß es kein Floppy-Kommando gibt, das zwei verschiedene Geräte behandeln kann! Deshalb ist die Angabe von mehr als einer Gerätenummer sowieso sinnlos.

Beispiele:

```
DIRECTORY U9          zeige das Inhaltsverzeichnis des Geräts Nr. 9
DIRECTORY             zeige das Inhaltsverzeichnis des Geräts Nr. 8

GN=8
DIRECTORY U(GN)       zeige das Inhaltsverzeichnis des Geräts GN (8)
```

2.4.5 Laufwerknummer

Die Laufwerksnummer muß hinter dem Zeichen 'D' (drive) stehen. Für sie sind die Werte 0 und 1 zulässig. Werden andere Werte angegeben, meldet der Rechner ILLEGAL QUANTITY ERROR.

Für Ersatzparameter gibt es folgende Möglichkeiten:

(1) Wird in einem Befehl nur 1 Dateiname benötigt, ergänzt der Rechner '0'.

(2) Werden in einem Befehl mehrere Dateinamen verwendet, ist der Ersatzparameter für den ersten Namen '0'.

(3) Werden in einem Befehl mehr als ein Dateiname verwendet, und die Laufwerksnummer für den ersten Dateinamen angegeben, ist der Ersatzparameter für die weiteren Datennamen gleich der letzten angegeben Nummer.

Bitte beachten Sie diesen Unterschied der Disk-BASIC-Befehle zu den normalen BASIC-Befehlen, bzw. zum Kommandostring: Auch wenn der Disk-BASIC-Befehl keine Laufwerksangabe enthält, wird dem Floppy ein bestimmtes Laufwerk vorgeschrieben. Dagegen wäre es bei lesenden Zugriffen durchaus möglich, dem Floppy nicht zu sagen, wo die Datei zu finden ist. Das DOS sucht dann automatisch auf dem anderen Laufwerk, falls es auf dem ersten die gewünschte Datei nicht findet. Dieser Service des DOS wird also bei Disk-BASIC-Befehlen grundsätzlich verhindert!

Beispiele:

DLOAD"PROG",D0	Lade Programm "PROG" vom Laufwerk 0
DLOAD"PROG"	"
DLOADD0,"PROG"	"
DLOAD"PROG",D1	Lade Programm "PROG" von Laufwerk 1
COPY "DAT1" TO "DAT2"	Kopiere "DAT1" auf Laufwerk 0 nach "DAT2" auf Laufwerk 0
COPY "DAT1",D1 TO "DAT2"	Kopiere "DAT1" auf Laufwerk 1 nach "DAT2" ebenfalls auf Laufwerk 1

Falsch:

DLOAD"PROG"D0	Zwischen den Parametern muß ein Komma stehen

2.4.6 Recordlänge

Die Recordlänge einer REL-Datei wird hinter dem Zeichen 'L' angegeben. Sie kann als Konstante oder als numerischer Ausdruck in Klammern angegeben werden.

Zugelassen sind die Werte 2 bis 254. Andere Werte bewirken einen 'ILLEGAL QUANTITY ERROR'.

Ersatzwert ist 1.

2.4.7 Dateiname und Diskname

Dateiname und Diskname benötigen keine Schlüsselzeichen, da sie als Strings schon eindeutig sind. Möglich sind String-**Konstanten** in **Anführungszeichen** '"' oder String-**Variablen** bzw. String-**Ausdrücke,** die in **Klammern** stehen müssen.

Der Dateiname und der Diskname darf **maximal 16 Zeichen** lang sein und muß **mindestens 1 Zeichen** lang sein. Ist er länger als 16 Zeichen, meldet der Rechner STRING TOO LONG ERROR, ist er kürzer als ein Zeichen, meldet er ILLEGAL QUANTITY ERROR.

Beispiele:

DLOAD"PROG"	Lade "PROG" von Laufwerk 0
NA$="PROGRAMMNAME" DLOAD(NA$)	Lade "PROGRAMMNAME" von Laufwerk 0
NA$="PROGRAMMNAME" DLOAD(NA$+"1")	Lade "PROGRAMMNAME1" von Laufwerk 0

Falsch

DLOAD"programmname-nummer1"	Der Dateiname ist länger als 16 Zeichen
NA$="" DLOAD(NA$)	Der Dateiname ist leer
DLOAD NA$	Variablen müssen in Klammern stehen

2.4.8 Diskettenidentität ID

Die ID wird durch das führende Zeichen 'I' gekennzeichnet. Sie muß aus zwei Zeichen bestehen. Für die ID sind nur zwei Zeichen erlaubt, die hinter dem Zeichen 'I' ohne Blank stehen müssen.

Bei mehr oder weniger Zeichen meldet der Rechner einen 'SYNTAX ERROR'.

Bei der ID sind **keine Ausdrücke (Variablen)** möglich.

Beispiele:

HEADER "DISKNAME",D1,I01	Formatiere die Disk im Laufwerk 1 mit der ID '01'.

Falsch:

HEADER "DISKNAME",D1,IA	Die ID besteht hier nur aus dem einen Zeichen 'A'.
HEADER "DISKNAME",D1,I(A$)	keine Variable erlaubt

2.4.9 Trennzeichen zwischen den Parametern

Die Parameter werden normalerweise durch **Komma** getrennt. Eine Ausnahme bildet der Parameter für die Gerätenummer 'U', wenn er in der Form 'ON U' angegeben wird. In diesem Fall **darf kein Komma** davor stehen.

Wird ein Komma eingegeben, so muß auch ein Parameter folgen.

Zwischen dem Befehl und dem ersten Parameter darf kein Trennzeichen stehen.

Beispiele:

DLOAD"PROG",D1,U8	Lade "PROG" von Gerät 8, Laufwerk 1
DLOAD"PROG",D1 ONU8	"

Falsch:

DLOAD,"PROG",D1	Zwischen Befehl und Parameter darf kein Komma stehen
DLOAD"PROG",D1,ONU8	Vor 'ON' darf kein Komma stehen

Bei Befehlen, die mehrere Dateinamen benötigen, wird die Richtung der Datenübertragung durch 'TO' angegeben. Dasselbe gilt bei Diskettenbefehlen, die ein Quell-Laufwerk und ein Ziel-Laufwerk angeben.

Beachten Sie hier folgenden Unterschied zwischen Disk-BASIC- und Kommandostring -Format:

Disk-BASIC-Format:	**Quelle**	**TO**	**Ziel**
Kommandostring-Format:	**Ziel**	=	**Quelle**

Während also bei Disk-BASIC links vom Trenn'zeichen' TO die Quelle und rechts das Ziel steht, ist es beim Kommandostringformat genau umgekehrt: Links von '=' steht das Ziel, rechts die Quelle.

Beispiele:

COPY D1,(NA$) TO D0,(NA$)	Kopiere Datei mit Namen NA$ von Laufwerk 1 nach Laufwerk 0
BACKUP D1 TO D0	Dupliziere Diskette von Laufwerk 1 nach 0

Falsch:

BACKUP D1,TO D0	Vor und hinter 'TO' darf kein Komma stehen

2.4.10 Leerzeichen (Blanks)

Zwischen Befehl und Parameter und zwischen den Paramatern selbst können beliebig viele Blanks (begrenzt durch die Zeilenlänge) stehen. Dasselbe gilt zwischen den Schlüsselzeichen und den Angaben.

Eine Ausnahme ist die ID. Genau die zwei Zeichen, die hinter dem 'I' stehen, werden als ID genommen.

2.5 Allgemeine Parameterbeschreibung

2.5.1 Einführung

In diesem Kapitel werden alle Parameter beschrieben, die für mehrere Befehle gebraucht werden. Bei den einzelnen Befehlen werden diese Parameter dann nicht nocheinmal beschrieben, sondern es wird vorausgesetzt, daß Sie dieses Kapitel gelesen haben. Bitte beachten Sie auch die ergänzenden Hinweise in den beiden vorausgegangenen Kapiteln 'Kommandos über den Kommandokanal senden' und 'Disk-BASIC-Kommandos'.

2.5.2 Logische Adresse (la)

Durch die Logische Adresse oder Logische Dateinummer wird bei jeder Ein- oder Ausgabeoperation ein Gerät mit der Gerätenummer gn und der Sekundäradresse sa angesprochen.

Die Logische Adresse muß bei den Kommandos angegeben werden, die Datenkanäle zum Floppy öffnen (OPEN, DOPEN) oder schließen (CLOSE, DCLOSE) bzw. Ein- (INPUT§, GET§) oder Ausgabeoperationen (PRINT§) über solche Datenkanäle durchführen. Außerdem wird sie bei RECORD benötigt.

Einzelheiten sind im Rechnerhandbuch bei OPEN beschrieben. Bitte beachten Sie, daß Sie für Schreibdateien (Ausgabe) nur la's kleiner als 128 verwenden sollten!

2.5.3 Gerätenummer (gn)

Jedes Peripheriegerät im CBM-System hat eine bestimmte Gerätenummer, die meistens in bestimmten Grenzen durch den Commodore-Service im Gerät eingestellt werden kann.

Die Standardeinstellung für das Floppy ist **Gerätenummer 8.** Durch eine Änderung innerhalb des Gehäuses kann die Adresse zwischen **8** und **15** variiert werden. Dies ist erforderlich, wenn mehrere Doppellaufwerke am Rechner angeschlossen werden sollen.

Generell sind also in Zusammenhang mit dem Floppy nur Gerätenummern zwischen 8 und 15 sinnvoll.

Die Gerätenummer wird auch Primäradresse, physikalische Adresse oder Geräteadresse genannt.

2.5.4 Sekundäradresse / Kanalnummer (sa)

Die Sekundäradresse wird vom Floppy unbedingt benötigt und muß bei OPEN angegeben werden. DOPEN verwaltet dagegen die SA automatisch, so daß Sie bei Verwendung von DOPEN zum nächsten Parameter übergehen können.

Die Sekundäradresse ist die Nummer eines **Kanals** zum / vom Floppy. Insgesamt sind für SA Werte zwischen **0** und **15** möglich. Davon haben drei Werte (Kanäle) spezielle Bedeutungen:

0 LOAD (Lesen einer PRG-Datei)
1 SAVE (Schreiben einer PRG-Datei)
15 Kommando- / Fehlerkanal

Die verbleibenden Adressen (**2 - 14**) richten **Daten**kanäle für Dateien ein.

Hier soll nicht weiter auf den Begriff des Kanals eingegangen werden. Sein Verständnis ist erst in Zusammenhang mit den Direktzugriffsoperationen erforderlich. Es genügt zu wissen, daß die Kanalnummer nur logische (keine physikalische) Bedeutung hat. Sie erfüllt fürs Floppy etwa die gleiche Funktion, wie die LA für den Rechner.

Entsprechend darf die gleiche Kanalnummer (SA) nur **einmal gleichzeitig** verwendet werden.

Von den 16 möglichen Kanälen dürfen **nur 5 gleichzeitig** geöffnet sein! Andernfalls meldet das Floppy '70 NO CHANNEL'.

Über diese maximal 5 Kanäle (10 Puffer) wird der Datenverkehr in beiden Richtungen abgewickelt. Dies gilt sowohl für den seriellen, als auch für den direkten Zugriff.

Wenn auch REL-Dateien geöffnet sind, erniedrigt sich diese Zahl. Deshalb muß bei jedem Öffnen eines Kanals (einer Floppy-Datei) die Floppy-Fehlermeldung abgefragt werden.

Etwas detaillierter sehen die Zusammenhänge wie folgt aus: Zu (logischen) Kanälen gehören (physikalische) Puffer im Floppy. Insgesamt stehen für Dateien (Kanäle) 10 Puffer zur Verfügung. Jede SEQ oder PRG Datei (Kanal) benötigt 2 Puffer, jede REL Datei dagegen 3 Puffer. Deshalb können maximal **5 SEQ/PRG** Dateien oder **3 REL** Dateien gleichzeitig offen sein. Mischungen wie 2 SEQ mit 2 REL Dateien sind ebenfalls möglich.

Wichtig ist die SA **15.** Sie bewirkt, daß der angegebenen logischen Adresse der sogenannte Kommando- und Fehlerkanal zugeordnet wird. Bei 'Kommandos über den Kommandokanal senden' und 'Fehlerbehandlung' finden Sie darüber weitere Einzelheiten. Der Kommandokanal hat nichts mit den Datenkanälen zu tun, es hat also auf mögliche Anzahl Datenkanäle keinen Einfluß, ob der Kommandokanal geöffnet ist oder nicht.

2.5.5 Laufwerknummer

Da das Floppy zwei Laufwerke hat, muß man ihm mitteilen, auf welchem die Operation durchgeführt werden soll.

Die Laufwerknummer **0** spricht das **rechte** Laufwerk an und entsprechend **1** das **linke** Laufwerk. Die Laufwerke sind durch Aufschriften 'drive 0' und 'drive 1' beschriftet.

2.5.6 Dateiname

Mit Ausnahme von Kommandos, die eine ganze Diskette betreffen, muß immer mindestens 1 Dateiname angegeben werden.

Der Dateiname ist ein String, der **maximal** 16 Zeichen lang sein darf und **mindestens** ein Zeichen lang sein muß. Er darf mit folgenden Ausnahmen alle druckbaren Zeichen enthalten:

```
* ? = , : ;
```

Diese Zeichen haben innerhalb des Kommandostrings folgende Bedeutung:

= , ; und : sind Trennzeichen im Kommandostring.

(Der Kommandostring wird bei Verwendung der Disk-BASIC-Kommandos nur vom Rechner intern verwendet.)

2.5.7 Joker-Zeichen im Dateinamen

'?' und '*' sind "Joker"-Zeichen. Sie können verwendet werden, um Dateien zu finden, die einem bestimmten Namensmuster entsprechen.

Dies gilt bei den Funktionen:

(1) Öffnen eines Datenkanals zum **Lesen** einer Datei (nicht zum Schreiben!)
(2) Laden und verifizieren eines **Programms** (nicht abspeichern!)
(3) Bestimmte Dateinamen im **Inhaltsverzeichnis** anzeigen
(4) Mehrere Dateien mit einem Kommando **löschen**

Dabei sind zwei Fälle zu unterscheiden:

- Bei den Funktionen (1) und (2) wird die **erste** Datei im Inhaltsverzeichnis genommen, die dem Namensmuster entspricht.

- Bei den Funktionen (3) und (4) werden **alle** Dateien der Diskette genommen, die dem Namensmuster entsprechen.

'?' kann an jeder Stelle des Namens für genau **1 Zeichen** stehen.

'*' steht am **Ende** des Namens für eine **beliebige Anzahl** von Zeichen.

Beispiele

Angegebenes Muster	**zutreffende Dateinamen**	**nicht zutreffende Dateinamen**
1234567890123456	1234567890123456	1234567890123456
"DATEN*"	"DATEN"	"DATE"
	"DATEN.1"	"DATUM"
	"DATEN.VIER"	".DATEN"
"DA?EN?"	"DATEN."	"DATEN"
	"DADENS"	"DADENS."
"A???*"	"ABC1"	"BCDE"
	"ABCDE.+"	"BCDEF"
	"A "	"A"

2.5.8 Diskettenname

Der Diskettenname hat nur (für den Anwender) den Zweck, Disketten durch Namen leicht unterscheiden zu können. Für den Diskettennamen gelten die Regeln des Dateinamens sinngemäß.

Für das DOS hat der Diskettenname keinerlei Bedeutung, er wird lediglich in jedem Inhaltsverzeichnis mit ausgegeben.

2.5.9 Diskettenidentität (ID)

Die ID ist eine zusätzliche Kennzeichnung einer Diskette. Sie hat für das Floppy 8050 keine Bedeutung für die Identifizierung der Diskette.

Beim Formatieren einer Diskette gibt die ID dem Floppy an, ob die Diskette vollständig neu formatiert wird, oder ob nur der Inhalt des Inhaltsverzeichnisses gelöscht werden soll. (Siehe HEADER-Befehl)

Die ID kann aus zwei beliebigen Zeichen bestehen.

Für Benutzer, die schon mit 3040 Floppy-Versionen gearbeitet haben, soll hier noch kurz erklärt werden, warum die ID keine Bedeutung mehr hat.

Die 3040/4040 Floppys können einen Diskwechsel nur an einem ID-Wechsel feststellen. Dagegen kann das 8050 einen Diskwechsel daran erkennen, daß eine Diskette aus dem Laufwerk genommen wird. Das hat auch zur Folge, daß alle offenen Dateien auf diesem Laufwerk geschlossen werden, wenn die Diskette gewechselt wird.

V. DIE KOMMANDOS

1. Übersicht

Die Kommandos sind grob gegliedert in solche, die

- die **ganze Diskette** betreffen
- **einzelne Dateien** betreffen
- **Programme** abspeichern / laden
- den **Datenverkehr** abwickeln
- die **Fehlermeldung** lesen

Bitte lesen Sie in diesem Zusammenhang auch die entsprechenden Teile des Rechnerhandbuches nocheinmal durch, falls Sie nicht ganz sattelfest bei den Ein-/Ausgabebefehlen sind.

Das Format der Kommando-Beschreibungen:

Bei 'Übermittlungsmöglichkeiten' wurde erklärt, daß es bei BASIC-4 für die meisten Floppy-Kommandos zwei Möglichkeiten der Darstellung innerhalb von BASIC gibt bzw. daß bei BASIC-3 nur die primitivere Möglichkeit existiert.

Bei jedem Kommando haben wir deshalb die Formate für beide Arten der Übermittlung beschrieben. Da Sie in der Regel entweder mit der einen oder der anderen Möglichkeit arbeiten, haben wir uns streng an das nachfolgend beschriebene Format gehalten. Dadurch können Sie immer sofort die gewünschte Information finden, ohne die andere ebenfalls lesen zu müssen.

Die Beschreibung eines Kommandos teilt sich jeweils in folgende Blöcke auf:

(1) Beschreibung (Zweck) mit allgemeinen Anmerkungen, die für beide Übermittlungsmöglichkeiten gelten.

(2) Kommando als Disk-BASIC-Befehl mit Formatbeschreibung

(3) Floppy-Kommandostring mit Formatbeschreibung
oder allgemeingültiger BASIC-Befehl (z. B. LOAD)

Bei der Formatbeschreibung fett gedruckte Teile müssen immer angegeben werden, nicht fett gedruckte können unter bestimmten Bedingungen wegfallen. Beachten Sie in diesen Fällen die angegebenen Ersatzwerte.

2. Disketten - Kommandos

2.1 Diskette löschen oder neu formatieren: HEADER / New

Jede Diskette, die noch nicht mit einem Floppy 8050 in Gebrauch war, muß mit diesem Kommando formatiert werden.

Disketten, die schon auf einem 8050-Laufwerk benutzt wurden, können gelöscht werden, ohne neu formatiert zu werden. Dann wird nur der Inhalt des Inhaltsverzeichnisses gelöscht. Natürlich können auch solche Disketten neu formatiert werden.

Das 8050 Floppy arbeitet mit 'soft-sectored' Disketten. D.h. die zum Betrieb notwendige Einteilung in Spuren und Sektoren wird erst durch die Formatierung auf die Diskette gebracht. Diese Einteilung besteht aus einem Vor- und einem Nachspann zu jedem einzelnen Block (Sektor). Dieser Rahmen wird auch als 'header' = 'Kopf' bezeichnet. Er enthält für den Floppy-Prozessor alle Angaben, die er benötigt, um jeden Block eindeutig identifizieren zu können.

Nach dem Formatieren ist die Diskette für den Benutzer gewissermaßen 'hard - sectored', da er an der Blockeinteilung und Länge genausowenig ändern kann wie bei einer echten 'hard - sectored' Diskette.

Zusätzlich zu dieser Blockeinteilung werden noch drei Informationen auf die 'frische' Diskette gebracht:

(1) Diskettenname

Dieser wird immer neu zugewiesen, wie er im Kommando angegeben ist. Der Diskettenname muß immer im Kommando stehen.

(2) Disketten-Identität ID

Wird im Kommando die ID angegeben, dann wird die Diskette neu **formatiert** und bekommt dabei auch eine neue ID zugewiesen.

Wird sie nicht angegeben, wird **nur das Inhaltverzeichnis gelöscht** und die ID beibehalten.

(3) Block-Verfügbarkeits-Tabelle BAM (**B**lock-**A**vailability-**M**ap)

Nach dem Formatieren oder Löschen werden immer alle Blöcke der Diskette freigegeben. Sie hat dann 2052 freie Blöcke.

Das Formatieren (mit Angabe der ID) benötigt ca. 180 sec, das Löschen (ohne ID) ca. 5 sec.

Wichtiger Hinweis

Die Folgen eines unbeabsichtigten Formatierkommandos sind unangenehm. Beim Kommando mit Angabe der ID gibt es keine Möglichkeit, den Inhalt der Diskette zurückzugewinnen.

Wer sofort bemerkt, daß der das Kommando doch nicht ausgeführt haben will, kann sofort die Klappe des betreffenden Laufwerks ausrasten. Falls der Schreib-/Lesekopf gerade dabei war, zur Spur 1 zu 'rattern', ist damit die Diskette gerettet. Dem Laufwerk wird dadurch kein Schaden zugefügt.

2.1.1 HEADER (BASIC-4)

Format:

HEADER "diskname" , D lw , U gn , I ii

Parameter:

diskname	Diskettenname
lw	Laufwerksnummer
gn	Gerätenummer
ii	ID, keine Variablen möglich

Anmerkungen:

Bei diesem Befehl muß die Laufwerksnummer unbedingt angegeben werden.

Wird dieser Befehl im Direkt-Modus eingegeben, fragt der Rechner nach, ob man es wirklich ernst meint:

ARE YOU SURE?

Das Kommando wird nur dann an das Floppy geschickt, wenn man hier 'YES' oder 'Y' eingibt.

Während der Ausführung wird vom Rechner der Disk-Status abgefragt. Er hängt also in der Abfrage und kann währenddessen nichts anderes machen.

Wenn das Floppy einen Fehler meldet, gibt der Rechner im Direkt-Modus die Fehlermeldung 'BAD DISK' aus. Die Floppy-Fehlermeldung kann aus den Variablen DS$ bzw. DS gelesen werden.

Steht der HEADER-Befehl im BASIC-Programm, wird die Fehlermeldung 'BAD DISK' nicht ausgegeben. Das Programm muß selbst den Floppy-Status aus DS$ bzw. DS lesen und verarbeiten.

HEADER schließt im Rechner und im Floppy alle Dateien des entsprechenden Floppys, auch den Kommandokanal!

Beispiele:

```
HEADER "DISKETTE-NR-1",D1          Lösche Diskette in Laufwerk 1

NA$="DISKETTE-NR-2":LW=0           Variablen für Diskname und Laufwerk
HEADER (NA$),D(LW)                 Löschen der Diskette

HEADER "DISKNAME1",D1,I01          Formatiere die Diskette in Laufw. 1
```

Falsch:

```
LW=1
HEADER "DISKNAME1",D LW            Variablen müssen in Klammern stehen

ID="02"
HEADER "DISKNAME",D1,I(ID)         Die ID kann keine Variable sein

NA$="DISKETTENNAME-NUMMER1"
HEADER (NA$),D1                    Diskname darf max. 16 Zeichen lang sein
```

2.1.2 New (Kommandokanal)

Format:

N **laufwerknummer : diskettenname** , ID

Beispiele (nach OPEN 15,8,15):

PRINT§15,"N0:TEXT.1,T1"

Die Diskette im Laufwerk 0 erhält den Namen 'TEXT.1' und die ID 'T1'. Durch Angabe der ID wird die Diskette formatiert. (s.o.!)

PRINT§15,"N1:TEXT.3

Die Diskette im Laufwerk 1 erhält den Namen 'TEXT.3' und behält ihre alte ID. Weil keine ID angegeben wurde, wird die Diskette nicht neu formatiert, sondern nur gelöscht.

2.2 Diskette duplizieren: BACKUP / Duplicate

Beim Duplizieren wird der gesamte Inhalt einer Diskette auf eine zweite Diskette kopiert:

Jeder Block (Sektor) wird auf der Zieldiskette an der gleichen Stelle (Spur, Sektor) abgelegt, von der er von der Quelldiskette genommen wurde.

Dadurch erhält man zwei Disketten mit identischem Inhalt. Insbesondere werden auch alle Direktzugriff-Sektoren, sowie die ID und der Diskettenname kopiert.

Beide Disketten müssen in gutem Zustand sein, da jeder Schreib- oder Lesefehler diese Operation vorzeitig beendet!

Das Duplizieren der Disketten funktioniert nur innerhalb eines Gerätes.

Wichtiger Hinweis

Da eine Verwechslung von Quell- und Zieldiskette zur Folge hat, daß die Diskette, deren Inhalt man eigentlich kopieren wollte, gelöscht wird, ist es ratsam, bei der **Quell**diskette den **Schreibschutz** zu 'kleben'.

2.2.1 BACKUP (BASIC-4)

Format:

BACKUP D lw1 TO D lw2 , U gn

Parameter:

lw1	Laufwerk, von dem runterkopiert wird	(Quellaufwerk)
lw2	Laufwerk, auf das dupliziert wird	(Ziellaufwerk)
gn	Gerätenummer	

Die Diskette in Laufwerk lw1 wird auf die Diskette in Laufwerk lw2 dupliziert.

Gibt man zweimal dasselbe Laufwerk an, findet das Floppy einen '30 SYNTAX ERROR'. BASIC bringt keine Fehlermeldung.

BACKUP fragt nicht automatisch den Floppy-Status ab. Der Rechner kommt also nach dem Senden des Kommandos sofort zurück.

BACKUP schließt alle offenen Dateien auf dieses Gerät, führt also automatisch ein DCLOSE durch. Dies hat folgende Wirkungen:

(1) Der (eventuell offene) Kommandokanal wird geschlossen.
(2) Lesedateien werden geschlossen.
(3) Schreibdateien werden ordnungsgemäß geschlossen, sind also gültige Dateien (kein Stern im Inhaltsverzeichnis).

Beispiele:

```
BACKUP D0 TO D1              Dupliziert Laufwerk 0 nach Laufwerk 1
BACKUP D1 TO D0              Dupliziert Laufwerk 1 nach Laufwerk 0

QL=1                         Dupliziert Laufwerk, das in QL angegeben
BACKUP D(QL) TO D(1-QL)      ist, auf das andere Laufwerk
```

Falsch:

```
BACKUP D1 TO D1              zweimal dasselbe Laufwerk

QL=1
BACKUP DQL TO D(1-QL)        Variablen müssen in Klammern stehen
```

2.2.2 Duplicate (Kommandostring)

Format:

D ziellaufwerk = quellaufwerk

Beispiele (nach OPEN 15,8,15):

```
PRINT§15 , "D0=1"     Kopie von 1 wird auf 0 gebracht

PRINT§15 , "D1=0"     Kopie von 0 wird auf 1 gebracht
```

2.3 Diskettenbelegung überprüfen und bereinigen: COLLECT / Validate

Dieses Kommando löscht alle Blöcke von der Diskette, die nicht durch serielle Dateien (SEQ, PRG, REL) im Directory belegt sind. Insbesondere werden alle Blöcke in der BAM freigegeben, die im Direktzugriff belegt wurden.

Nach 'SCRATCH' - Operationen kann es vorkommen, daß nicht alle Blöcke freigegeben werden, die eigentlich frei sein müßten. Auch solche Blöcke werden freigegeben.

Da nach dem Überprüfen der Diskette die neue BAM geschrieben wird und evtl. Änderungen im Inhaltsverzeichnis vorgenommen werden, darf der **Schreibschutz** nicht 'geklebt' sein.

Wenn während der Bearbeitung ein Lesefehler auftritt, wird die Operation abgebrochen und die Diskette bleibt unverändert, weil die neue BAM nicht auf die Diskette geschrieben wird.

Dateien, die mit einem '*' im **Inhaltsverzeichnis** gekennzeichnet sind, werden aus dem Inhaltsverzeichnis **gelöscht** und Ihre Blöcke in der BAM **freigegeben.** Dies gilt nicht für Dateien, zu denen noch ein offener Kanal existiert, die also gerade zum Schreiben geöffnet sind.

2.3.1 COLLECT (BASIC-4)

Format:

COLLECT D lw , U gn

Parameter:

lw	Laufwerksnummer
gn	Gerätenummer

Wird die Laufwerksnummer nicht angegeben, wird das zuletzt benutzte Laufwerk vom Floppy ausgewählt.

COLLECT schließt alle offenen Dateien auf dieses Gerät, beinhaltet also die Funktion von DCLOSE. Dies hat folgende Wirkungen:

(1) Der eventuell offene Kommandokanal wird geschlossen.
(2) Lesedateien werden geschlossen.
(3) Schreibdateien werden ordnungsgemäß geschlossen und sind wieder lesbar (haben keinen Stern im Inhaltsverzeichnis).

Beispiele:

COLLECT D1	Bereinige Disk in Laufwerk 1
COLLECT D0	Bereinige Disk in Laufwerk 0
COLLECT D0,U9	Bereinige Disk in Laufwerk 0 auf Gerät 9

2.3.2 Validate (Kommandostring)

Format:

V laufwerknummer

Beispiele (nach OPEN 15,8,15):

PRINT§15,"VO" Diskette im Laufwerk 0 überprüfen

PRINT§15,"V1" Diskette im Laufwerk 1 überprüfen

2.4 Inhaltsverzeichnis lesen: DIRECTORY, CATALOG / $

Mit diesem Kommando kann man das Inhaltsverzeichnis der Diskette lesen und auf dem Bildschirm oder auf dem Drucker ausgeben. Diese Ausgabe ermöglicht also nur das Drucken einer Liste, der Inhalt des Inhaltsverzeichnisses kann nicht von BASIC-Programmen ausgewertet werden.

Format des Inhaltsverzeichnisses auf dem Bildschirm:

Die erste Zeile enthält die Laufwerknummer, den Diskettennamen, die ID und die DOS-Version.

Die weiteren Zeilen enthalten jeweils die Anzahl der Blöcke, die die Datei auf der Diskette belegt, den Dateinamen und den Typ.

In der letzten Zeile wird gemeldet, wieviele Blöcke noch frei sind.

2.4.1 DIRECTORY / CATALOG (BASIC-4)

Die Befehle DIRECTORY und CATALOG wirken genau gleich und haben auch dasselbe Format.

Format:

DIRECTORY D lw , U gn
CATALOG D lw , U gn

Parameter:

lw	Laufwerknummer
gn	Gerätenummer

DIRECTORY bringt das Inhaltsverzeichnis der Diskette auf den Bildschirm. Durch Umlenken des Standard-Ausgabekanals mit CMD kann es auch auf den Drucker ausgegeben werden.

Mit DIRECTORY kann man nicht Dateinamen mit einem bestimmten Namensmuster auswählen. Dieses Kommando bringt immer das gesamte Inhaltsverzeichnis.

Das Lesen nach einem bestimmten Namensmuster finden Sie bei 'Laden des Inhaltsverzeichnisses'.

DIRECTORY öffnet zum Lesen des Inhaltsverzeichnisses eine Datei mit la=14 und sa=0. Sind zu diesem Zeitpunkt bereits 10 Dateien offen, wird TOO MANY FILES ERROR gemeldet. Wenn weniger als 10 Dateien geöffnet sind und la=14 bereits einmal verwendet wurde, entsteht dadurch kein Fehler. Nach Beenden des Kommandos wird die zweite la=14 sofort wieder geschlossen.

Beispiele:

DIRECTORY D0	Drucke Inhaltsverzeichnis des Laufwerks 0 auf Bildschirm
OPEN4,4 CMD4 DIRECTORY	Drucke beide Inhaltsverzeichnisse auf den Drucker

2.4.2 Laden des Inhaltsverzeichnisses: LOAD"$...

Durch LOAD wird das Inhaltsverzeichnis in den Speicher geladen. Da dadurch jedes BASIC-Programm überschrieben wird, kann der Inhalt des Inhaltsverzeichnisses über diesen Zugriff nicht von BASIC-Programmen ausgewertet werden. Dieses Inhaltsverzeichnis kann mit 'LIST' auf den Bildschirm oder Drucker ausgegeben werden.

Der Vorteil gegenüber DIRECTORY liegt darin, daß Sie Jokerzeichen verwenden können und dadurch eine überschaubarere Liste der Dateinamen erhalten.

Format:

LOAD"$lw:name**",gn**

Parameter:

lw	Laufwerksnummer
gn	Gerätenummer
name	Namensmuster

wobei **name** nach folgendem Muster angegeben wird:

name = **dateiname** =Typ

Für **Typ** gibt es die Möglichkeiten:

S	SEQ-Dateien
P	PRG-Dateien
R	REL-Dateien
U	USR-Dateien

Beispiele:

LOAD"$0",8	Lade Inhaltsverz. von Laufwerk 0
LOAD"$0:DAT*",8	wie vor, aber nur alle Dateien mit 'DAT' laden
LOAD"$0:DAT*=P",8	wie vor, aber nur alle PRG-Dateien laden

2.5 Diskette initialisieren: Initialize

In der Regel ist beim 8050 die Initialisierung nicht nötig. Es gibt aber einen Bedienerfehler, den man durch explizites Initialisieren korrigieren kann: Wenn bei einem Laufwerk beim Einschalten bereits eine Diskette im Schlitten eingerastet ist und diese Diskette dann bearbeitet werden soll, erfolgt ein 29 DISK ID MISMATCH Fehler.

Entweder kann man nun durch

```
PRINT§15, "I0"
PRINT§15, "I1"
```

die Diskette initialisieren oder man muß den Bediener auffordern, die entsprechende Diskette kurz aus dem Schlitten auszurasten und wieder hineinzuschieben. Dadurch wird die Diskette vom DOS automatisch initialisiert.

Für Initialisieren existiert kein Disk-BASIC-Befehl, weil es normalerweise nicht nötig ist.

Was bedeutet initialisieren?

Auf der BAM ist verzeichnet, welche Blöcke der Diskette frei und welche belegt sind. Die BAM wird nach dem Einlegen einer Diskette beim ersten Zugriff von der Diskette in den DOS-Arbeitsspeicher gelesen. Das DOS muß sich darauf verlassen können, daß die BAM im Arbeitsspeicher auch wirklich zu der Diskette gehört, die gerade im Laufwerk liegt. Um diese Zuordnung zu gewährleisten, überwacht ein Kontakt, ob eine Diskette aus dem Laufwerk entfernt bzw. eingelegt wird. Dieser Kontakt meldet dem DOS, daß eine neue Diskette eingelegt wurde, daß also vor dem nächsten Zugriff die BAM gelesen werden muß.

Nur in dem oben angeführten Sonderfall funktioniert diese Logik nicht.

3. Datei-Kommandos

3.1 Dateien kopieren: COPY / Copy

Mit diesem Kommando können einzelne Dateien (Einzelkopieren) oder alle Dateien einer Diskette (Sammelkopieren) kopiert werden.

Da Kopieren ein schreibender Zugriff ist, darf der Schreibschutz auf der Zieldiskette nicht 'geklebt' sein, sonst wird '26 WRITE PROTECT ON' gemeldet.

Alle Dateitypen können kopiert werden, nicht aber Blöcke, die durch Direktzugriff belegt wurden, die also keine Verbindung zum Directory haben.

Kein Name darf ein Jokerzeichen enthalten.

Einzelkopieren

Beim Einzelkopieren können Quell- und Zieldatei auf der gleichen oder auf verschiedenen Disketten liegen.

Im Kommando müssen die Dateinamen der Quell- und der Zieldatei stehen, auch wenn die Namen identisch sind. Beide Namen dürfen keine Jokerzeichen enthalten. Das Floppy meldet sonst '30 SYNTAX ERROR'.

Existiert die Quelldatei nicht, folgt Fehler 'FILE NOT FOUND'.
Existiert die Zieldatei schon, folgt Fehler 'FILE EXISTS'.

Sammelkopieren

Werden im Kommando keine Dateinamen angegeben, dann werden alle Dateien der Quelldiskette auf die Zieldiskette kopiert. Existiert ein Quell-Dateiname schon auf der Zieldiskette, so wird das Kommando mit der Fehlermeldung '63 FILE EXISTS' abgebrochen.

Beachten Sie die folgenden Unterschiede zwischen Sammelkopieren und BACKUP:

COPY kann **keine Direktzugriff-Sektoren** kopieren, bringt also u.U. nicht die ganze gewünschte Information auf die Zieldiskette.

Die Diskette wird durch COPY unter Umständen reorganisiert: COPY nimmt einen Dateinamen nach dem anderen aus dem Quell-Directory und kopiert die dazugehörigen Sektoren. Sollten die Sektoren auf der Quelldiskette also durch häufige Änderungen 'in alle Winde verstreut' sein, werden sie durch Copy wieder geordnet. Dies kann erwünscht sein, muß es aber nicht!

3.1.2 COPY (BASIC-4)

Format:

COPY D lw1 , **"dateiname1" TO** D lw2, **"dateiname2"** , U gn Einzelkopieren

COPY D lw1 TO D lw2 , U gn Sammelkopieren

Parameter:

lw1	Quell-Laufwerk
lw2	Ziel-Laufwerk
dateiname1	Quell-Dateiname
dateiname2	Ziel-Dateiname
gn	Gerätenummer

lw1 und lw2 können identisch sein.

dateiname1 und dateiname2 können identisch sein, wenn lw1 und lw2 verschieden sind. (Auf einer Diskette können nicht zwei Dateien den gleichen Namen haben.)

gn darf nur einmal angegeben werden, da Kopieren nur innerhalb eines Gerätes möglich ist!

Beispiele:

COPY D1,"DAT1" TO D0,"DAT2"	Kopiere DAT1 auf Laufwerk 1 nach DAT2 auf Laufwerk 0
COPY D0,"DATALT" TO "DATNEU"	Kopiere DATALT nach DATNEU auf demselben Laufwerk
QL=1	Quell-Laufwerk
DN$="DATNAME"	Dateiname
COPY D(QL),(DN$) TO D(1-QL),(DN$)	Kopiere DATNAME von 1 nach 0

Falsch:

COPY D0,"DAT1" TO "DAT1"	Die Zieldatei auf demselben Laufwerk darf nicht denselben Namen haben.
QL=1	Quell-Laufwerk
DN$="DATNAME"	Dateiname
COPY DQL,DN$ TO D1-QL,DN$	Variablen und numerische Ausdrücke müssen in Klammern stehen

3.1.3 Copy (Kommandostring)

Format:

C lw1:zielname = lw2:quellname Einzelkopieren

C lw1 = lw2 Sammelkopieren

wobei:
lw1 = Laufwerknummer der Zieldatei
lw2 = Laufwerknummer der Quelldatei

Beispiele (nach OPEN 15,8,15):

PRINT§15, "C0:DATNEU=1:DATALT"

Eine Kopie der Datei 'DATALT' vom Laufwerk 1 wird unter dem Namen 'DATNEU' auf dem Laufwerk 0 abgelegt.

Wenn DATALT auf 1 nicht existiert, folgt Fehler 62 FILE NOT FOUND.

Wenn DATNEU auf 0 schon existiert, folgt Fehler 63 FILE EXISTS.

PRINT§15, "C0=1"

Alle Dateien von Laufwerk 1 auf Laufwerk 0 kopieren.

3.2 Dateien zusammenhängen: CONCAT / Copy

Mit diesem Kommando können mehrere Dateien des gleichen Typs zu einer einzigen Datei zusammengefaßt werden.

Das bedeutet aber nicht, daß dies eine Möglichkeit ist, BASIC- oder Assemblerprogramme aus mehreren Teilprogrammen zusammensetzen zu können! Wegen des Zeigers am Anfang von PRG-Dateien ist es sinnlos, PRG-Dateien mit diesem Kommando zu verbinden.

REL-Dateien können nicht zusammengehängt werden. Wird es versucht, meldet das Floppy 'FILE TYPE MISMATCH'. Diese Fehlermeldung taucht auch auf, wenn eine PRG mit einer SEQ Datei verknüpft werden soll.

Kein Dateiname darf Jokerzeichen enthalten.

Da der Kommandopuffer des Floppys nur 40 Zeichen fassen kann, ist es nur möglich, Dateien mit Namen, die kürzer als 16 Zeichen sind, zusammenzuhängen. Sind beide Dateinamen gleich lang, ist die maximale Länge 10 Zeichen. Bei längeren Namen meldet das Floppy 'FILE NOT FOUND', da ein Quellname abgeschnitten wird.

Die Zieldatei, also die Datei, an die eine andere angehängt wird, wird dadurch verändert, sie existiert nach dem Kommando in ihrer ursprünglichen Form nicht mehr. Die angehängte Datei existiert dagegen weiterhin zusätzlich in der alten Form.

3.2.1 CONCAT (BASIC-4)

Format:

CONCAT D lw1 , **"dateiname1" TO** D lw2 , **"dateiname2"** , U gn

Die Datei "dateiname1" auf Laufwerk lw1 wird an die Datei "dateiname2" auf Laufwerk lw2 angehängt.

Beispiele:

CONCAT D1,"DAT1" TO D0,"DAT2"	Die Datei DAT1 auf Laufwerk 1 wird an die Datei DAT2 auf Laufwerk 0 angehängt.
CONCAT D0,"DAT1" TO "DAT2"	Die Datei DAT1 auf Laufwerk 0 wird an die Datei DAT2 auf demselben Laufwerk angehängt.

Falsch:

N$="DATEI-INHALT1234"	Dateiname ist 16 Zeichen lang
CONCAT D1,(N$) TO D0,(N$)	Die Dateinamen sind zu lang

3.2.2 Copy (Kommandostring)

Format:

C lw1:ziel-name = lw2:quell-name1 , lw3:quell-name2

lw1, lw2, lw3 sind Laufwerknummern der jeweiligen Dateien.

Beispiele (nach OPEN 15,8,15):

PRINT§15, "C0:DATNEU=1:DATALT1,0:DATALT2

Eine Kopie der Dateien 'DATALT1' vom Laufwerk 1 und 'DATALT2' von 0 wird zur Datei 'DATNEU' auf Laufwerk 0 zusammengefaßt.

PRINT§15, "C1:DATALT1=1:DATALT1,0:DATALT2

Die Datei DATALT2 von Laufwerk 0 wird an die Datei DATALT1 auf Laufwerk 1 angekoppelt. Die alte Datei DATALT1 wird dadurch verändert.

3.3 Dateien umbenennen: RENAME / Rename

Mit diesem Kommando kann einer Datei ein neuer Name zugewiesen werden.

Diese Operation bezieht sich nur auf das Inhaltsverzeichnis und ändert am Inhalt der Datei nichts.

Wenn der alte Name nicht existiert, folgt die Meldung 'FILE NOT FOUND'. Wenn der neue Name schon existiert, folgt die Meldung 'FILE EXISTS'.

Die Dateinamen dürfen keine Jokerzeichen enthalten.

Da es sich um einen schreibenden Zugriff auf das Directory handelt, darf die Diskette nicht schreibgeschützt sein. Andernfalls wird '26 WRITE PROTECT ON ERROR' gemeldet.

3.3.1 RENAME (BASIC-4)

Format:

RENAME D lw , **"dateiname1" TO "dateiname2"** , U gn

Parameter:

lw	Laufwerksnummer, darf nur einmal angegeben werden
dateiname1	Quell-Dateiname
dateiname2	Ziel-Dateiname
gn	Gerätenummer

Beispiele:

RENAME D1,"DATALT" TO "DATNEU"	Die Datei DATALT wird in DATNEU umbenannt
A$="DATALT"	alter Dateiname
B$="DATNEU"	neuer Dateiname
RENAME D(LW),(A$) TO (B$)	wie vorher

Falsch:

RENAME D1,"DAT1" TO D0,"DAT2"	es darf nur 1 Laufwerk angegeben sein

3.3.2 Rename (Kommandostring)

Format:

R lw:neuer name = lw:alter name

lw = Laufwerk der Datei

Beide Male muß das gleiche Laufwerk angegeben werden!

Beispiele (nach OPEN 15,8,15):

PRINT 15,"R1:NEU.NAM=1:ALT.NAM"

Die Datei ´ALT.NAM´ auf Laufwerk 1 erhält den neuen Namen ´NEU.NAM´.

Falsch:

PRINT§15 "R1:NEU.NAM=0:ALT.NAM"

Beide Male muß das gleiche Laufwerk angegeben werden!

3.4 Dateien löschen: SCRATCH / Scratch

Mit diesem Kommando werden Dateien von der Diskette gelöscht.

Dabei wird der Name der Datei aus dem Inhaltsverzeichnis gelöscht und die von ihr belegten Blöcke in der BAM freigegeben. Der Inhalt der Datei wird nicht gelöscht, er könnte also mit den Mitteln des Direktzugriffs noch gelesen werden.

Der Platz der Datei im Inhaltsverzeichnis wird freigegeben, aber nicht aufgefüllt. Löscht man also eine Datei und speichert danach eine andere ab, so taucht sie an der Stelle des Directory auf, wo vorher die gelöschte war.

Beim Löschen können entweder eindeutige Dateinamen angegeben, oder Jokerzeichen benützt werden.

Bei den Jokerzeichen ist natürlich Vorsicht geboten. Solange man die Wirkung dieser Zeichen nicht völlig verstanden hat, sollte man sie in diesem Zusammenhang nicht verwenden, da dabei leicht Dateien gelöscht werden, die man gerne noch weiterhin auf der Diskette gehabt hätte!

Nach dem S-Kommando meldet das Floppy nach '01 FILES SCRATCHED' die Anzahl der gelöschten Dateien. Falls die angegebene Datei nicht existierte, wird nicht '62 FILE NOT FOUND' gemeldet, sondern '01 FILES SCRATCHED **00**'!

Da durch 'S' der Diskettеninhalt verändert wird, darf der Schreibschutz nicht 'geklebt' sein.

Es kann vorkommen, daß beim Löschen nicht alle Blöcke freigegeben werden, die eigentlich frei sein müßten. Mit 'COLLECT' können auch diese Blöcke freigegeben werden.

Dateien, die mit einem Stern im Inhaltsverzeichnis gekennzeichnet sind, werden wie folgt behandelt:

(1) Schreibdateien, die noch einen Kanal im Floppy reservieren, die also gerade bearbeitet werden, werden nicht gelöscht.

(2) Dateien, die irgendwann nicht richtig geschlossen wurden, werden gelöscht.

3.4.1 SCRATCH (BASIC-4)

Format:

SCRATCH D lw , **"name"** , U gn

Parameter:

lw	Laufwerksnummer
name	Dateiname oder Namensmuster mit Jokerzeichen
gn	Gerätenummer

Wird der Befehl im Direkt-Modus eingegeben, fragt der Rechner nach, ob man sicher ist mit:

ARE YOU SURE?

Das Kommando wird nur dann an das Floppy geschickt, wenn hier 'YES' oder 'Y' eingegeben wird.

Es ist möglich, bei "name" mehrere Dateinamen durch Komma getrennt anzugeben, solange der gesamte Dateiname 16 Zeichen nicht überschreitet.

Beispiele:

```
SCRATCH D1,"MISTDATEI"    Lösche MISTDATEI in Laufwerk 1
SCRATCH D1,"MIST*"        Lösche alle Dateien in Laufwerk 1, die mit MIST
                          beginnen
```

Falsch:

```
NA$=""
SCRATCH D1,(NA$)          Der Dateiname darf nicht leer sein

LW=0
SCRATCH DLW,"DATEI"       Variablen müssen in Klammern stehen
```

3.4.2 Scratch (Kommandostring)

Format:

S lw1:name1 , lw2:name2

Es können also mehrere Namen angegeben werden. Aber der Kommandostring darf seine maximale Länge von **40 Zeichen** nicht überschreiten.

lw1, lw2 = Laufwerknummern der Dateien

Beispiele (nach OPEN 15,8,15):

PRINT§15,"S0:MISTDATEI"

Vom Laufwerk 0 wird die Datei mit dem Namen 'MISTDATEI' gelöscht.

PRINT§15,"S1:M?ST*"

Vom Laufwerk 1 werden alle Dateien gelöscht, deren Name mit 'M?ST' beginnt, wobei '?' für jedes beliebige Zeichen steht.

PRINT§15,"S1:*"

Löscht alle Dateien auf Laufwerk 1. Dieses Kommando braucht aber u.U. länger als New (HEADER) ohne ID, obwohl es die gleiche Wirkung hat.

4. Programmbehandlung

4.1 Programme laden: DLOAD / LOAD

Diese BASIC-Befehle bewirken, daß die als Dateiname angegebene PRG-Datei in den Speicher geladen wird. Diese PRG-Datei ist bei normalen Anwendungsfällen ein BASIC-Programm, das in den BASIC-Speicher geladen wird und dort ablaufen kann.

Allgemein kann mit DLOAD/LOAD ein beliebiger Speicherbereich geladen werden. Die Anfangsadresse steht in den ersten beiden Bytes der PRG-Datei. Die darauf folgenden Bytes werden einfach Byte für Byte in den Speicher kopiert.

Wird das im Dateinamen stehende Programm nicht gefunden oder meldet das Floppy einen anderen Fehler (z.B. FILE TYPE MISMATCH), gibt der Rechner die Fehlermeldung FILE NOT FOUND ERROR aus.

Jokerzeichen

Bei der Angabe des Namens kann man die Möglichkeiten der Jokerzeichen benutzen. Werden solche Zeichen verwendet, wird die **erste** Datei im Inhaltsverzeichnis geladen, die mit dem angegebenen Muster übereinstimmt.

Hier ist Voraussetzung, daß die erste zutreffende Datei eine PRG Datei ist, sonst wird FILE TYPE MISMATCH gemeldet.

4.1.1 DLOAD (BASIC-4)

Format:

DLOAD "name" , D lw , U gn

Parameter:

name	Dateiname, muß PRG-Datei sein
lw	Laufwerk
gn	Gerätenummer

Wird als Dateiname '*' angegeben, wird die erste Datei auf der Diskette in Laufwerk 0 geladen.

Beispiele:

DLOAD"PROG1"	Lade Programm PROG1 von Laufwerk 0
DLOAD"PROG*",D1	Lade von Laufwerk 1 das erste Programm, das mit PROG* beginnt
N$="PROG*" DLOAD(N$),D1	wie vor, aber Programmname in Variable

Falsch:

LW=1 DLOAD"PROG*",DLW	Variablen müssen in Klammern stehen

Systemstart

Die Taste SHIFT/RUN lädt das erste Programm der Diskette in Laufwerk 0 und startet es sofort. Dies ist eine komfortable Möglichkeit, einen 'Systemstart' zu machen. Man muß nur als erstes Programm auf der Diskette ein Programm haben, das dem jeweiligen Benutzer erlaubt, alle Programme und Hilfsprogramme zu laden, die er benötigt.

Voraussetzung ist natürlich, daß die erste Datei eine PRG-Datei ist.

4.1.2 LOAD (normales BASIC)

Format:

LOAD "name" , gn

name := (laufwerknummer :) programmname

Von 'name' gibt es zwei Sonderformen, die unten erklärt sind.

gn ist die Gerätenummer

Die Laufwerknummer einschließlich des Doppelpunkts kann entfallen. Die angegebene Datei wird dann zuerst auf demjenigen Laufwerk gesucht, auf das zuletzt zugegriffen wurde. Wird sie dort nicht gefunden, wird auf dem anderen Laufwerk gesucht.

Für den Ausdruck 'name' gibt es zwei Sonderformen:

(1) name = *
(2) name = $ laufwerknummer (: dateiname (= typ))

Fall 2 ist bei DIRECTORY behandelt.

4.1.3 LOAD "*" , 8

'*' kann zwei verschiedene Wirkungen haben:

Solange noch keine Datei von der Diskette geladen wurde (z.B. gleich nach dem Einschalten), bewirkt 'LOAD"*",8', daß die **erste** Datei des Inhaltsverzeichnisses geladen wird.

Wenn schon ein Programm geladen wurde, wird das Programm geladen, das **zuletzt** angesprochen wurde.

Wenn zusätzlich zum Stern noch das Laufwerk angegeben wird, wird in jedem Fall die erste Datei geladen: LOAD "0:*" , 8

4.2 Programme abspeichern: DSAVE / SAVE

Mit diesen Kommandos können **BASIC**-Programme auf Diskette abgespeichert werden.

Auf der Diskette, die im angegebenen Laufwerk steckt, darf noch keine Datei mit dem angegebenen Namen existieren, sonst erfolgt die Fehlermeldung '63 FILE EXISTS'.

Der Dateiname darf keine Jokerzeichen enthalten, sonst meldet das Floppy '33 SYNTAX ERROR'.

Zu beachten ist, daß sich der Rechner nach dem Abspeichern immer mit 'READY' zurückmeldet, auch wenn ein Floppy-Fehler aufgetreten ist, also die Fehlerlampe leuchtet.

Da das Abspeichern ein schreibender Zugriff ist, darf der Schreibschutz nicht 'geklebt' sein.

4.2.1 DSAVE (BASIC-4)

Format:

DSAVE "name" , D lw , U gn

Parameter:

name	Dateiname, darf noch nicht vorhanden sein
lw	Laufwerksnummer
gn	Gerätenummer

Beispiele:

DSAVE"PROGNAME"	Speichere PROGNAME auf Laufwerk 0
DSAVE"PROG1",D1	Speichere PROG1 auf Laufwerk 1
N$="PROG1"	Programmname in Variable
DSAVE(N$)	wie vor

Falsch:

```
LW=1
N$="PROG1"
DSAVE N$,DLW          Variablen müssen in Klammern stehen
```

4.2.2 SAVE (normales BASIC)

Bei 'SAVE' muß unbedingt die Laufwerknummer angegeben werden. Man beachte diesen Unterschied zu 'LOAD'!

Format:

SAVE "laufwerknummer:programmname" , gn

Beispiel:

```
SAVE"0:PROG1",8       BASIC-Programm "PROG1" auf Laufwerk 0 abspeichern
```

5. Datenverkehr

5.1 Datenkanal öffnen: DOPEN / OPEN

Mit diesen Kommandos können Datenkanäle zum Floppy geöffnet werden. Bei 'Allgemeine Parameterbeschreibung / Sekundäradresse' ist kurz erklärt, was es mit den Kanälen auf sich hat.

5.1.1 DOPEN (BASIC-4)

Format:

Zur besseren Verständlichkeit wird die Format-Beschreibung in drei Fälle aufgeteilt. Dies ist sinnvoll, weil das Vorhandensein oder Fehlen von Parametern die Funktion des Befehls bestimmt.

(1) Öffnen einer SEQ-Datei zum Schreiben

DOPEN §la,"name" ,W , D lw , U gn

Parameter:

§la	Logische Adresse
name	Dateiname, darf keine Jokerzeichen enthalten
W	dieses Zeichen gibt an, daß eine SEQ-Datei zum Schreiben geöffnet wird.
D lw	Laufwerk
U gn	Gerätennummer

Existiert die Datei schon auf dem Laufwerk, ergibt sich die Fehlermeldung 'FILE EXISTS'.

Mit diesem Befehl lassen sich**nur SEQ-Dateien** zum Schreiben öffnen. Wegen PRG oder USR Dateien lesen Sie bitte bei OPEN nach.

'W' kann nicht durch eine Variable repräsentiert werden.

Beispiele:

```
DOPEN §5 , "TESTDATEI" , W              auf Laufwerk 0 , Gerät 8
DOPEN §5 , "TESTDATEI" , W , D1 , U9    auf Laufwerk 1 , Gerät 9
```

Beide Kommandos öffnen die Datei 'TESTDATEI' zum Schreiben unter der logischen Adresse 5.

(2) Öffnen einer SEQ-/PRG- oder USR-Datei zum Lesen oder einer bereits vorhandenen REL-Datei zum Schreiben und Lesen

DOPEN §la,"name" , D lw , U gn

Parameter:

§la	Logische Adresse
name	Dateiname
lw	Laufwerk
gn	Gerätennummer

Dieser Befehl kann Dateien vom Typ SEQ,PRG oder USR zum Lesen öffnen. Der Typ muß nicht angegeben werden, die Angabe des Dateinamens ist eindeutig.

Ist die Datei eine REL-Datei, kann sie **gelesen und beschrieben** werden. Sie muß allerdings schon vorher eingerichtet worden sein mit der Möglichkeit (3).

Ist die Datei nicht vorhanden, meldet das Floppy 'FILE NOT FOUND'.

Beispiel:

DOPEN §3 , "NAME" , D1 , U9

Datei NAME unter logischer Adresse 3 auf Laufwerk 1 von Gerät 9 zum Lesen öffnen, wenn ihr Typ SEQ, PRG oder USR ist, bzw. zum Lesen und Schreiben, wenn der Typ REL ist.

(3) Öffnen einer REL-Datei

DOPEN §la,"name" , L rl , D lw , U gn

Parameter:

§la	Logische Adresse
name	Dateiname
rl	Recordlänge dieser REL-Datei
lw	Laufwerk
gn	Gerätennummer

Ist die REL-Datei noch nicht vorhanden, wird sie mit der angegebenen Recordlänge eingerichtet. Die Recordlänge kann Werte zwischen 2 und 254 haben.

Werden als Recordlänge Werte kleiner als 1 oder größer als 254 angegeben, meldet der Rechner ILLEGAL QUANTITY ERROR.

Ist die REL-Datei schon vorhanden, wird sie geöffnet und die angegebene Recordlänge mit der Recordlänge der Datei verglichen. Stimmen diese nicht überein, meldet das Floppy '50 RECORD NOT PRESENT'. Die Datei ist dann im Floppy nicht geöffnet. Sie muß aber im Rechner vom Programm geschlossen werden, da sonst beim nächsten Öffnen ein FILE OPEN ERROR gemeldet wird.

Beispiel:

DOPEN §3 , "DIR-DATEI" , L 50

Unter logischer Adresse 3 die neue REL-Datei 'DIR-DATEI' mit Satzlänge 50 zum Schreiben und Lesen öffnen.

LN=150
DOPEN §3 , "name" , L(LN)

Wie vor, aber Satzlänge 150 Bytes und als Variable angegeben.

Zu REL-Dateien

Beim Öffnen einer REL-Datei muß nicht angegeben werden, ob sie zum Schreiben oder zum Lesen geöffnet wird. Nach dem Öffnen kann sowohl aus der Datei gelesen werden, als auch in die Datei geschrieben werden.

Nach dem Öffnen wird der Record-Zeiger auf den ersten Record der Datei gesetzt. D.h. die nächste Eingabe von dieser Datei bringt den ersten Record.

Wird eine REL-Datei neu eingerichtet und noch nicht beschrieben, werden für sie zwei Blöcke belegt, davon ein Block für Daten und ein Block für die interne Organisation.

Anzahl offener Dateien im Floppy

Jede offene Datei belegt im Floppy eine bestimmte Anzahl von Puffern. Die Gesamtanzahl der Puffer begrenzt damit die Anzahl der gleichzeitig möglichen offenen Dateien in einem Floppy.

Eine REL-Datei belegt 3 Puffer, jede andere Datei als REL belegt zwei Puffer. Das Floppy hat insgesamt 10 Puffer zur Verfügung.

Damit ergeben sich die Maximalzahlen an geöffneten Dateien zu:

5 SEQ
3 REL
2 REL und 2 SEQ
1 REL und 3 SEQ

Zur Sekundäradresse

Sie müssen sich beim Gebrauch von DOPEN nicht um die Zuweisung der SA kümmern, da dies der Rechner bei der Ausführung des Befehls selbst macht.

Wenn Sie aber noch andere Kanäle zum Floppy öffnen, die nur mit OPEN funktionieren, sollten Sie wissen, welche SA's das DOPEN vergibt, damit Sie nicht dieselben verwenden und damit die Kanalorganisation des Floppys durcheinanderbringen.

DOPEN beginnt die Zuweisung der SA bei SA=2. Jedes weitere DOPEN zählt die SA um eins hoch. Ist diese SA schon belegt, wird sie nicht verwendet, sondern die nächste freie SA verwendet.

5.1.2 OPEN

Wegen der generellen Beschreibung von OPEN schlagen Sie bitte im Rechnerhandbuch nach.

Format

OPEN la,gn,sa,dn

la	Logische Adresse	s. Allgemeine Parameterbeschreibung
gn	Gerätenummer	"
sa	Sekundäradresse/Kanalnummer	"
dn	Dateiname/Kommandostring	

la, gn und sa **müssen** angegeben werden. Die zulässigen Werte und ihre Bedeutung sind bei ´Allgemeine Parameterbeschreibung´ zu finden.

Dateiname (dn) / Kommandostring

Der vierte Parameter des OPEN ist ein String. Er kann verschiedene Informationen enthalten:

- Eine Dateidefinition (s. unten)
- Eine Puffernummer (s. Direktzugriff)
- Ein Kommando (s. Übermittlung des Kommandostrings)

Die Möglichkeit, eine Dateidefinition anzugeben, soll im folgenden behandelt werden. Durch Angabe eines Dateinamens im OPEN eröffnet man eine serielle Datei des Typs SEQ (oder auch PRG, USR) zum Lesen oder Schreiben. Einzelheiten zur Behandlung solcher Dateien sind im Abschnitt ´Dateibehandlung und Datenverkehr´ zu finden. Hier soll nur der Inhalt von dn erklärt werden.

Format von dn (4. OPEN-Parameter)

(laufwerk :) name (, typ) , zugriff (, länge)

laufwerk	Laufwerknummer	0 oder 1
name	Dateiname	s. Dateiname
typ	Typ	P oder U, Ersatzwert = SEQ
zugriff	Zugriffsart	W (schreiben) oder L (REL = schreiben/lesen) oder A (Append = anhängen)
länge	Record-Länge	1 ... 254

Hinweise zum Format:

Wird das Laufwerk nicht angegeben, muß auch der Doppelpunkt entfallen.

Im String dürfen keine Blanks enthalten sein, es sei denn, sie gehören zum Dateinamen.

Wenn DN aus mehreren Strings zusammengesetzt wird, darauf achten, daß die Kommas vor Typ, Zugriff und Länge nicht vergessen werden!

Laufwerknummer

Beim schreibenden Zugriff **muß** das Laufwerk angegeben werden, beim Lesen kann es (einschließlich des Doppelpunktes!) entfallen. Wird die Laufwerknummer beim Lesen nicht angegeben, wird die Datei auch auf dem anderen Laufwerk gesucht, falls sie auf dem ersten nicht gefunden wird!

Dateityp

In den beiden häufigsten Fällen kann die Angabe des Typs entfallen:

SEQ: Wird anstatt des Typs W (schreiben) angegeben, wird automatisch eine SEQ-Datei eingerichtet:

OPEN 2,8,2,"1:SCHREIBDATEI,W"

REL: Wird anstatt des Typs L (Länge) angegeben, wird automatisch eine REL-Datei eingerichtet:

OPEN 2,8,2,"1:S/L-DATEI,L,"+CHR$(50) (Recordlänge 50 Bytes)

Will man eine **PRG**-Datei öffnen, kann auch hier auf die Angabe des Typs verzichtet werden, wenn Kanalnummer 0 (lesen) oder 1 (schreiben) gewählt wurde:

OPEN 2,8,0,"PROGRAMM"

Kanal 0 ('LOAD') stellt automatisch 'lesen' und 'P' ein, unabhängig davon, was als dritter und vierter Parameter von DN angegeben wurde! Entsprechend stellt Kanal 1 ('SAVE') automatisch 'schreiben' und 'P' ein!

Damit ist die Angabe des Typs überhaupt nur nötig, wenn entweder eine PRG-Datei unbedingt eine Kanalnummer zwischen 2 und 14 haben soll, oder eine USR-Datei gewünscht wird.

Beim **Lesen** von Dateien wird der Typ automatisch dem Inhaltsverzeichnis entnommen.

Zugriffsart

Wird die Zugriffsart nicht angegeben, so wird **Lesen** angenommen.

W (write) richtet eine SEQ (oder PRG oder USR) Datei zum **Schreiben** ein.

L (length=Länge) bewirkt, daß erstens eine (neue) REL Datei eingerichtet wird und zweitens der nachfolgende Parameter als Record-Länge interpretiert wird.

A (append = anhängen) bewirkt, daß an eine schon bestehende Datei weitere Daten angehängt werden können (s. APPEND).

Recordlänge

Nach L als Zugriffsart muß **durch Komma getrennt** ein Byte folgen, daß die Recordlänge angibt:

OPEN 2,8,2,"1:DATEINAME,L,"+CHR$(50) (Recordlänge 50 Bytes)

Beispiele

```
OPEN LA,8,KA,LW$+":"+NA$+","+TY$+","+LS$
```

Dieses Beispiel zeigt einen sehr allgemeinen Aufbau, in dem jeder Parameter über eine eigene Variable eingestellt werden kann:

Der **logischen Datei** LA
wird (vom Rechner-Betriebssystem)
auf dem **Gerät** 8
der **Kanal** KA zugeordnet.

Diesem Kanal wird (vom Floppy-Betriebssystem)
auf dem **Laufwerk** LW$
die Datei mit
dem **Namen** NA$,
dem **Typ** TY$
und dem **Zugriff** LS$
zugeordnet.

```
OPEN LA,8,KA,"0:LESEDATEI1,S,R"
OPEN LA,8,KA,"0:LESEDATEI1"          identische Kurzform
```

Hier soll die Übergabe aller Dateidaten in einer einzigen Stringkonstanten gezeigt werden:

Auf **Laufwerk** 0 wird die **SEQ** - Datei **'LESEDATEI1'** zum **Lesen** geöffnet. Auf diese Datei kann dann nur mit 'INPUT§' oder 'GET§', aber nicht mit 'PRINT§' zugegriffen werden!

```
OPEN LA,8,KA,"1:SCHREIBDATEI1,P,W"
```

Auf **Laufwerk** 1 wird die Datei **SCHREIBDATEI1** als **PRG** -Datei zum **Schreiben** angemeldet. Auf diese Datei kann dann nur mit 'PRINT§', aber nicht mit 'INPUT§' oder 'GET§' zugegriffen werden!

```
OPEN 3,8,2,"1:"+N$+",P,W"
```

Auf dem **Laufwerk** 1 des Floppy 8 wird über den **Kanal** 2 die **PRG** Datei N$ zum **Schreiben** über 'PRINT§ 3,..' angemeldet.

```
OPEN 2,8,2,"1:DATEINAME,L,"+CHR$(50)
```

Die REL-Datei 'DATEINAME' wird mit einer Recordlänge von 50 Bytes eingerichtet.

Falsche OPEN Statements:

OPEN **A$**,B,C,N$

falsch, weil die ersten drei Parameter keine **Strings** sein dürfen! Ergibt SYNTAX ERROR (IN ..)

OPEN 1,8,7,"0:";N$;",S,W"

falsch, weil es sich um eine Stringverknüpfung und nicht um ein 'PRINT' handelt! Die ';' müssen durch '+' ersetzt werden.

OPEN 3,8,2,"1:"+N$+"P,W"

falsch, weil das Komma vor P fehlt (falls es nicht in N$ enthalten ist).

OPEN 2,8,2,"0:DATEINAME,L,"+"50"

falsch, weil 50 als 1 Byte und nicht als Ziffernstring übergeben werden muß.

5.2 Datei weiterbeschreiben APPEND / OPEN

Dieses Kommando wirkt im Rechner genauso wie ein DOPEN/OPEN auf eine serielle Datei zum Schreiben. Im Floppy werden aber die Schreibzeiger auf das Dateiende gesetzt. Die nächste Schreiboperation (PRINT§) auf diese Datei schreibt ab dem Ende der Datei weiter.

Damit besteht die Möglichkeit, zusätzliche Daten in eine schon geschlossene serielle Datei (SEQ, PRG, USR) zu schreiben.

Das Kommando kann nur auf schon bestehende Dateien angewandt werden, sonst meldet das Floppy '63 FILE NOT FOUND'.

Das Kommando kann natürlich nur zum Schreiben verwendet werden. Der Schreibschutz darf deshalb auch nicht geklebt sein.

5.2.1 APPEND (BASIC-4)

Format:

APPEND §la , "name" , D lw , U gn

Parameter:

la	Logische Adresse
name	Dateiname
lw	Laufwerk
gn	Gerätenummer

Beispiel:

APPEND §3 , "DATEI" , D1

An die Datei 'DATEI' mit der logischen Adresse 3 auf Laufwerk 1 sollen weitere Daten angehängt werden.

5.2.2 OPEN (normales BASIC)

Format:

OPEN la , gn , sa , dn

Alle Parameter sind so definiert, wie bei OPEN beschrieben. Beim Parameter 'Zugriff' wird A eingetragen (anstatt W oder L).

Beispiel:

OPEN 2,8,2,"1:ANHAENGEN,A"

An die Datei 'ANHAENGEN' auf Laufwerk 1 können mit PRINT§2 weitere Daten angehängt werden.

5.3 Datenkanäle schließen: DCLOSE / CLOSE

'DCLOSE/CLOSE' meldet die Datei ab, die durch 'DOPEN/OPEN' der logischen Adresse la zugeordnet wurde. Man sagt auch, die Datei wird **geschlossen.** Jedem Öffnen einer Datei im Programm muß ein Schließen zugeordnet werden, sobald die betreffende Datei nicht mehr gebraucht wird.

'DCLOSE/CLOSE' wirkt sowohl auf das Rechner-Betriebssystem, als auch auf das Floppy DOS. Im Rechner wird dadurch die betreffende logische Dateinummer wieder freigegeben. Im Floppy wird der entsprechende Kanal freigegeben. Diese beiden Wirkungen gelten für schreibenden und lesenden Zugriff.

Lesender Zugriff:

Lesedateien werden im Rechner und im Floppy einfach geschlossen, wobei an der Datei selbst nichts verändert wird. Dies bezieht sich nicht auf REL-Dateien, da diese immer zum Schreiben **und** Lesen geöffnet werden.

Schreibender Zugriff:

Beim schreibenden Zugriff auf **serielle Dateien** sind folgende Wirkungen zu beachten:

(1) Der letzte Datenblock wird vom Puffer im Floppy auf die Diskette geschrieben und als letzter Block der Datei gekennzeichnet.

(2) Die BAM (Verzeichnis der belegten / freien Blöcke der Diskette) wird auf die Diskette geschrieben. Erst damit sind die von dieser Datei belegten Blöcke auf der Diskette als belegt gekennzeichnet.

Dateien, die nicht mehr gebraucht werden, sollten sofort geschlossen werden. Damit vermeidet man zwei böse Konsequenzen:

(1) Wie bei 'Datenkanal öffnen' erwähnt, dürfen gleichzeitig nicht mehr als 10 Dateien geöffnet sein. Versuchen Sie, eine 11 Datei zu öffnen, erhalten Sie die Fehlermeldung TOO MANY FILES.

(2) Wenn eine serielle Datei beschrieben wurde und nicht irgendwann vor Programmende geschlossen wird, wird diese Datei auf der Diskette nicht geschlossen.

Die Folgen von 2 sind:

Solche Dateien können nicht gelesen oder kopiert werden. Die betreffende Datei erscheint im Inhaltsverzeichnis mit einem '*' und wird beim nächsten 'Bereinigen der Diskette (COLLECT)' vernichtet. Der letzte angefangene Datenblock wird nicht auf die Diskette geschrieben, er könnte also auch nicht mit Methoden des Direktzugriffs gelesen werden!

Es macht keinen Unterschied, ob die Datei durch DOPEN/OPEN oder durch APPEND geöffnet wurde. Mit APPEND können also Dateien, die schon einmal ordnungsgemäß geschlossen wurden, doch noch zerstört werden.

REL-Dateien:

REL-Dateien werden schon während der Bearbeitung blockweise auf dem aktuellen Stand gehalten. Auch die BAM wird bei jedem Zugriff auf einen neuen Block wieder aktualisiert. Beim Schließen der REL-Datei wird der letzte Blockzugriff in der Datei auf der Diskette in Ordnung gebracht.

Wird eine REL-Datei nicht geschlossen, wird der letzte zugegriffene Block nicht auf die Diskette geschrieben. Damit fehlt auf der Diskette diese Information. Die Anzahl belegter Blöcke dieser Datei wird ebenfalls nicht aktualisiert. Aber alle von dieser Datei belegten Blöcke sind in der BAM als belegt gekennzeichnet. Dadurch können diese Blöcke nicht mehr überschrieben werden.

Es ergibt sich also möglicherweise kein großer Verlust, wenn eine REL-Datei nicht ordnungsgemäß abgeschlossen wird.

Wichtige Hinweise

Wird der Fehlerkanal (15) geschlossen, so schließt das **Floppy DOS alle anderen Kanäle** ebenfalls. Für den Rechner sind die entsprechenden Dateien aber noch offen, wodurch es zu Fehlermeldungen kommen kann. Ehe nicht alle Datenkanäle durch DCLOSE oder CLOSE geschlossen wurden, darf also der Fehlerkanal nicht geschlossen werden.

Außerdem schließt das Floppy alle Dateien eines Laufwerks, wenn die Diskette herausgenommen wird. Sie müssen also darauf achten, daß Sie die Diskette nicht zum 'Anschauen' herausnehmen dürfen solange Dateien geöffnet sind, solange also die Laufwerkleuchte brennt.

Durch 'LOAD' oder 'RUN' wird die Zuordnungstabelle von logischen Adressen zu Geräten und Sekundäradressen gelöscht. Für den Rechner sind alle Dateien dadurch geschlossen. Dies trifft aber nicht für das Floppy zu, da eben nur die Tabelle 'vergessen' wurde, aber kein ordnungsgemäßes 'CLOSE' durchgeführt wurde.

Dies gilt nicht, wenn 'LOAD' im BASIC Programm ausgeführt wird (OVERLAY). In diesem Fall bleiben alle Dateien geöffnet.

Wenn Sie Floppy-Dateien trotzdem noch schließen möchten, nachdem sie im Rechner schon geschlossen sind, können Sie dies tun, indem Sie den Kommandokanal öffnen und wieder schließen.

Wenn Sie eine Lesedatei geöffnet haben und versuchen, unter dem gleichen Kanal eine Schreibdatei anzumelden, ohne die Lesedatei durch CLOSE zu schließen, erhalten Sie weder vom Rechner, noch vom Floppy eine Fehlermeldung, haben aber trotzdem beliebig großen Ärger am Hals. Wenn Sie mit OPEN arbeiten, müssen Sie also streng darauf achten, den gleichen Kanal nicht gleichzeitig zwei verschiedenen Dateien zuzuordnen!

Den Befehl CLOSE schlagen Sie bitte im Rechnerhandbuch nach.

5.3.1 DCLOSE (BASIC-4)

Format:

DCLOSE § la , U gn

Parameter:

§ la	Logische Adresse
U gn	Gerätenummer

DCLOSE kann mehrere Dateien auf einem Gerät mit einem Befehl schliessen. Folgende Möglichkeiten gibt es:

(1) **Ohne** einen **Parameter** wird für die Gerätenummer der Ersatzparameter 8 genommen und alle Dateien auf Gerät 8 geschlossen.

Beispiel:

DCLOSE	Schließe alle Dateien auf dem Gerät Nr. 8

(2) Wird eine **Gerätenummer** angegeben, werden alle Dateien auf diesem Gerät geschlossen.

Beispiel:

DCLOSE U9	Schließe alle Dateien auf dem Gerät Nr. 9

(3) Bei Angabe einer **Logischen Adresse** wird nur die Datei mit dieser la geschlossen.

Beispiel:

DCLOSE §3	Schließe die Datei mit der Logischen Adresse 3

(4) Bei Angabe der **Logischen Adresse und Gerätenummer** wird die Gerätenummer ignoriert und nur die Datei mit der angegebenen la geschlossen, da die la eindeutig ist.

Beispiel:

DCLOSE §3,U10	Schließe die Datei mit der Logischen Adresse 3, ohne Berücksichtigung, auf welchem Gerät diese ist.

Die Angabe von la **und** gn ist also sinnlos.

5.4 Record-Zeiger positionieren: RECORD / Position

Dieses Kommando positioniert den Record-Zeiger einer REL-Datei auf ein Byte im angebenen Record (Datensatz).

Da REL-Dateien in wahlfreier Reihenfolge gelesen und beschrieben werden können, muß vor jeder Ein- oder Ausgabe-Operation der Record-Zeiger auf den gewünschten Record gesetzt werden.

Record-Zeiger auf neuen Block setzen

Wird der Record-Zeiger hinter den letzten beschriebenen Record der Datei gesetzt, gibt es zwei Möglichkeiten:

(1) Wird für den neu benötigten Record kein neuer Block auf der Diskette benötigt, merkt das Floppy nicht, daß auf diesem Record noch keine Information abgespeichert ist. Es bringt also auch keine Fehlermeldung.

Eingabe:

Bei einer nachfolgenden Eingabe mit 'INPUT§' wird ein Zeichen mit dem Code 255 übertragen.

Ausgabe:

Ein 'PRINT§' auf diesen Record legt die Daten in diesen neuen Record richtig ab.

(2) Wird für den angewählten Record ein neuer Block auf der Diskette benötigt, meldet das Floppy: '50, RECORD NOT PRESENT'.

Eingabe:

Wird danach ein INPUT§ auf die Datei durchgeführt, bleibt der Rechner in der Eingabe 'hängen' und ist damit 'gestorben'.

Ein GET§ bringt den Code 13 (CHR$(13)) und den Status 64.

Wichtig:

Vor einer Eingabe aus einer REL-Datei muß der Record-Zeiger gesetzt und danach die Floppy-Statusmeldung abgefragt werden. Meldet diese '50,RECORD NOT PRESENT', darf die Eingabe nicht durchgeführt werden!

Ausgabe:

Wird nach '50,RECORD NOT PRESENT' mit PRINT§ auf diesen Block geschrieben, so werden vom Floppy soviele neue Blöcke für diese Datei reserviert, wie zwischen dem letzten existierenden Record und dem neuen Record benötigt werden.

Es ist auch möglich, beliebig viele leere Records dadurch einzurichten. Stellt das Floppy fest, daß auf der Diskette nicht genügend freie Blöcke sind, meldet es den Fehler '52, FILE TOO LARGE'.

5.4.1 RECORD (BASIC-4)

Format:

RECORD § la , r , b

Parameter:

§ la	Logische Adresse der Datei
r	Record-Nummer in dieser REL-Datei Werte von 1 bis 65535
b	Stellung im Record Werte von 1 bis 254, Ersatzparameter ist 1

Die Parameter in diesem Befehl sind **stellungsabhängig.** Sie müssen also an den angegebenen Stellen stehen.

Dieser Befehl positioniert den Record-Zeiger der REL-Datei mit der Logischen Adresse la auf das b-te Zeichen im Record r.

Hat b einen Wert, der größer ist als die Recordlänge dieser REL-Datei, meldet das Floppy '51 OVERFLOW IN RECORD'.

Liegen r oder b nicht im erlaubten Bereich, meldet der Rechner ILLEGAL QUANTITY ERROR.

Beispiele:

RECORD §1,200	positioniere Recordzeiger der Datei mit la=1 auf 200
LA=1 RZ=200 RECORD §(LA),RZ	wie vor, mit Variablen

Falsch:

RECORD §1	die Angabe des Recordzeigers fehlt
RZ = 300	Recordzeiger 300
SR = 100	Stellung im Record soll 100 sein
RECORD §1,SR,RZ	die Variablen sind vertauscht (stellungsabhängig!)

5.4.2 Position (Kommandokanal)

Format:

P sa rl rh bp

sa	Sekundäradresse für Position (Kanalnummer) + 96
rl	Record-Nummer-Low-Byte
rh	Record-Nummer-High-Byte
bp	Byte-Position

Alle vier Parameter sind Byte-Parameter, müssen also z.B. mit Hilfe von CHR$ erzeugt werden:

"P"+CHR$(SA)+CHR$(RL)+CHR$(RH)+CHR$(BP)

Sekundäradresse

Bitte beachten Sie, daß sa die Sekundäradresse, also die Kanalnummer der entsprechenden Datei ist und nicht ihre logische Adresse wie bei RECORD. Dies ist der einzige Fall, daß auf eine Datei über die sa zugegriffen wird und nicht über die la!

Zum tatsächlichen Wert der sa muß 96 addiert werden. Wenn Sie also die Datei durch OPEN 1,8,2,... , also auf sa=2 geöffnet wurde, müssen Sie hier bei sa den Wert 98 angeben.

Recordnummer

Die Recordnummer muß in zwei Bytes zerlegt werden, da mit einem Byte nur 256 Werte darstellbar sind, eine REL-Datei aber mehr als 256 Records haben kann.

Wenn r die absolute Recordnummer ist, errechnen sich die beiden Teil-Bytes wie folgt:

```
rh = INT (r / 256)
rl = r - rh * 256
```

Byte-Position

Der Wert für die Byte-Position kann von 1 bis 254 reichen.

Beispiel (nach OPEN 15,8,15):

Sie haben eine REL-Datei mit der Kanalnummer 5 offen und wollen den Record/Byte-Zeiger auf Record 1000, Byte 15 stellen:

```
RH = INT (1000 / 256)
RL = 1000 - RH * 256

PRINT§15,"P"+CHR$(5+96)+CHR$(RL)+CHR$(RH)+CHR$(15)
```

6. Fehlermeldung und Fehlerbehandlung

6.1 Fehlertabelle

DS	DS$	F1	F2	
F	F$			
	Keine Fehler			
00	OK	0	0	Alles in Ordnung
01	FILES SCRATCHED	A	0	F1 enthält Anzahl gelöschter Dateien
	Lesefehler			
20	READ ERROR	T	S	Kein Block 'header'
21	READ ERROR	T	S	Kein Synchronisationszeichen
22	READ ERROR	T	S	Datenblock nicht vorhanden
23	READ ERROR	T	S	Prüfsummenfehler im Datenblock
24	READ ERROR	T	S	Byte wurde falsch dekodiert
27	READ ERROR	T	S	Prüfsummenfehler im 'header'
71	DIR ERROR	0	0	Directory Fehler
	Schreibfehler			
25	WRITE ERROR	T	S	Schreib- / Prüffehler
28	WRITE ERROR	T	S	Datenblock zu lang
	Syntax Fehler			
30	SYNTAX ERROR	0	0	Allgemeine Syntax
31	SYNTAX ERROR	0	0	Ungültiges Kommando
32	SYNTAX ERROR	0	0	Zu langes Kommando
33	SYNTAX ERROR	0	0	Ungültiger Dateiname
34	SYNTAX ERROR	0	0	Dateiname fehlt
39	SYNTAX ERROR	0	0	Ungültiges Kommando
	Bedienungsfehler			
26	WRITE PROTECT ON	T	S	Schreibschutz 'geklebt'
29	DISK ID MISMATCH	T	S	Disk ID stimmt nicht überein
50	RECORD NOT PRESENT	T	S	Record ist nicht vorhanden
51	OVERFLOW IN RECORD	T	S	Record ist zu kurz für die Daten
52	FILE TOO LARGE	T	S	Recordnummer ist zu hoch
60	WRITE FILE OPEN	0	0	Bereits Datei zum Schreiben geöffnet
61	FILE NOT OPEN	0	0	Datei ist nicht 'offen'
62	FILE NOT FOUND	0	0	Datei wurde nicht gefunden
63	FILE EXISTS	0	0	Datei existiert bereits
64	FILE TYPE MISMATCH	0	0	Dateitypendurcheinander
65	NO BLOCK	T	S	Kein Block frei
66	ILLEGAL TRACK AND SECTOR	T	S	Unerlaubte Spur oder Sektor
67	ILLEGAL SYSTEM T OR S	T	S	Unerlaubte Spur oder Sektor
70	NO CHANNEL	0	0	Kein Kanal frei
71	DIR ERROR	T	S	BAM ist defekt
72	DISK FULL	T	S	Diskette voll
73	CBM DOS V2.5	0	0	Meldung der DOS-Version
74	DRIVE NOT READY	0	0	Laufwerk enthält keine Diskette

6.2 Allgemeine Hinweise zur Fehlermeldung

Bei der Ausführung von Floppy - Kommandos kann es zu Fehlern kommen. Dabei gibt es Fehler, die zwangsläufig und vorhersehbar eintreten, wie etwa WRITE PROTECT ERROR. Daneben können aber auch Fehler auftreten, die zufälliger Natur sind, wie etwa der Fehler 25.

Die Fehler- bzw. Zustandsmeldungen des Floppy können über den Fehlerkanal abgefragt werden.

Wenn ein Fehler auftritt, wird dies durch die Fehlerlampe zwischen den beiden Laufwerken angezeigt. Dieser Hinweis dürfte allerdings nur beim Arbeiten im Direktmodus interessant sein. Sobald Floppy - Kommandos von Programmen aus gegeben werden, empfehlen wir, grundsätzlich nach dem Kommando die Fehlermeldung einzulesen, um den Benutzer möglichst schnell auf Fehler hinweisen zu können.

Da die meisten Fehlermeldungen durch ein neues Kommando gelöscht werden, muß nach jedem einzelnen Kommando die Fehlermeldung gelesen und ausgewertet werden. Dabei sind zwei Möglichkeiten zu unterscheiden:

(1) Die Fehlermeldung wird unmittelbar **nach** dem jeweiligen Kommando gelesen.

(2) Die Fehlermeldung wird unmittelbar **vor** dem nächsten Kommando gelesen.

Die erste Möglichkeit ist im Programm einfacher zu realisieren, hat aber den Nachteil, daß der Anwender warten muß, bis das Floppy das betreffende Kommando ausgeführt hat. INPUT§ wartet nämlich, bis die betreffende Operation zu Ende ist.

Die zweite Möglichkeit bietet den Vorteil, daß der Dialog weiterlaufen kann, während das Floppy arbeitet. Da das Programm aber dann 'rückwirkend' auf Fehler reagieren muß, wird die Struktur des Dialogteils erheblich anspruchsvoller als im ersten Fall.

6.2.1 Die Bestandteile der Fehlermeldung

Die Fehlermeldung besteht aus vier Teilen. Zuerst wird die **Fehlernummer** (F) gemeldet, dann der zugehörige **englische Klartext** (F$). Die beiden folgenden Zahlen geben **Spur** (F1) und **Sektor** (F2) an, wo der Fehler aufgetreten ist, soweit diese Angabe sinnvoll ist.

6.2.2 Lesen der Fehlermeldung durch DS und DS$ (ab BASIC-4)

Bei BASIC-4 stehen die beiden Variablen-Funktionen DS und DS$ zur Verfügung. Jeder Aufruf einer der beiden Variablen bewirkt, daß die Floppy-Fehlermeldung gelesen wird, sofern seit dem letzten Aufruf eine Floppy-Operation durchgeführt wurde.

DS enthält die Fehlernummer und DS$ die gesamte Fehlermeldung (einschließlich der Fehlernummer).

DS und DS$ holen automatisch die Meldung des zuletzt angesprochenen Gerätes ein. Sie können damit ohne weiteres mehrere Floppy-Geräte mit dieser einen Funktion bearbeiten.

6.2.3 Lesen der Fehlermeldung mit INPUT§ (nicht nötig ab BASIC-4)

Bei BASIC-3 kann die Fehlermeldung nicht im Direktmodus, sondern nur im BASIC Programm durch INPUT§ gelesen werden. INPUT§ setzt voraus, daß ein OPEN auf den Fehlerkanal (sa=15) durchgeführt wurde.

Der Klartext (2. Meldung) muß in eine Stringvariable gelesen werden, die drei anderen Meldungen können sowohl in String- als auch in Zahlenvariable gebracht werden:

```
10 INPUT§ 15, F, F$, F1, F2
```

Durch

```
20 PRINT F; F$; F1; F2
```

kann die Meldung auf den Bildschirm gebracht werden.

6.2.4 Fehlerbehandlung

Wenn vom Programm die weitere Fehlerbehandlung durchgeführt wird, reicht es, nur F zu lesen. Dann kann über F z.B. auf eine Tabelle mit deutschen Fehlermeldungen zugegriffen werden, oder/und in verschiedene Fehlerbehandlungsroutinen verzweigt werden.

6.3 Ursachen und Zusammenhänge

Im Folgenden sind die einzelnen Fehlermeldung beschrieben.

Die 'Fehlermeldungen' kann man aufteilen in die beiden 'Zustandsmeldungen' 0 und 1, die Lesefehler, Schreibfehler, Syntaxfehler und Bedienungsfehler.

6.3.1 Lesefehler

Bei Lesefehlern kann man mehrmals versuchen, die Datei zu lesen. Dabei kann hilfreich sein, die Diskette herauszunehmen, von Hand zu zentrieren und beim Einlegen sehr vorsichtig vorzugehen.

In jedem Fall sollte bei Lesefehlern sofort alles kopiert werden, was wichtig ist und noch kopiert werden kann!

Wenn beim Einlesen von Dateien Lesefehler vom Programm nicht durch eigene Prüfungen erkannt werden, sollte nach jedem INPUT oder GET die Fehlermeldung des Floppy gelesen werden.

6.3.2 Schreibfehler

Der Fehler 25 kann bei schlechten Disketten auftreten. Das Floppy liest einen gerade aufgezeichneten Sektor (Block) und vergleicht ihn mit dem Inhalt des Puffers im Arbeitsspeicher. Dadurch wird geprüft, ob die Aufzeichnung fehlerfrei und lesbar erfolgt ist.

Wenn also Fehler 25 gemeldet wird, sollte die betreffende Diskette vermutlich möglichst umgehend aus dem Verkehr gezogen werden!

Da nur mit dem Inhalt des Floppy-Arbeitsspeichers verglichen wird und nicht mit dem des Rechner-Speichers, werden Übertragungsfehler auf dem Weg zum Floppy durch diesen Test nicht erkannt. Wir empfehlen deshalb, wichtige Aufzeichnungen durch 'VERIFY' bei PRG-Dateien oder durch Lesen und Vergleichen bei sonstigen Dateien zu überprüfen.

6.3.3 Syntaxfehler (Formatfehler)

Die sechs Syntaxfehlermeldungen geben detailliert Auskunft über die Art des Fehlers.

30 und 39: Allgemeine Syntax

Hier wurde gegen die generellen Regeln des Kommandostringaufbaues verstossen. Mögliche Fälle:

- vor ':' nicht '0' oder '1'
- kein '=' zwischen Ziel- und Quellnamen
- kein ',' zwischen mehreren Quellnamen
- führendes Blank im Kommando

Siehe dazu auch die Beispiele bei den jeweiligen Kommandos!

31: Ungültiges Kommando

Hier wurde ein nicht definiertes Kommandozeichen gefunden. Die häufigste Ursache dafür dürfte sein, daß vergessen wurde, das Kommandozeichen zu schreiben und dadurch die Laufwerknummer oder der Dateiname an die Stelle des Kommandos gerückt ist.

32: Zu langes Kommando

Die Länge des Kommandostrings ist auf **40 Zeichen** begrenzt.

33: Ungültiger Dateiname

Hier wurde gegen die Regeln für den Dateinamen verstoßen. Mögliche Fehler wären z.B.:

- Länge größer als 16 Zeichen
- Ein verbotenes Zeichen wurde verwendet

In manchen Fällen wird aber bei längeren Namen einfach der hintere Teil abgeschnitten (beim Lesen und Schreiben). Ebenso werden unerlaubte Zeichen im Namen entweder mißinterpretiert, oder Fehler 30 wird gemeldet.

Es ist also angebracht, in Programmen eine Syntaxprüfung des Dateinamens selbst durchzuführen, um Überraschungen zu vermeiden!

34: Kein Dateiname angegeben

Bei den Kommandos N, und S muß mindestens 1 Name angegeben werden. Bei R müssen genau 2, bei C mindestens 2 Namen angegeben werden.

6.3.4 Fehler bei REL-Dateien

50: Record existiert nicht

Nachdem durch RECORD bzw. P ein (noch) nicht vorhandener Record angesprochen werden sollte, erscheint diese Fehlermeldung.

Werden dann durch PRINT§ Daten zu diesem Record gesandt, wird er eingerichtet und alles ist in Ordnung.

Dagegen darf auf keinen Fall versucht werden, nach dieser Meldung durch INPUT§ aus diesem Record zu lesen, der Rechner 'stirbt' daran!

51: Zu viele Daten für diesen Satz

Zu einem Satz dürfen nur so viele Bytes ohne CR (CHR$(13)) geschickt werden, wie die beantragte Satzlänge ist. Ist also Satzlänge 50 vereinbart, können nur maximal 49 Bytes + 1 CR in den Satz geschrieben werden. Das CR zählt also wie ein Datenbyte, da es mit in den Satz geschrieben wird.

Wenn 51 gemeldet wird, wurden die überzähligen Bytes 'weggeworfen'!

52: Satznummer zu groß

Wenn der verlangte Satz noch eingerichtet werden würde, würde DISK FULL auftreten. Diese Meldung kann ebenso wie 50 nach RECORD auftreten.

6.3.5 Bedienungsfehler

Die folgenden Fehlermeldungen werden als Bedienungsfehler bezeichnet, weil sie im Gegensatz zu (einigen) Schreib- / Lesefehlern bei entsprechender Vorgeschichte immer eintreten.

Sie sind also prinzipiell durch entsprechende Intelligenz des Programmes zu vermeiden. Natürlich kann man es bei manchen Meldungen auch 'drauf ankommen lassen' und im Fehlerfall definiert reagieren. Dies kann u.U. für den Benutzer unmerklich erfolgen.

26: Schreibschutz

Hier wurde versucht, schreibend auf eine schreibgeschützte Diskette zuzugreifen. Der Schreibschutz wird vor jedem schreibenden Zugriff überprüft.

Nachdem man den Schreibschutz entfernt hat, kann die Datei beschrieben werden.

29: Disk ID stimmt nicht überein

siehe 'Initialisieren'

60: Die Datei ist bereits zum Schreiben geöffnet

Auf eine Datei kann gleichzeitig entweder schreibend oder mehrmals lesend zugegriffen werden. Wenn also auf eine bestimmte Datei bereits schreibend zugegriffen wird (eine Datei geöffnet ist), kann nicht gleichzeitig eine zweite Datei unter dem gleichen Namen geöffnet werden. Dies schließt auch einen lesenden Zugriff ein.

61: Datei ist nicht geöffnet worden

Es ist grundsätzlich möglich, daß eine Datei zwar im CBM - Betriebssytem angemeldet ist, aber das Floppy dies durch aufgetretene Fehler wieder 'vergessen' hat. Dies ist z.B. immer der Fall nach CLOSE 15 oder nach dem Herausnehmen von Disketten.

62: Datei wurde nicht gefunden

Dies bedeutet, daß auf eine Datei lesend zugegriffen werden sollte, die im Inhaltsverzeichnis (unter dem angegebenen) Namen nicht enthalten ist.

Meistens ist ein Schreibfehler beim Dateinamen die Ursache für diese Fehlermeldung.

63: Datei existiert bereits

Hier sollte eine Datei unter einem Namen abgelegt werden, (schreibender Zugriff) der bereits im Inhaltsverzeichnis eingetragen ist.

Entweder gibt man einen anderen Namen an, oder man löscht die vorhandene Datei oder benennt sie um.

64: Dateitypen - Durcheinander

Bei Kommandos mit mehreren Dateinamen müssen sich alle Namen auf Dateien des gleichen Typs beziehen (PRG oder SEQ).

65: Kein Block frei

Meldung nach B-A, wenn der gewünschte Block nicht frei ist. Der 3. und 4. Parameter der Meldung enthält einen freien Block, der in der Regel der nächste freie Block ist.

70: No Channel

Mehr als 5 Datenkanäle stehen nicht zur Verfügung!

Hier sollte ein 6. Kanal geöffnet werden.

66 und 67: Nicht mögliche Spur und Sektor

Bei einem Direktzugriffkommando wurde eine nicht mögliche Spur-/Sektor-Kombination angegeben.

71: BAM zerstört

Die BAM im Arbeitsspeicher ist nicht in Ordnung. Versuchen Sie, durch Initialisieren nochmal die BAM der Diskette in den Arbeitsspeicher zu laden. Wenn der Fehler auch dann noch auftaucht, ist die Diskette nicht mehr lesbar.

Durch diese 'Rettungsaktion' werden die Blöcke offener serieller Schreibdateien wieder freigegeben, diese Datei-Inhalte sind also verloren.

72: Diskette voll

Alle Blöcke der Diskette sind bereits belegt, oder 224 Dateinamen sind bereits im Inhaltsverzeichnis.

73: Meldung der DOS-Version

Nach dem Einschalten des Floppy (Reset) steht diese Meldung im Fehlerkanal.

74: Laufwerk enthält keine Diskette

Wenn entweder die Klappe nicht geschlossen ist, oder eine nicht formatierte Diskette oder eine nicht auf 8050 formatierte Diskette enthalten ist, wird diese Meldung abgesetzt.

VI. DATEIBEHANDLUNG UND DATENVERKEHR

1. Der Dateibegriff

1.1 Einführung

Hier soll nicht versucht werden, Datei im Sinne der Informatik zu definieren, sondern nur Begriffe wie 'logische Datei', 'serielle' und 'sequentielle' Dateiverwaltung, 'Direktzugriff', 'Feld' und 'Satz' erklärt werden.

Eine 'Datei' enthält 'Daten' und Daten sind aus Bytes zusammengesetzt. Statt Daten kann man auch Datenfelder oder kurz Felder sagen. Mehrere Felder können einen (Daten-)satz bilden.

Für die BASIC - Ebene reicht es, ein Byte als kleinste Einheit des Arbeitsspeichers zu definieren. Ein Byte kann dezimal ausgedrückt Werte zwischen 0 und 255 annehmen, also 256 verschiedene Werte oder Zustände darstellen.

Alle Daten im Rechner sind in einem oder mehreren Bytes dargestellt. So steht eine Gleitkommazahl z.B. in 5 Bytes, ein Variablenname in 2 Bytes und ein String braucht soviele Bytes, wie er Zeichen hat. BASIC- oder Assemblerprogramme sind ebenfalls Byte für Byte im Speicher abgelegt.

Da der Rechner vollkommen auf der Bytestruktur (des Mikroprozessors) aufbaut, liegt es nahe, daß auch ein Massenspeicher wie das Floppy auf der Bytestruktur aufgebaut ist.

Damit ist das Floppy in der Lage, Bytes vom Rechner zu übernehmen und später auf Anforderung wieder zurückzugeben. Dazu sind natürlich Regeln nötig, um aus den gut 1.000.000 möglichen Bytes die richtigen auszuwählen.

Damit ist ein sehr wichtiges Merkmal der Datei erwähnt worden. Jede Datei braucht Regeln, wie sie aufgebaut ist. Diese Regeln bestimmen immer die Reihenfolge der Daten, manchmal auch ein bestimmtes 'Muster'.

Ehe auf die beiden speziellen Dateitypen des Floppy eingegangen wird, soll kurz das Konzept der 'logischen Datei' erläutert werden.

1.2 Die 'logische' Datei

Auf diese abstrakte Bedeutung wird deshalb eingegangen, weil sie für das tiefere Verständnis der Zusammenhänge zwischen OPEN und den eigentlichen Ein/ Ausgabebefehlen 'PRINT§', 'INPUT§', 'GET§' wichtig ist.

'Hinter' diesen drei Befehlen steckt immer eine Datei oder ein Datensatz des Typs 'seriell'. Wohin, bzw. woher die Daten kommen, ist dabei zweitrangig. So besteht bei der Eingabe kein prinzipieller Unterschied zwischen Tastatur, Bildschirm, Band oder Floppy. Dabei gilt die Analogie zur Tastatur für GET§ und zum Bildschirm für INPUT§. Entsprechend ist für die Ausgabe unwichtig, ob sie auf Bildschirm, Band, Floppy oder Drucker geht.

Diesem Umstand wurde Rechnung getragen, indem die Ein- / Ausgabebefehle mit 'logischen Dateien' über deren Nummern verkehren. Erst durch 'OPEN' wird einer logischen Datei ein physikalisches Gerät mit speziellen Eigenschaften zugeordnet.

Dadurch ist das Programm nur noch auf eine logische Datei**struktur** festgelegt, und nicht auf ein bestimmtes Peripheriegerät.

1.3 Vorstellung der verschiedenen Datei- bzw. Zugriffsarten.

1.3.1 Die serielle Datei (SEQ, PRG, USR)

Die einfachste Dateiform ist die 'Serielle Datei'. Hier werden einfach die Bytes genau in der Reihenfolge abgespeichert, wie sie ans Floppy übergeben werden. Entsprechend gibt das Floppy später die Bytes in genau der gleichen Reihenfolge wieder an den Rechner zurück.

Unter 'serieller Datei' fassen wir die drei Dateitypen **SEQ**, **PRG** und **USR** zusammen. Sie sind im Aufbau identisch. Ihre wichtigsten Eigenschaften sind, daß sie durch einen **Namen gekennzeichnet** sind, (fast) beliebig lang oder kurz sein können und vollständig vom Floppy-DOS verwaltet werden.

Der Begriff 'serielle Dateiverwaltung' ist nicht zu verwechseln mit der seriellen Datenübertragung zum Floppy. Der Datenverkehr erfolgt nämlich immer seriell, d.h., es wird ein Byte nach dem anderen übertragen.

Wie erwähnt, werden bei serieller Abspeicherung alle Bytes einfach in ihrer zeitlichen Reihenfolge abgelegt. Bei Programmen (PRG) ist dies sinnvoll, bei anderen Daten meist nicht.

1.3.2 Sätze, Felder, Schlüssel, ISAM

Bei den meisten kommerziellen Anwendungen benötigt man nicht Dateien, die aus einer endlosen Byte-Kette bestehen, sondern unterteilte Dateien. So wie ein ganzer Text in Sätze unterteilt ist, besteht eine derartige Datei ebenfalls aus sogenannten Datensätzen. Diese Sätze sind wiederum in Daten-Felder unterteilt. Am Beispiel einer Adressdatei sollen diese Begriffe erläutert werden:

Eine Adresse bildet einen Datensatz. In den einzelnen Feldern dieses Satzes stehen Angaben zu 'Name', 'Postleitzahl', 'Wohnort' usw.

Was erwartet nun der Anwender von einer Adressverwaltung? Zunächst will er Adressen 'erfassen' (eingeben und abspeichern). Später will er weitere Adressen eingeben können oder nicht mehr benötigte Adressen löschen können. Hauptsächlich will er aber eine bestimmte Adresse möglichst schnell auf dem Bildschirm (oder Papier) stehen haben.

Diese 'bestimmte Adresse' will er aber möglichst nicht nur unter einer Nummer, z.B. der Kundennummer, sondern auch unter einem immer verfügbaren Kennzeichen, z.B. dem Namen des Kunden abrufen können.

Der Programmierer, der diese Aufgabe lösen soll, erwartet seinerseits eine Dateiverwaltung, die seine Daten möglichst ohne Klimmzüge 'schluckt'. Denkbar wäre folgendes Verfahren: Er 'bestellt' beim DOS eine Datei mit dem Namen 'Adressen'. Diese Datei soll Sätze mit jeweils 8 Feldern haben. Die Länge der einzelnen Felder ist frei wählbar, muß aber bei der 'Bestellung' festgelegt werden. Daraus ergibt sich die Länge eines Satzes.

Der Zugriff auf einen bestimmten Satz erfolgt entweder einfach über die laufende Nummer des Satzes oder über den Inhalt eines bestimmten Feldes, den sogenannten Schlüssel.

Der Schlüssel kann z.B. der Nachname sein. Alle Nachnamen werden dann in einem getrennten Verzeichnis gehalten, das sehr schnell das Auffinden des gewünschten Namens erlaubt. Aus dem Verzeichnis geht dann hervor, wo der Daten-Satz steht, der zu diesem Namen gehört.

Die nächste Stufe stellt der Zugriff über mehrere 'Schlüssel' dar. Z.B. kann der Zugriff auf eine Adresse über die Kundennummer, den Namen oder die Telefonnummer erfolgen.

Diese Dateiverwaltungsform wird 'Index-Sequentielle-Verwaltung' oder ISAM = 'Index Sequential Access Method' genannt. In dieser komplexen Form ist sie nicht im DOS vorhanden. Aufbauend auf der REL-Datei kann aber durch BASIC- oder Assembler-Ergänzungen die eine oder andere Spielart einer ISAM-Datei realisiert werden.

1.3.3 Die REL-Datei

Die REL-Datei bietet einen eingeschränkten ISAM-Zugriff: Löschen und Einfügen von Sätzen ist nicht möglich, aber Überschreiben und Anhängen ist möglich. Der Zugriff erfolgt über die Satznummer, also nicht über einen alpha-numerischen Schlüssel.

Diese Zugriffsart wird auch mit 'Direktzugriff' bezeichnet, weil man über die Nummer direkt auf einen bestimmten Satz zugreifen kann. Genauer ist es ein Direkt-Zugriff mit wählbarer Satzlänge.

Diese Form der Dateiverwaltung wird deshalb REL-Datei genannt (REL = relativ), um sie vom absoluten Direktzugriff zu unterscheiden. Relativ ist dieser Zugriff deshalb, weil die Sätze nur innerhalb der Datei nummeriert sind, also relativ zum Anfang der Datei.

REL-Dateien werden genauso wie serielle Dateien unter einem Namen im Inhaltsverzeichnis der Diskette abgelegt.

1.3.4 Der Direktzugriff

Der Direktzugriff des Floppy DOS kennt sogenannte Blöcke auf der Diskette, die durch Spur- und Sektornummer adressiert werden. Jeder Block kann maximal 255 Bytes aufnehmen. Eine Diskette bietet maximal 2082 Blöcke für Direktzugriff. Ein Block kann in 0.5 - 2 sec beschrieben oder gelesen werden.

Diese Art des Zugriffs ist in der Regel vom Programmieraufwand her dem REL--Zugriff unterlegen und wird deshalb nur für Sonderanwendungen im Kapitel 'Direktzugriff' beschrieben.

Hauptnachteile gegenüber REL sind:

Feste Blocklänge, was meistens Platzverschwendung bedeutet.

Keine laufende Satznummer, sondern Spur- und Sektor-Nummer.

Nicht über Namen rufbar, was sehr starre Einteilung der Diskette bringt.

2. Generelle Vorgehensweise

Ehe wir im Einzelnen erklären, wie SEQ oder REL Dateien behandelt werden, wollen wir das generelle Schema, das immer gültig ist, vorstellen. Die benötigten Befehle sind im Rechnerhandbuch (PRINT§, INPUT§, GET§) beschrieben bzw. zusätzlich im Floppy-Handbuch: (D)OPEN, APPEND, (D)CLOSE.

2.1 Datenübertragung

Die Datenübertragung wird durch die Befehle 'PRINT§', 'INPUT§' und 'GET§' organisiert. 'PRINT§' ist der einzige Befehl, mit dem Daten vom Rechner gesendet werden können, er bringt also Daten zum Floppy. 'INPUT§' und 'GET§' besorgen die Datenübertragung in die andere Richtung, sie lesen also Daten.

2.2 Datei anmelden

Ehe Sie aber mit einem dieser drei Befehle auf eine Floppy-Datei zugreifen können, muß diese sowohl beim Rechner als auch beim Floppy 'angemeldet' werden. Diese Funktion wird durch 'DOPEN' / 'OPEN' oder APPEND ausgeführt. Sie übergibt folgende Information:

An den Rechner:

(1) Die Logische Adresse, über die sich nachfolgende 'PRINT§', 'INPUT§' oder 'GET§' auf diese Datei beziehen.

(2) Die Gerätenummer, auf der die Datei liegt oder eingerichtet wird.

(3) Die Kanalnummer, die bei Verwendung von 'DOPEN' intern vom Rechner verwaltet wird.

An das Floppy:

(1) Die Kanalnummer, die für das Floppy ungefähr dieselbe Funktion hat wie die Logische Adresse für den Rechner.

(2) Die Laufwerksnummer

(3) Den Dateinamen und Dateityp der gewünschten Datei. Dieser wird im Directory (Inhaltsverzeichnis) auf der Diskette vom DOS verwaltet.

(4) Bei seriellen Dateien wird dem Floppy mitgeteilt, ob die Datei beschrieben oder gelesen werden soll.

2.3 Datei abmelden

Soll die Behandlung einer Datei abgeschlossen werden, muß sie wieder abgemeldet werden. Dies ist nur mit den Befehlen 'DCLOSE' oder 'CLOSE' möglich.

3. Serielle Dateien

3.1 Einführung

Die serielle Datei ist dadurch gekennzeichnet, daß in ihr die Daten nur in der Reihenfolge gespeichert werden können, wie sie zum Floppy gesendet werden. Entsprechend können sie nur in derselben Reihenfolge wieder gelesen werden.

Die Länge einer seriellen Datei ist nach oben nur durch die Kapazität der Diskette begrenzt. Hätte man also nur eine einzige Datei auf einer Diskette, so könnte diese 2052 * 254 = 521 208 Bytes fassen.

Die kleinste organisatorische Einheit auf der Diskette ist ein Block mit 254 Bytes gültiger Information. Es können also immer nur ganze Blöcke belegt werden. Daraus folgt, daß eine gültige serielle Datei mindestens einen Block belegt und der letzte Block der Datei immer ganz belegt wird, auch wenn er nur 1 Byte gültige Information enthält.

Das Floppy kennt die drei seriellen Dateitypen SEQ, PRG und USR. Die wichtigsten Typen sind SEQ für Abspeicherung von Daten und PRG für Programme. USR ist identisch mit SEQ. Der einzige Unterschied liegt in der anderen Bezeichnung im Directory.

3.2 Beispielprogramm für SEQ-Dateien

Das Beispielprogramm soll Ihnen die Zugriffsmöglichkeiten auf SEQ-Dateien verdeutlichen. Wir werden Sie bei der Beschreibung im folgenden immer wieder auf das Beispiel verweisen. Die meisten Fehlermöglichkeiten können Sie damit nachvollziehen. Deshalb ist es wohl sinnvoll, wenn Sie das Beispiel vor dem Durcharbeiten dieses Kapitels in Ihren Rechner eintippen, damit Sie dann sofort alle Möglichkeiten nachvollziehen können.

Hinweis: Bei den Programmbeschreibungen bedeuten die Nummern in Klammern die Zeilennummern des Beispiels.

Das Beispiel beinhaltet

die **Funktionen**	**im Zeilenbereich**
(1) Daten erfassen	200 - 299
(2) Daten lesen	300 - 399
(3) Datei weiter beschreiben	400 - 499

Diese Funktionen werden durch den Programm-Verteiler (100-180) dem Benutzer angeboten. Durch Eingabe der davorstehenden Zahl kann die gewünschte Funktion ausgewählt werden. Bei Eingabe einer '0' wird das Programm beendet.

In jeder Funktion werden Sie zuerst nach dem Dateinamen (DN$) und der Laufwerksnummer (LW) gefragt.

3.3 Der schreibende Zugriff (Daten erfassen)

3.3.1 Einleitung

Bei 'schreibend' wird eine neue Datei unter dem angegebenen Namen eingerichtet. Diese wird während der Bearbeitung im Inhaltsverzeichnis mit einem Stern gekennzeichnet. Erst beim richtigen Abmelden der Datei mit 'DCLOSE' oder 'CLOSE' wird der Stern im Inhaltsverzeichnis gelöscht. Dann erst kann die Datei von der Diskette wieder gelesen werden.

3.3.2 Fehlermöglichkeiten

Wird während der Bearbeitung der Datei die Diskette herausgenommen, schließt das Floppy alle Kanäle auf diesem Laufwerk. Damit kann die Datei nicht mehr richtig abgeschlossen werden und bleibt mit einem Stern gekennzeichnet. Auf sie kann nicht mehr zugegriffen werden. Bei einem 'Bereinigen' der Diskette (COLLECT) wird sie wieder aus dem Inhaltsverzeichnis gelöscht.

Existiert der angegebene Dateiname schon, erfolgt die Fehlermeldung 63 FILE EXISTS.

Ist die Datei allerdings mit einem Stern gekennzeichnet, weil sie nicht richtig abgeschlossen wurde, kommt diese Fehlermeldung nicht, sondern die ungültige Datei wird durch die gültige ersetzt.

Wenn eine Datei zum Schreiben angemeldet ist und man versucht unter demselben Dateinamen eine Schreibdatei zu eröffnen, meldet das Floppy keine Fehlermeldung. Allerdings kann das Floppy danach die Dateien nicht mehr auseinanderhalten und damit sind beide nicht mehr zu lesen. Sie müssen deshalb unbedingt selbst durch Programmabfrage diese Möglichkeit unterbinden.

Will man eine zum Schreiben angemeldete Datei nochmal zum Lesen öffnen, meldet das Floppy '60 WRITE FILE OPEN'.

Ist der Schreibschutz geklebt, wird '26 WRITE PROTECT ON' gemeldet.

Auf eine Schreib-Datei kann nur mit 'PRINT§' zugegriffen werden, nicht mit 'INPUT§' oder 'GET§'. Sollte dies versucht werden, kommt keine Fehlermeldung. Lediglich im Status ST wird der Fehler mit ST=2 gemeldet. Das Floppy weigert sich also, aus einer Schreibdatei zu lesen und der Rechner erkennt eine Zeitüberschreitung beim Lesen.

3.3.3 Programmbeschreibung

Nach Eingabe von Dateiname und Laufwerk wird die Datei zum Schreiben geöffnet (220). Beachten Sie, daß im 'DOPEN' das 'W' für Schreiben (Write) nicht durch eine Variable repräsentiert sein kann!

Danach wird bei einer Floppy-Fehlermeldung (230) auf den Fehlerausgang (295) gesprungen.

In 235 bis 260 kann beliebig oft Text eingegeben werden (240), der sofort auf die Datei geschrieben wird (260). Nach jedem Schreibzugriff wird der Status ST abgefragt. Wenn er nicht Null ist, gibt das Programm eine Meldung aus und springt auf den Ausgang. Danach (265) geht es, abhängig vom Floppy-Status auf die nächste Eingabe (240) oder auf den Fehlerausgang (295).

Wird als Text "ENDE" eingegeben, springt das Programm auf den Ausgang der Routine (290). Hier wird die Datei geschlossen und wieder zum Programmverteiler verzweigt.

Beim Fehlerausgang (295) wird die Floppy-Fehlermeldung ausgedruckt und die Datei geschlossen. Die Datei muß im Rechner geschlossen werden, auch wenn sie auf der Diskette nicht mehr richtig abgeschlossen werden kann. Sonst würde beim nächsten Öffnen der Datei ein 'FILE OPEN ERROR' kommen.

3.4 Der Lesende Zugriff (Daten lesen)

3.4.1 Einleitung

Hier wird eine bestehende Datei zum Lesen geöffnet und die Daten in der Reihenfolge wie beim Schreiben gelesen.

3.4.2 Fehlermöglichkeiten

Ist die angegebene Datei nicht vorhanden, meldet das Floppy '62 FILE NOT FOUND'.

Auf eine Lese-Datei kann nur mit INPUT§ oder GET§ zugegriffen werden, nicht aber mit PRINT§. Wird dies versucht, kommt **keine** Fehlermeldung. Lediglich im Status ST wird der Fehler mit ST=-127 gemeldet. Das Floppy weigert sich also in eine Lesedatei zu schreiben und der Rechner erkennt eine Zeitüberschreitung beim Schreiben.

Wird während der Dateibearbeitung die Diskette herausgenommen, schließt das Floppy die Datei. Will der Rechner danach weiter von dieser Datei lesen, bringt das Floppy keine Fehlermeldung. Aber der Status ST wird auf ST=2 gesetzt, der Rechner erkennt also hier eine Zeitüberschreitung beim Lesen.

3.4.3 Programmbeschreibung

Nach der Eingabe von Dateiname (305) und Laufwerk (310) wird die Datei zum Lesen geöffnet (320). Danach muß sofort der Floppy-Status abgefragt werden und bei einer Fehlermeldung auf den Fehlerausgang gesprungen werden.

Zur Datenbehandlung

In diesem Beispiel werden die Daten nacheinander aus der Datei gelesen und mit der laufenden Nummmer (I) auf den Bildschirm gedruckt. Sie werden also nicht in einem Feld gespeichert und sind nach dem nächsten Floppyzugriff wieder verloren.

Vor dem Lesen der Daten wird der Zähler (I) auf Eins gesetzt (340). Die Leseschleife folgt dann in den Zeilen 350 bis 380.

Nach dem Einlesen eines Satzes mit INPUT§ (350) wird der Status ST in der Variablen FS gemerkt. Dies ist notwendig, da beim nachfolgenden Lesen des Floppy-Status DS der Status wieder verändert wird und außerdem vor der Behandlung des Status ST der eingelesene Satz noch behandelt werden muß.

Ist in (360) der Floppy-Status DS nicht Null, springt das Programm sofort auf den Fehlerausgang, da dann auch nicht mehr die eingelesene Information behandelt werden muß.

Danach wird bei einem Status ST ungleich 0 (hier in FS gemerkt) aus der Schleife herausgesprungen. Bei Status gleich Null wird der Zähler (I) um 1 erhöht und der nächste Satz gelesen.

In Zeile 385 wird ein Status ST ungleich Null behandelt. Bei Status ST = 64 ist die Datei zu Ende, sie wird geschlossen und das Programm geht wieder zum Programm-Verteiler. Ist der Status aber nicht 64, liegt ein Fehler vor, was durch 'Fehler-Status' gemeldet wird. Dann wird die Datei ebenfalls geschlossen.

Zusammenfassung

Wichtig beim Lesen ist die richtige Behandlung des Status ST. Hier kann ST sowohl 'Ende' der Datei als auch einen Lesefehler anzeigen. Deshalb muß zuerst noch die eingelesene Information bearbeitet werden und dann abhängig vom Status verzweigt werden. Beachten Sie auch, daß nach jedem Peripheriezugriff die Variable ST verändert wird. Soll sie im Progamm später noch verwendet werden, muß sie also gemerkt werden.

3.5 Der schreibende Zugriff für Datei weiter beschreiben

3.5.1 Einleitung

Das Weiterschreiben einer Datei ist ein Sonderfall des schreibenden Zugriffs. Hier wird der Schreibzeiger nicht auf den Beginn einer neu angelegten Datei gesetzt, sondern auf das Ende einer schon bestehenden Datei. Ab dieser Stelle können weitere Daten in die Datei geschrieben werden.

3.5.2 Fehlermöglichkeiten

Ist die angegebene Datei nicht vorhanden, meldet das Floppy '62 FILE NOT FOUND'.

Die anderen Fehlermöglichkeiten sind identisch mit denen des schreibenden Zugriffs beim Erfassen einer neuen Datei.

Eine Fehlerkonsequenz soll hier noch herausgestellt werden. Wird die Datei nach dem Öffnen zum Weiterbeschreiben nicht richtig abgeschlossen, kann sie nicht mehr gelesen werden. Damit ergibt sich also die Möglichkeit, eine schon existierende richtig abgeschlossene Datei nachträglich zu zerstören.

3.5.3 Programmbeschreibung

Die Routine für das Weiterbeschreiben einer Datei steht in 400 bis 495.

Sie ist bis auf Zeile 420 identisch mit schreibendem Zugriff (200 bis 295). Für besseres Verständnis wurde sie hier als eigenständige Routine realisiert.

Nach der Eingabe von Dateiname (405) und Laufwerk (410) wird die Datei zum Weiterbeschreiben geöffnet mit dem Befehl 'APPEND'. Beachten Sie, daß 'APPEND' eine Sonderform des **'DOPEN' für Schreiben** ist, hier muß also nicht durch 'W' der Modus angegeben werden.

Alles weitere entspricht dem schreibenden Zugriff, wie er oben schon beschrieben ist.

3.6 SEQ - DATEI - DEMO

```
 100 PRINT"■PROGRAMM-VERTEILER"
 110 PRINT"0 -> PROGRAMMENDE"
 120 PRINT"1 -> DATEN ERFASSEN"
 130 PRINT"2 -> DATEN LESEN"
 140 PRINT"3 -> DATEI WEITER BESCHREIBEN"
 170 INPUTA
 180 ONA+1GOTO190,200,300,400 GOTO130
 190 END
 200 PRINT"■DATEN ERFASSEN"
 205 INPUT"DATEINAME";DN$
 210 INPUT"LAUFWERK";LW
 220 DOPEN#1,(DN$),D(LW),W
 230 IFDSGOTO295
 235 PRINT"BITTE TEXT EINGEBEN (BEENDEN -> ENDE) "
 240 INPUT TE$
 250 IFTE$="ENDE"GOTO290
 260 PRINT#1,TE$
 262 IFSTTHENPRINT"■FEHLER-STATUS: "ST:GOTO290
 265 IFDSGOTO295
 275 GOTO240
 290 DCLOSE#1:GOTO100
 295 PRINT"■FLOPPY-FEHLER: "DS$:DCLOSE#1:GOTO100
 300 PRINT"■DATEI LESEN"
 305 INPUT"DATEINAME";DN$
 310 INPUT"LAUFWERK";LW
 320 DOPEN#1,(DN$),D(LW)
 330 IFDSGOTO395
 340 I=1
 350 INPUT#1,TE$
 360 FS=ST:IFDSGOTO395
 370 PRINTI,TE$
 375 IFFSGOTO385
 380 I=I+1:GOTO350
 385 IFFS<>64THENPRINT"■FEHLER-STATUS "FS
 390 DCLOSE#1:GOTO100
 395 PRINT"■FLOPPY-FEHLER: "DS$:DCLOSE#1:GOTO100
 400 PRINT"■DATEI WEITER BESCHREIBEN"
 405 INPUT"DATEINAME";DN$
 410 INPUT"LAUFWERK";LW
 420 APPEND#1,(DN$),D(LW)
 430 IFDSGOTO295
 435 PRINT"BITTE TEXT EINGEBEN (BEENDEN -> ENDE) "
 440 INPUT TE$
 450 IFTE$="ENDE"GOTO290
 460 PRINT#1,TE$
 462 IFSTTHENPRINT"■FEHLER-STATUS: "ST:GOTO290
 465 IFDSGOTO295
 475 GOTO240
 490 DCLOSE#1:GOTO100
 495 PRINT"■FLOPPY-FEHLER: "DS$:DCLOSE#1:GOTO100
READY.
```

4. Relative Dateien, Direktzugriff

4.1 Einleitung

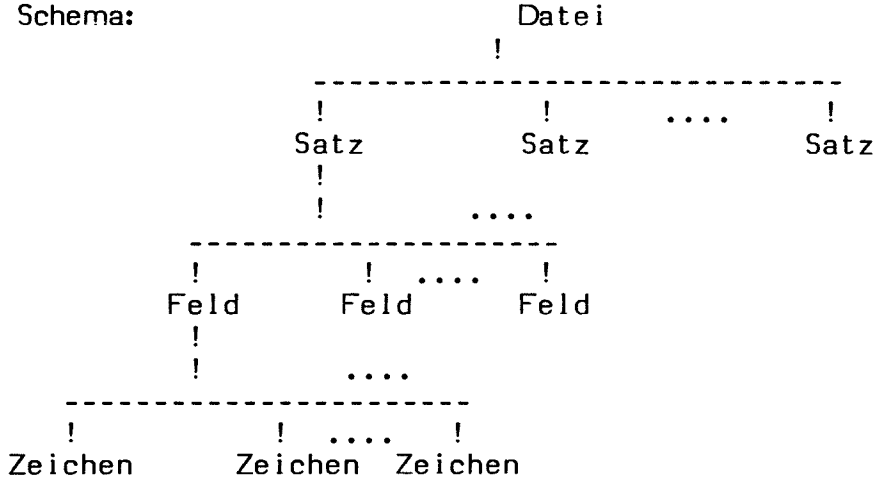
4.1.1 Begriffserklärungen

Hier sollen zuerst noch einige Begriffe erklärt werden, die in den folgenden Beschreibungen verwendet werden.

In den Befehlsbeschreibungen wurde bei Relativen Dateien immer über 'Records' geschrieben. Zu einem 'Record' werden wir im folgenden 'Satz' sagen. Eine Datei besteht also aus einzelnen Sätzen. Diese Sätze können aus mehreren Feldern bestehen, die innerhalb des Satzes durch Trennzeichen getrennt sind oder durch eine Formatvereinbarung schon eindeutig sind.

Die Felder wiederum bestehen aus Zeichen. Bei Textdateien werden die Zeichen normale druckbare Zeichen, also Text sein. Allgemein kann eine Datei aus beliebigen Zeichen des 256 Zeichen umfassenden Zeichensatzes bestehen.

Schema:



4.1.2 Die REL-Datei

REL-Dateien bieten die Möglichkeit, in wahlfreier Reihenfolge auf die Sätze einer Datei zuzugreifen, und zwar kann gleichzeitig in einen Satz geschrieben und aus dem Satz gelesen werden, ohne dafür zwei getrennte Dateien öffnen zu müssen.

Durch diese besonderen Möglichkeiten einer REL-Datei ergibt sich die Notwendigkeit, daß die einzelnen Sätze auf der Datei eine feste Satzlänge haben. Das bedeutet für den Anwender, daß er vor dem Erstellen einer REL-Datei die Maximallänge seiner Sätze kennen muß.

Die **Satzlänge** kann zwischen **1 und 254 Zeichen** liegen. Eine REL-Datei kann **maximal 65535 Sätze** haben, wenn diese Anzahl in 182880 Bytes Platz findet.

Eine REL-Datei kann **maximal 720 Blöcke** belegen, also **182880 Bytes** beinhalten.

Die Verwaltung von REL-Dateien wird ganz vom Floppy ausgeführt. Beim Neuerstellen der Datei reserviert das DOS sofort zwei Sektoren. Davon ist ein Sektor für Daten und der andere für die Organisation der Datei vorgesehen. Je nach Satzlänge sind also auch schon 1 (bei Satzlänge 254) bis 254 (bei Satzlänge 1) Sätze angelegt.

Wird vom Benutzer ein Satz gewünscht, der noch nicht eingerichtet ist, meldet das DOS beim Positionieren einen Fehler '50 RECORD NOT PRESENT'. Diese Fehlermeldung muß unbedingt beachtet werden, wenn man aus der Datei lesen will, da hier noch keine Information steht. Beim Schreiben wird die Datei vom DOS um soviele Blöcke (Sektoren auf der Disk) erweitert, wie die verbleibenden Sätze bis zu diesem neuen Platz benötigen. Außerdem werden eventuell noch neue Blöcke für die Organisation eingerichtet. Vorher prüft das DOS noch, ob genügend Platz auf der Diskette vorhanden ist und meldet es gegebenenfalls.

4.1.3 Beispielprogramm für REL-Dateien

Wir werden bei der Beschreibung der Möglichkeiten von REL-Dateien möglichst oft auf das Beispielprogramm verweisen, mit dem Sie die meisten (Fehler-) Möglichkeiten nachvollziehen können. Deshalb raten wir Ihnen, vor dem Durcharbeiten der Beispiele das Programm in den Rechner einzutippen.

Bei den Verweisen auf das Beispielprogramm sind die Zeilennummern in Klammern angegeben.

Das Programm beinhaltet die Funktionen:

(1) Neue REL-Datei einrichten
(2) Bestehende REL-Datei lesen und schreiben

Diese Funktionen werden über den Programm-Verteiler (100-180) angeboten und können durch Eingabe der davor angegebenen Zahl ausgewählt werden. Bei Eingabe von '0' wird das Programm beendet.

4.2 Einrichten einer REL-Datei

4.2.1 Einleitung

Das Neu-Einrichten einer REL-Datei ist als getrennte Funktion realisiert, weil nur hier die Satzlänge der Datei angegeben werden muß. Bei späteren Zugriffen auf die REL-Datei kann die Angabe der Satzlänge entfallen, muß aber nicht. Deshalb meldet das DOS auch keinen 'FILE EXISTS'-Fehler, wenn in dieser Funktion eine bestehende REL-Datei ausgewählt wird.

Bei der Datenerfassung wird zuerst die Satznummer abgefragt und dann der Text, der in diesen Satz geschrieben werden soll.

4.2.2 Fehlermöglichkeiten

Beim **Anmelden** der REL-Datei mit 'DOPEN' sind folgende Fehler möglich:

Besteht auf der Diskette schon eine Datei mit demselben Namen, die aber keine REL-Datei ist, meldet das Floppy '64 FILE TYPE MISMATCH'.

Besteht schon eine REL-Datei mit dem Namen, vergleicht das Floppy die angegebene Satzlänge mit der Satzlänge der Datei. Stimmen diese nicht überein, meldet es '50 RECORD NOT PRESENT'. Beachten Sie, daß diese Fehlermeldung damit zwei Funktionen hat.

Beim **Arbeiten** mit der Datei gibt es die Fehlermöglichkeiten:

Wird beim Positionieren auf einen Satz festgestellt, daß der verlangte Satz hinter dem letzten Block der Datei liegen würde, meldet das Floppy ebenfalls '50 RECORD NOT PRESENT'. Dies ist aber für einen nachfolgenden Schreibzugriff keine Fehlermeldung, da bei diesem Zugriff die nötigen Blöcke eingerichtet werden. Können sie nicht eingerichtet werden, weil auf der Diskette nicht genügend Platz ist, meldet das Floppy '52 FILE TOO LARGE'. Zusätzlich wird der Status ST auf ST=1 gesetzt, was eine Zeitüberschreitung beim Schreiben bedeutet.-

Ist die Datei im Floppy geschlossen worden, was durch Herausnehmen der Diskette erfolgt, meldet das Floppy nach dem 'RECORD'-Befehl einen Fehler 70 NO CHANNEL.

4.2.3 Programmbeschreibung

Nach der Eingabe des Dateinamens (205), der Satzlänge (210) und des Laufwerks (215) wird die REL-Datei durch 'DOPEN' geöffnet (220). In dem 'DOPEN'-Befehl wird der Typ der zu öffnenden Datei eindeutig durch die Angabe der Satzlänge hinter 'L' bezeichnet.

Nach dem 'DOPEN' wird der Floppy-Status DS abgefragt (230) und bei DS ungleich Null auf den Fehlerausgang (295) gesprungen.

In den Zeilen 235 is 275 steht die Datenerfassung, die solange wiederholt wird, bis für die Satznummer '0' eingegeben wird. Dies wird auch beim 'INPUT' auf die Satznummer (235) im Hinweistext ausgedruckt.

Nach der Eingabe wird sofort dieses Endekriterium überprüft und bei Ende zum Aussprung (290) gegangen, wo die Datei mit 'DCLOSE' abgemeldet wird.

Mit dem 'RECORD'-Befehl wird auf den gewünschten Satz positioniert (240). Wenn danach der Floppy-Status DS=50 ist, meldet das Programm, daß ein neuer Satz eingerichtet werden muß (245). Bei einem Floppy-Status, der nicht DS=0 und nicht DS=50 ist (248), geht das Programm auf den Fehlerausgang.

Nun wird die Eingabe des zu erfassenden Textes verlangt (250), der in der Variablen TE$ gespeichert wird. TE$ wird danach mit 'PRINT§' auf die Datei geschrieben (260). Hinter dem 'PRINT§'-Befehl müssen wieder sowohl der Status ST als auch der Floppy-Status DS abgefragt werden. Hier wurde die Reihenfolge so gewählt, daß zuerst der Floppy-Status DS (262) und dann erst der Status ST (265) abgefragt wird. Durch diese Reihenfolge muß der Status ST sofort nach dem 'PRINT§' in der Variablen FS gemerkt werden, da er beim Lesen des Floppy-Status DS wieder verändert wird.

Diese Reihenfolge der Statusabfrage muß nicht unbedingt so sein. Sie wurde von uns aus folgendem Grund in dieser Form gewählt:

Nach dem Schreiben auf einen Satz, der nicht mehr auf die Diskette paßt, kommt die Fehlermeldung '52 FILE TOO LARGE' und gleichzeitig steht in bestimmten Fällen der Status ST auf 1. Wird also ST zuerst abgefragt, verzweigt das Programm schon beim Feststellen des Wertes ST=1, stellt die Fehlermeldung 'FILE TOO LARGE' nicht mehr fest und kann sie deshalb auch nicht melden. Das Programm weiss also nur von einem Fehlerstatus zu berichten und weiß von dem wirklichen Fehler nichts.

Dies passiert nicht, wenn wie hier der Floppy-Status zuerst abgefragt wird.

Ergibt sich beim Schreiben des Textes kein Fehler, geht das Programm (275) wieder auf die Eingabe der Satznummer (235).

4.3 Bestehende REL-Datei lesen und schreiben

4.3.1 Einleitung

Ist eine REL-Datei eingerichtet, muß beim späteren Anmelden der Datei mit 'DOPEN' die Satzlänge nicht mehr angegeben werden. Das 'DOPEN' sieht also genauso aus wie beim Anmelden irgendeiner anderen Datei zum Lesen.

Die Datenerfassung ist bei dieser Funktion so realisiert, daß nach der Eingabe der Satznummer der Satz gelesen und für die Eingabe mit 'INPUT' vorgegeben wird. Durch Drücken von RETURN wird der Text übernommen und wieder abgespeichert. Er kann aber auch beliebig geändert werden.

4.3.2 Fehlermögichkeiten

Ist die angegebene Datei nicht vorhanden, meldet das Floppy '62 FILE NOT FOUND'.

Ist die Datei eine andere als REL-Datei, meldet das Floppy beim Anmelden mit 'DOPEN' keinen Fehler. Der falsche Dateityp wird erst festgestellt beim ersten Positionieren auf einen Satz mit 'RECORD'. Dann meldet das Floppy '64 FILE TYPE MISMATCH'.

Die anderen möglichen Fehlermeldungen sind schon beim Einrichten der REL-Datei beschrieben.

Zum Positionieren auf einen Satz hinter dem Dateiende muß hier noch hinzugefügt werden:

Meldet das Floppy nach 'RECORD' einen '50 RECORD NOT PRESENT', da der Satz hinter dem derzeitigen Ende der Datei liegen würde, darf auf keinen Fall mit 'INPUT§' aus dieser Datei gelesen werden. Die Eingabe greift nämlich 'ins Leere' und der Rechner stirbt. Deshalb ist bei der Datenbehandlung die Abfrage und Behandlung des Floppy-Status DS=50 sehr wichtig.

4.3.3 Programmbeschreibung

Nach der Eingabe des Dateinamens (405) und des Laufwerks (415) wird die Datei geöffnet (420). Der 'DOPEN'-Befehl hat hier dasselbe Format wie ein 'DOPEN' zum Lesen auf einen beliebigen Dateityp. Deshalb kann das Floppy hier noch nicht feststellen, ob ein falscher Dateityp angemeldet wurde. Nach 'DOPEN' wird bei gesetztem Floppy-Status DS auf den Fehlerausgang (595) gesprungen.

Die **Datenerfassungs-Schleife** steht im Zeilenbereich 435 bis 540. Sie wird beendet durch Eingabe einer Satznummer 0 in 435.

Nach der Eingabe der Satznummer (435) wird das Endekriterium abgefragt (438). Bei normalem Durchlauf wird dann mit 'RECORD' auf die Satznummer positioniert. Da auf den Satz nachher mit 'INPUT' lesend zugegriffen wird, ist hier die Abfrage des Floppy-Status DS (445) sehr wichtig. Bei einem Floppy-Status DS=50 wird der lesende Zugriff übersprungen, das Programm läuft bei 500 weiter. Bei einem anderen Floppy-Status springt es auf den Fehlerausgang.

Nach dem 'INPUT§' (450) wird der Status ST abgefragt. Bei diesem Vergleich darf ST=64 nicht berücksichtigt werden, da der Status ST nach richtigem Lesen eines Satzes den Wert 64 hat. Deshalb wird ST in der Abfrage mit dem Wert 255-64 UND-verknüpft. Bei einem Status ST, der nicht 0 und nicht 64 ist, meldet das Programm einen FehlerStatus und verzweigt zum Ausgang.

Nach dem Status ST wird der Floppy-Status DS abgefragt und bei gesetzten DS auf den Fehlerausgang gesprungen.

Nun wird der gelesene Text dem Benutzer zur Änderung angeboten. Er wird auf den Bildschirm ab der dritten Spalte gedruckt und danach der Cursor in die davorstehende Zeile (480) gesetzt. Damit setzt das nachfolgende 'INPUT' in der Textzeile auf.

Vor dem Schreiben des Textes in die Datei muß der Satzzeiger der REL-Datei wieder auf den aktuellen Satz positioniert werden, da er durch das 'INPUT§' in 450 schon um eine Stelle vorgerückt ist. Nach dem Positionieren wird bei einem Floppy-Status DS, der nicht 50 und nicht 0 ist, auf den Fehlerausgang verzweigt.

Jetzt wird der String TE$ in die Datei geschrieben (510). Danach wird bei einem Fehlerstatus ST ungleich 0 oder bei einem Floppy-Status ST ungleich 0 der Programmteil beendet. Wird kein Fehler angezeigt, geht die Datenerfassung bei 435 weiter.

4.5 Abspeicherformat

4.5.1 Endezeichen

Bei seriellen Dateien müssen die Texte durch Trennzeichen voneinander abgetrennt werden, damit ein 'INPUT§' in einem Text ein Zeilenende finden kann.

Bei REL-Dateien gibt es folgende Möglichkeiten:

(1) **Ein Satz besteht nur aus einem Feld.**

Dann wird mit einem Einlesebefehl der gesamte Satz gelesen. Dieser Fall wurde bei den obigen Beispielen zugrunde gelegt.

Hier muß der Satz nicht mit einem Trennzeichen abgeschlossen sein, da das Floppy nur soviel schickt, wie vorher in den Satz geschrieben worden ist. Dadurch kann die gesamte Satzlänge für den Text ausgenützt werden.

Beachten Sie, daß durch ein 'PRINT§', das nicht mit ';' abgeschlossen ist, ein CR (Carriage Return) hinterhergeschickt wird. Ist in diesem Fall die Textlänge so lang wie die Satzlänge, ergibt sich ein Floppy-Fehler '51 OVERFLOW IN RECORD', da der Rechner ein Zeichen mehr schickt.

Wird der Text mit 'INPUT§' eingelesen, muß die 80-Zeichen-Grenze beachtet werden. Sie können also maximal Satzlängen von 80 Zeichen behandeln. Beim Einlesen von längeren Sätzen meldet der Rechner einen STRING TOO LONG ERROR.

(2) **Ein Satz besteht aus mehreren Feldern.**

Dieses Problem ergibt sich, wenn Sie Sätze bearbeiten wollen, die länger als 80 Zeichen sind. Diese Sätze können nicht in einem Stück mit 'INPUT§' gelesen werden. Hier ist es nötig, die Sätze in mehrere Felder aufzuteilen, die nacheinander gelesen werden. Diese Felder müssen durch ein CR getrennt sein, damit die Datenübertragung in den Rechner an diesen Stellen unterbrochen wird.

Die Trennzeichen ',' und ':' statt CR können hier nicht verwendet werden, da sie nicht die Datenübertragung in den Rechner unterbrechen und damit ein STRING TOO LONG ERROR auftreten würde.

Schreiben mehrerer Felder

Das Schreiben von mehreren durch CR getrennten Feldern darf nicht durch mehrere 'PRINT§'-Befehle hintereinander ausgeführt werden. Wie beim seriellen Zugriff meldet der Rechner nach jedem 'PRINT§', daß die Information zu Ende ist und das Floppy positioniert den Satzzeiger der REL-Datei auf den nächsten Satz. Durch das nächste 'PRINT§' wird also schon auf den nächsten Satz der Datei geschrieben.

Sie müssen im Programm so vorgehen, daß der gesamte Text, der in einen Satz geschrieben wird, noch im Rechner zu einem langen String zusammengesetzt wird. Hier müssen an den Trennstellen auch schon die CR's mit eingebaut werden. Dieser lange String wird dann in die REL-Datei geschrieben.

Beispiel

Die Teilstrings stehen im Feld TE$(1) bis TE$(3). Jeder Teilstring ist maximal 80 Zeichen lang. Das Schreiben in den Satz kann dann so aussehen:

```
100 R$=CHR$(13)
110 FOR I = 1 TO 3
120 TE$ = TE$(I) + R$
130 NEXT
140 PRINT§1,TE$;
```

Bei diesem Beispiel können die wenigen Elemente auch in einem 'PRINT§'-Befehl stehen:

```
100 PRINT§1,TE$(1)R$TE$(2)R$TE$(3)
```

Das letzte CR wird automatisch durch den 'PRINT§'-Befehl angehängt.

4.5.2 Endekriterium beim Lesen

Nach jedem 'PRINT§'-Befehl meldet der Rechner durch eine besondere Leitung des IEEE-Bus, daß nun die Information zu Ende ist.

Ist bei Übertragungsende der Satz noch nicht voll, werden vom Floppy die restlichen Zeichen in diesem Satz mit Kode Null (CHR$(0)) aufgefüllt.

Beim Einlesen gibt es also **zwei Endekriterien:**

(1) Findet das DOS vor dem Ende des Satzes nur noch Kode 0 bis zum Satzende, bricht es nach dem letzten Nicht-Null-Zeichen die Übertragung ab und meldet dem Rechner, daß die Information zu Ende ist.

(2) Sonst werden alle Zeichen bis zum Satzende übertragen.

4.5.3 Abspeichern von binären Daten

Für das Abspeichern von binären Daten muß das Endekriterium bei der Übertragung beachtet werden.

Binäre Daten können aus beliebigen Zeichen bestehen, es können also auch beliebig viele Zeichen mit Kode 0 in ihnen enthalten sein. Bei normaler Abspeichung ist es also möglich, daß die hinteren Zeichen des Satzes aus Kode 0 bestehen. Beim Einlesen würde das Floppy hier ein Endekriterium sehen und dem Rechner nicht mehr alle Information zurückschicken, die er in dem Satz abgespeichert hat.

Man muß also dafür sorgen, daß als letztes Zeichen im Satz ein Kode steht, der nicht 0 ist. Bei bekannter Satzlänge können also alle gültigen Zeichen eingelesen werden. Man muß dabei beachten, daß die Satzlänge um ein Zeichen länger vorbelegt wird, als man gültige Zeichen benötigt.

4.6 Positionieren auf ein Byte eines Satzes

Der 'RECORD'-Befehl bietet die zusätzliche Möglichkeit, den Satzzeiger auf ein bestimmtes wählbares Zeichen im Satz zu setzen.

Dies wird durch den Parameter angegeben, der hinter der Satznummer im Befehl steht, wir nennen ihn hier Bytenummer.

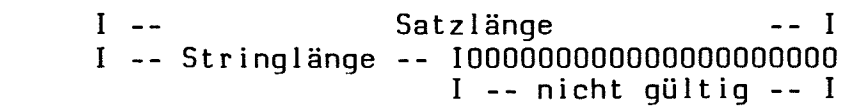
Die Bytenummer kann Werte von 1 bis 254 annehmen. Als Ersatzparameter für die Bytenummer wird vom Rechner der Wert 1 genommen.

Wenn die Sätze nicht immer ganz vollgeschrieben werden müssen die folgenden Dinge unbedingt beachtet werden:

Durch die Angabe einer Bytenummer, die nicht 1 ist, kann es vorkommen, daß der Satzzeiger hinter der gültigen Information im Satz aufsetzt.

Beispiel

Die Satzlänge ist 20 Zeichen und der letzte in diesen Satz geschriebene String war 10 Zeichen lang. Dann sind die restlichen 10 Zeichen mit Kode Null aufgefüllt. Wird die Bytenummer auf 17 gesetzt, steht der Satzzeiger in einem Bereich, der keine gültige Information beinhaltet.



Ein 'INPUT§' auf den Satz, das im nichtgültigen Bereich aufsetzt, bekommt vom Floppy beliebig viele Zeichen geschickt, ohne daß der Status ST auf den Wert 64 gesetzt wird. Deshalb meldet der Rechner einen STRING TOO LONG ERROR.

Steht als letztes gültiges Zeichen im String ein CR und setzt ein 'INPUT§' bei diesem Zeichen auf, 'stirbt' der Rechner. Er bleibt in der Eingabe hängen.

Sie sollten also die Möglichkeit, aus dem Satz ab einer bestimmten Bytenummer zu lesen, nur dann anwenden, wenn in den Sätzen der Datei wirklich formatierte Strings stehen, deren Länge immer gleich der Satzlänge ist.

4.7 REL - DATEI - DEMO

```
100 PRINT"■PROGRAMM-VERTEILER"
110 PRINT"0 -> PROGRAMMENDE"
120 PRINT"1 -> NEUE REL-DATEI EINRICHTEN"
130 PRINT"2 -> BESTEHENDE REL-DATEI LESEN UND SCHREIBEN"
170 INPUTA
180 ONA+1GOTO190,200,400
190 END
200 PRINT"■NEUE REL-DATEI EINRICHTEN"
205 INPUT"DATEINAME     ",DN$
210 INPUT"SATZLAENGE    ",RL
215 INPUT"LAUFWERK      ";LW
220 DOPEN#1,(DN$),L(RL),D(LW)
230 IFDSGOTO295
235 INPUT"SATZNUMMER (BEENDEN -> 0) ";RN
238 IFRN=0GOTO290
240 RECORD#1,(RN)
245 IFDS=50THENPRINT"■NEUER SATZ":GOTO250
248 IFDS<>0GOTO295
250 INPUT"TEXT                          ",TE$
260 PRINT#1,TE$;
262 FS=ST:IFDSGOTO295
265 IFFSTHENPRINT"■FEHLER-STATUS: "FS:GOTO290
275 GOTO235
290 DCLOSE#1:GOTO100
295 PRINT"■FLOPPY-FEHLER: "DS$:DCLOSE#1:GOTO100
400 PRINT"■BESTEHENDE REL-DATEI LESEN UND SCHREIBEN"
405 INPUT"DATEINAME     ";DN$
415 INPUT"LAUFWERK      ",LW
420 DOPEN#1,(DN$),D(LW)
430 IFDSGOTO595
435 INPUT"SATZNUMMER (BEENDEN -> 0) ";RN
438 IFRN=0GOTO590
440 RECORD#1,(RN)
445 IFDS=50THENPRINT"■NEUER SATZ":GOTO500
448 IFDSGOTO595
450 INPUT#1,TE$
460 IF(255-64)ANDSTTHENPRINT"■FEHLER-STATUS: "ST:GOTO590
470 IFDSGOTO595
480 PRINT"  "TE$:PRINT"■"
500 INPUTTE$
505 RECORD#1,(RN)
508 IFDS<>0ANDDS<>50GOTO595
510 PRINT#1,TE$;
520 IFSTTHENPRINT"■FEHLER-STATUS: "ST:GOTO590
530 IFDSGOTO595
540 GOTO435
590 DCLOSE#1:GOTO100
595 PRINT"■FLOPPY-FEHLER: "DS$:DCLOSE#1:GOTO100
READY.
```

5. PRG-Dateien

Der Aufbau einer PRG-Datei ist identisch mit SEQ- oder USR-Dateien. Sie kann auch mit dem 'DOPEN'-Befehl genau wie eine andere Datei zum Lesen geöffnet werden. Das DOS macht dabei keinen Unterschied zwischen den verschiedenen Dateitypen. Da bei 'DOPEN' keine Kanalnummer und kein Dateityp angegeben werden kann, muß zum Anmelden einer PRG-Datei für schreibenden Zugriff der 'OPEN'-Befehl verwendet werden.

Die speziellen Eigenschaften der PRG-Datei werden erst bei den Kanälen 0 und 1, bzw. beim Laden (DLOAD/LOAD) und Abspeichern (DSAVE/SAVE) sichtbar.

Zwei **Besonderheiten** sind demnach zu beachten:

Kanal 0 öffnet immer eine PRG Datei zum Lesen (DLOAD/LOAD/VERIFY) und Kanal 1 entsprechend zum Schreiben (DSAVE/SAVE). Deshalb braucht beim Laden und Abspeichern nicht der Dateityp und die Zugriffsart angegeben werden, weil beim Laden automatisch die Kanalnummer 0 und beim Abspeichern die Kanalnummer 1 gesendet wird.

'DSAVE' bzw. 'SAVE' übergeben in den ersten beiden Bytes einen Zeiger auf den Beginn des Speicherbereiches, der ab dem dritten Byte in der PRG-Datei abgelegt wird.

'DLOAD' bzw 'LOAD' interpretieren entsprechend die ersten beiden Bytes als Zeiger und laden ab dem dritten Byte alle Bytes ab der Zeigerstelle in den Arbeitsspeicher.

Es kann nun durchaus interessant sein, über 'PRINT§' oder 'INPUT§' auf eine PRG-Datei zuzugreifen. Ein Beispiel soll dies verdeutlichen:

Ein Assembler kann durch 'PRINT§' den erzeugten Maschinencode direkt auf die Diskette schreiben, was den großen Vorteil hat, daß der Code nicht erst im Arbeitsspeicher abgelegt werden muß. Dadurch wird nicht unnütz Arbeitsspeicher belegt und es können auch keine Konflikte zwischen dem Assembler und dem von ihm erzeugten Code entstehen.

Im umgekehrten Fall kann ein Disassembler direkt von der Diskette den Code lesen, den er bearbeiten soll.

VII DOS-ÜBERSICHT

1. Floppy Organisation

Dieser Abschnitt soll die interne Organisation des Floppy-Disk-Laufwerks beschreiben. Dabei wird nicht auf den physikalischen Aufbau eingegangen, sondern dem Programmierer sollen hier nur die logischen Verbindungen im Floppy erklärt werden.

Dieses Wissen ist für die Behandlung von SEQ, REL usw. Dateien nicht notwendig, da man mit den komfortablen FloppyKommandos immer direkt Dateien oder Programme auf der Diskette ansprach und sich somit nicht darum kümmern mußte, wie das Floppy diese Kommandos ausführt und auf welchen Wegen Daten von oder zur Diskette transportiert werden.

Will man aber die tieferen Möglichkeiten des Floppy-Disk-Betriebssystems (Abkürzung DOS = **D**isk **O**perating System) nutzen, dann sollte man einen Überblick über die interne Organistation des DOS haben.

1.1 DOS mit Pipeline-Struktur

Das DOS steuert die Handhabung des gesamten Informationsaustausches zwischen den Disketten und dem IEEE-488-Bus. Über diesen Bus wird die Verbindung zum Rechner organisiert. Die Handhabung des IEEE-488-Busses sollte dem Benutzer schon bekannt sein.

Wenn im folgenden vom "Floppy" die Rede ist, ist damit nicht die Diskette gemeint, sondern die Verbindung des IEEE-488-Bus, da hier die Stelle ist, an die der Rechner direkt angeschlossen ist.

Dazu steht dem DOS ein interner Schreib-/Lesespeicher zur Verfügung. Dieser Speicher wird als Puffer benutzt, um die Information, die von oder zu der Disk übertragen wird, zwischenzuspeichern. Dies ist sinnvoll, weil die Übertragung Rechner-Floppy wesentlich schneller geht als die Übertragung Floppy-Diskette.

Eine solche Struktur wird als **"Pipeline-Struktur"** bezeichnet. Der Pufferspeicher ist die "Pipeline" zwischen Rechner und Diskette und speichert die schnell anfallende Information vom Rechner, um sie dann an das langsamere Medium Diskette weiterzugeben.

Die Übertragung von Disk zum Rechner wird durch diese Pipeline-Sruktur auch unterstützt. Wenn der Rechner die anfallende Information nicht so schnell verarbeiten kann wie sie von der Diskette gelesen wird, wird sie im Pufferspeicher zwischengespeichert und kann bei Bedarf schnell an den Rechner weitergegeben werden. Dieser muß also nicht erst warten, bis die weitere Information von der Disk gelesen wird.

Das Floppy liest also "auf Verdacht" von der Diskette, denn es wäre natürlich auch der Fall möglich, daß der Rechner die weitere Information nicht mehr benötigt.

1.2 Pufferspeicher

Der Pufferspeicher hat eine Kapazität von 4 K-Bytes. Diese 4 K sind in 16 Bereiche zu je 256 Bytes aufgeteilt. Von diesen Bereichen werden 15 als Puffer verwendet, die mit Puffernummern von 0 bis 14 bezeichnet werden. 10 Puffer werden für die Datenübertragung verwendet, die restlichen benötigt das DOS selbst.

1.3 Kanalorganisation

Das Dateisystem ist durch sogenannte **Kanäle** organisiert. Diese Kanäle sind keine physikalischen, sondern rein logische Verbindungen im Floppy.

Man kann sie mit der Dateistruktur beim Rechner vergleichen. Wird dort eine Datei mit Logischer Adresse, Primäradresse und Sekundäradresse durch OPEN geöffnet, dann werden in einer Liste im Betriebssystem-Speicher die verschiedenen Adressen einander zugeordnet. Will der Programmierer nachher auf diese Datei zugreifen, muß er nur die logische Adresse ansprechen. Das Betriebssystem kann aus der Liste selbständig alle anderen Adressen entnehmen und damit den Datenaustausch organisieren.

Beim Floppy werden durch einen OPEN-Befehl einem Kanal ein, zwei oder drei Pufferbereiche zugeordnet. Bei OPEN auf eine Datei vom Typ SEQ oder PRG interessiert den Benutzer nicht, welcher Kanal welchem Puffer zugeordnet worden ist.

1.4 Gesamtübersicht über das DOS mit Verbindungen

Generell kann man also sagen, daß für den Programmierer alle Verbindungen im Floppy über Kanäle angesprochen werden. Der Kanal 15 als Kommandokanal hat einen Sonderstatus. Über ihn können **keine Daten** von oder zu der Disk übertragen werden.

Es gibt also im Floppy folgende logische Verbindungen, die auch im Bild eingetragen sind:

Für den Benutzer direkt zugreifbar:

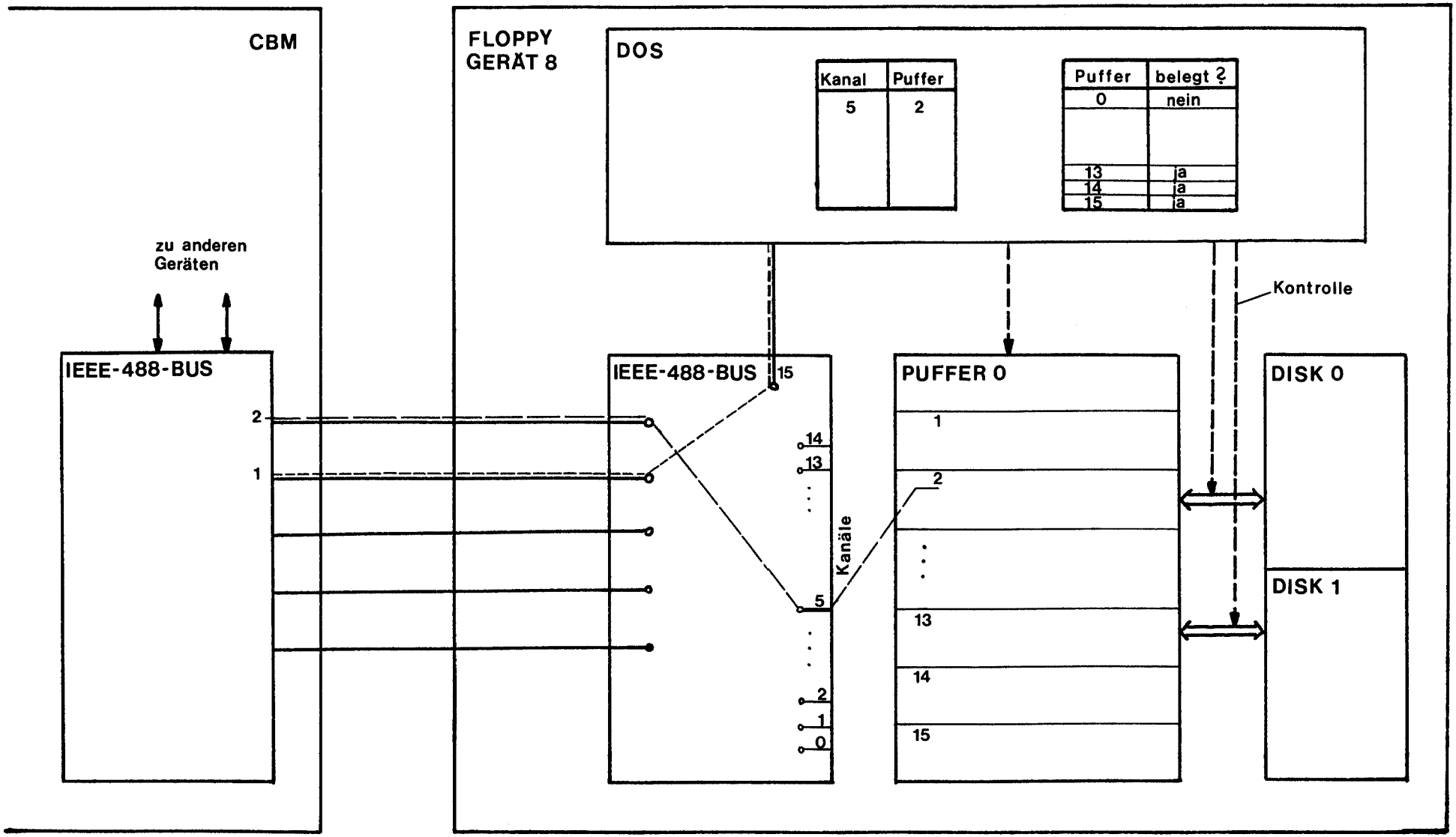
a) Rechner - Floppy IEEE-488-Bus (durch Logische Adresse)
b) Floppy IEEE-488-Bus - Pufferspeicher (durch Sekundäradresse = Kanal)
c) Floppy IEEE-488-Bus - DOS (durch Kommandokanal)

Für den Benutzer nicht direkt zugreifbar:

d) Pufferspeicher - Diskette

Diese Verbindung ist indirekt zu beeinflussen, indem man ein Kommando schickt, (über Kanal 15) in dem das DOS angewiesen wird, Information in einer gewünschten Richtung auf dieser Verbindung zu übertragen.

Schema der logischen Verbindungen im Floppy



Beispiel: OPEN 2,8,5"#2"
OPEN 1,8,15

2. Diskettenorganisation

2.1 Einteilung in Spur und Sektor

Damit man auf der Diskette auf bestimmte Teile der abgespeicherten Information zugreifen kann, ist diese in Spuren und Sektoren eingeteilt. Diese Einteilung wird bei der Formatierung einer Diskette vorgenommen.

Die Spuren sind konzentrische Kreise um den Mittelpunkt der Diskette. Diese sind weiter aufgeteilt in Sektoren.

Im Gegensatz zu den sonst üblichen Diskettenformaten ist die Anzahl der Sektoren pro Spur nicht auf der ganzen Diskette gleich. Da auf den äußeren Spuren der Umfang größer ist als weiter innen, hat hier eine Spur auch mehr Sektoren. Diese Art der Disketteneinteilung hat den großen Vorteil, daß ohne Erhöhung der maximalen Bitdichte wesentlich mehr Information auf die Diskette gepackt werden kann.

Es gibt insgesamt 77 Spuren, die mit Nummern 1 bis 77 durchnummeriert sind. Die Nummerierung der Sektoren beginnt in jeder Spur bei 0.

Beispiel: Bei Spur 1 gibt es die 29 Sektoren 0 bis 28.

Spur			**Sektoranzahl**		
			pro Spur	insgesamt	
1	bis	39	29	1131	
40	bis	53	27	378	
54	bis	64	25	275	
65	bis	77	23	299	
				2083	
				- 29	für Directory
				- 2	für BAM
				2052	Blöcke für freie Benutzung

2.2 Blockbelegung

Die kleinste auf der Diskette adressierbare Einheit, die mit Spur und Sektor angesprochen wird, nennt man **Block.** Spur/Sektor wird im folgenden als Blockadresse bezeichnet.

Ein Block enthält 256 Bytes an Information. Der Rechner kann 255 Bytes von der Diskette lesen oder auf die Diskette schreiben.

2.3 BAM = Block-Verfügbarkeits-Tabelle

Die BAM (**B**lock **A**vailability **M**ap) ist eine Liste, in der für jeden Block angegeben ist, ob dieser belegt ist oder nicht.

Wird zum Beispiel eine Datei auf den Blöcken Spur 17 Sektor 0 und Spur 17 Sektor 8 abgelegt, dann werden diese Blöcke in der BAM als belegt gekennzeichnet. Dadurch wird gewährleistet, daß diese Blöcke beim Abspeichern von anderen Dateien nicht mehr benutzt werden. Beim Löschen von Dateien werden die von dieser Datei belegten Blöcke wieder freigegeben.

Die BAM steht auf der Diskette auf Spur 38 Sektor 0 und 3. Ein Block steht nach dem Initialisieren immer im Pufferspeicher und wird bei Bedarf gegen den anderen ausgetauscht.

2.4 Inhaltsverzeichnis

Beim Abspeichern einer seriellen Datei auf eine Diskette wird der Name dieser Datei im Inhaltsverzeichnis vermerkt.

Für das Inhaltsverzeichnis ist auf der Diskette die ganze Spur 39 reserviert. Auf diese Spur werden deshalb keine Dateien abgespeichert.

In einem Block können bis zu acht Dateinamen stehen. Die Reihenfolge der Namen hängt von der Reihenfolge der Abspeicherungen ab. Beim Löschen einer Datei wird der Name im Inhaltsverzeichnis als gelöscht gekennzeichnet. Dadurch wird er beim Lesen des Inhaltsverzeichnisses nicht mehr berücksichtigt.

Beim Abspeichern einer Datei auf die Diskette wird das Inhaltsverzeichnis von vorne nach dem ersten freien Platz durchsucht. Dieser wird dann für den Namen der abzuspeichernden Datei verwendet.

In derselben Reihenfolge, wie die gültigen Namen im Inhaltsverzeichnis abgespeichert sind, werden sie auch gelesen.

Die Reihenfolge, der Dateinamen ist wichtig, wenn man nach dem Einschalten des Systems ein bestimmtes Programm mit SHIFT/RUN laden möchte. Wie schon beschrieben muß dieses Programm als erstes im Inhaltsverzeichnis stehen.

Dies erreicht man auf zwei Arten:

Wenn noch keine Datei auf der Diskette steht, wird dieses Programm als erstes abgespeichert.

Wenn die Diskette schon beschrieben ist, muß die erste Datei gelöscht und dann das benötigte Programm abgespeichert werden. Dessen Name wird in dem freien Platz abgespeichert.

2.5 Verwaltung der seriellen Dateien

Serielle Dateien werden auf der Diskette so abgespeichert, daß das DOS von Spur 39 nach unten und oben die nächsten freien Blöcke sucht. Normalerweise werden Blöcke verwendet, die nicht benachbart sind (Zeitoptimierung). Allgemein können serielle Dateien auf beliebigen Blöcken in beliebiger Reihenfolge stehen.

Damit das DOS beim Zugriff auf eine serielle Datei diese wieder richtig finden kann, werden die Blöcke durch Verweise verkettet.

Der **Zugriff** auf eine serielle Datei geht in folgender Reihenfolge:

Beim Namen der Datei im Inhaltsverzeichnis stehen Spur und Sektor des ersten Blocks dieser Datei.

Auf jedem Block der Datei steht ein Verweis auf den nächsten Block in den ersten zwei Bytes, und zwar die Spur in Byte 0, der Sektor in Byte 1.

Der letzte Block einer Datei ist dadurch gekennzeichnet, daß in Byte 0 der Wert 255 steht. In Byte 1 steht hier ein Zeiger auf das Ende der Datei im Block.

3. Tabellen für Direktzugriff

3.1 Standard-Sprung-Tabelle

Kommando	Adresse	Kommentar
U1 oder UA		BLOCK-READ-Ersatz
U2 oder UB		BLOCK-WRITE-Ersatz
U3 oder UC	$1300	frei verfügbare Anwender-Adressen
U4 oder UD	$1303	"
U5 oder UE	$1306	"
U6 oder UF	$1309	"
U7 oder UG	$130C	"
U8 oder UH	$130F	"
U9 oder UI	$10F0	NMI Vektor
U: oder UJ		POWER UP Vektor

Die Adressen U3 bis U8 liegen im Puffer-Bereich des RAM und können deshalb vom Programmierer mit eigenen Werten belegt werden.

3.2 Verteilung der Blöcke über die Spuren

Spur-Nummer	Block-Bereich
1 bis 39	0 bis 28
40 bis 53	0 bis 26
54 bis 64	0 bis 24
65 bis 77	0 bis 22

3.3 Vorspann des Inhaltsverzeichnisses (Directory Header Block)

Adresse: Spur 39 / Sektor 0

Byte	Inhalt	Definition
0, 1	38, 0	Spur und Sektor des ersten BAM-Blocks
2	67	'C' = 8050-Format
3- 5	0	noch nicht verwendet
6- 21		Disketten-Name (mit SHIFT-SPACEs aufgefüllt)
22, 23	160	SHIFT-SPACEs
24, 25		Disk-ID
26	160	SHIFT-SPACE
27, 28	50, 67	'2C' = DOS-Version und Format-Typ
29- 32	160	SHIFT-SPACEs
33-255	0	nicht verwendet

3.4 Erster BAM-Block

Adresse: Spur 38 / Sektor 0

Byte	Inhalt	Definition
0, 1	38, 3	Spur und Sektor des zweiten BAM-Blocks
2	67	'C' = 8050-Format
3	0	nicht verwendet
4	1	Niedrigste Spur-Nummer, die in diesem BAM-Block vertreten ist
5	51	Höchste Spur-Nummer+1, die in diesem BAM-Block vertreten ist
6		Anzahl ungenutzte Blöcke der Spur 1
7- 10		Bitweises Verzeichnis der verfügbaren Blöcke auf Spur 1
11-255	*	BAM der Spuren 2-50, jeweils 5 Bytes pro Spur

3.5 Zweiter BAM-Block

Adresse: Spur 38 / Sektor 3

Byte	Inhalt	Definition
0, 1	39, 1	Spur und Sektor des ersten Directory-Blocks
2	67	'C' = 8050-Format
3	0	nicht verwendet
4	51	Niedrigste Spur-Nummer, die in diesem BAM-Block vertreten ist
5	78	Höchste Spur-Nummer+1, die in diesem BAM-Block vertreten ist
6		Anzahl ungenutzte Blöcke der Spur 51
7- 10		Bitweises Verzeichnis der verfügbaren Blöcke auf Spur 51
11-140	*	BAM der Spuren 52-77, jeweils 5 Bytes pro Spur
141-255		Ungenutzt

3.6 Struktur der BAM für eine Spur

Byte	Definition
0	Anzahl der verfügbaren Sektoren dieser Spur
1	Bits der Sektoren 0- 7
2	Bits der Sektoren 8-15
3	Bits der Sektoren 16-23
4	Bits der Sektoren 24-31

Bit gesetzt = Sektor ist verfügbar
Bit gelöscht = Sektor ist belegt

3.7 Directory-Format

Das Directory belegt die ganze Spur 39. Sektor 0 enthält den Vorspann mit dem Namen der Diskette und ab Sektor 1 stehen die Dateinamen.

Byte	Definition
0, 1	Spur und Sektor des nächsten Directory-Blocks
2- 31	Datei-Eintrag 1
34- 65	Datei-Eintrag 2
66- 97	Datei-Eintrag 3
98-129	Datei-Eintrag 4
130-161	Datei-Eintrag 5
162-193	Datei-Eintrag 6
194-225	Datei-Eintrag 7
226-255	Datei-Eintrag 8

3.8 Format eines Datei-Eintrags

Byte	Definition	
0	0 = gelöscht 1 = SEQ 2 = PRG 3 = USR 4 = REL	Wenn Bit 7 gesetzt ist, wurde die Datei richtig geschlossen. Ist Bit 7 gelöscht, trägt die Datei einen Stern im Inhaltsverzeichnis und ist nicht mehr lesbar.
1, 2	Spur und Sektor des ersten Daten-Blocks	
3-18	Datei-Name mit SHIFT-SPACEs aufgefüllt	
19,20	Nur bei REL: Spur und Sektor des ersten Verwaltungs-Blocks	
21	Nur bei REL: Record-Länge	
22-25	nicht verwendet	
26,27	Spur und Sektor der ersetzenden Datei, wenn 'ERSETZEN' arbeitet	
28,29	Anzahl der Blöcke der Datei (LOW/HIGH)	

3.9 Struktur der Dateitypen SEQ, PRG und USR

Byte	Definition
0, 1	Spur und Sektor des nächsten Blocks oder Byte 1 = 0 und Byte 2 = letztes gültiges Byte dieses Blocks beim letzten Block der Datei
2-256	254 Daten-Bytes

3.10 Daten-Block der REL-Datei

Byte	Definition
0, 1	Spur und Sektor des nächsten Daten-Blocks
2-256	254 Daten-Bytes Leere Records enthalten im ersten Byte $FF und in allen folgenden 0 Teilweise beschriebene Records sind mit 0 aufgefüllt

3.11 Verwaltungs-Block der REL-Datei

Byte	Definition
0, 1	Spur und Sektor des nächsten Verwaltungsblocks
2	Nummer des Verwaltungs-Blocks (0-5)
3	Record-Länge
4, 5	Spur und Sektor des 1. Verwaltungsblocks (Nummer 0)
6, 7	Spur und Sektor des 2. Verwaltungsblocks (Nummer 1)
8, 9	Spur und Sektor des 3. Verwaltungsblocks (Nummer 2)
10, 11	Spur und Sektor des 4. Verwaltungsblocks (Nummer 3)
12, 13	Spur und Sektor des 5. Verwaltungsblocks (Nummer 4)
14, 16	Spur und Sektor des 6. Verwaltungsblocks (Nummer 5)
16-256	Spur- und Sektor-Zeiger für 120 Daten-Blöcke

VIII. DOS-Hilfsroutinen

1. Einführung

Für besondere Anwendungen, in denen die Möglichkeiten nicht ausreichen, die von den seriellen und REL-Dateien geboten werden, stehen DOS-Kommandos zur Verfügung, die die Dateiverwaltung, vor allem das Directory umgehen. Diese Kommandos ermöglichen direkten Zugriff auf die einzelnen Blöcke (Sektoren) der Diskette bzw. auf bestimmte Arbeitsspeicherbereiche des DOS, vor allem auf die Datenpuffer.

Diese Kommandos sind vor allem für erfahrene Systemprogrammierer, zum Teil auch nur für Assembler-Programm-Entwickler gedacht. Nur-BASIC-Programmierer sollten sich nicht mit der Programmierung auf der unteren DOS-Ebene befassen.

1.1 Übersicht über die Kommandos

Bei Direktzugriff ist zu unterscheiden zwischen:

- Diskette (B- R/W/E)
- BAM (B- A/F)
- Puffer (B- P)
- sonstigen DOS-Speicher (M- W/R/E)
- Sprungtabelle (U)

Die Kommandos BLOCK LESEN/SCHREIBEN übertragen Daten von einem Diskettensektor in einen Puffer oder umgekehrt. BLOCK AUSFÜHREN holt ein Maschinenprogramm von einem Diskettensektor in einen Puffer und startet das Programm sofort.

Durch die beiden BAM-Kommandos BLOCK BELEGEN/FREIGEBEN werden Diskettensektoren vor dem Zugriff der normalen Dateiverwaltung geschützt bzw. für diesen Zugriff freigegeben.

Das Kommando PUFFER-ZEIGER kann den Schreib-/Lesezeiger der Pufferverwaltung manipulieren.

Durch SPEICHER LESEN/SCHREIBEN/AUSFÜHREN kann auf beliebige Adressen im DOS-Arbeitsspeicher schreibend oder lesend zugegriffen werden oder es kann ein dort vorhandenes Programm gestartet werden.

Die Sprungtabelle schließlich erlaubt einen einfachen Aufruf von vorhandenen DOS-Routinen oder eigenen Maschinenprogrammen.

1.2 Übersicht über die Direkt-Zugriffs-Kommandos

Kommando	Abkürz.	Format
BLOCK-READ	B-R	"B-R";KA;LW;SP;SE
BLOCK-READ	U1	"U1";KA;LW;SP;SE
BLOCK-WRITE	B-W	"B-W";KA;LW;SP;SE
BLOCK-WRITE	U2	"U2";UA;LW;SP;SE
BLOCK-EXECUTE	B-E	"B-E";UA;LW;SP;SE
BUFFER-POINTER	B-P	"B-P";KA;PZ
BLOCK-ALLOCATE	B-A	"B-A";LW;SP;SE
BLOCK-FREE	B-F	"B-F";LW;SP;SE
MEMORY-WRITE	M-W	"M-W/adl/adh/ab/bytes"
MEMORY-READ	M-R	"M-R/adl/adh"
MEMORY-EXECUTE	M-E	"M-E/adl/adh"
USER	U	"Ui :parms "

Wobei folgende Abkürzungen gelten:

KA	Kanalnummer
LW	Laufwerk
SP	Spur
SE	Sektor
PZ	Pufferzeiger
adl	Adresse Low Byte
adh	Adresse High Byte
ab	Anzahl Bytes
bytes	Bytefolge
i	Index in die USER-Tabelle
parms	Parameter

2. Behandlung des Direkt-Zugriffs

Für die folgenden Kapitel sollte man sich das Schema der logischen Floppy-Verbindungen parallel anschauen. Damit hat man die beschriebenen Verbindungen anschaulich vor sich.

Man sollte sich nochmal klarmachen, daß der Rechner nur auf den Pufferspeicher direkt zugreifen kann. Für den Transfer Pufferspeicher - Diskette ist das DOS zuständig.

2.1 Schreibzugriff auf Diskette

Will man Daten auf die Diskette schreiben, dann kann das nur in zwei Stufen geschehen. Der Benutzer muß die Daten in den Speicher des DOS direkt schreiben Will er diese Daten aus dem DOS-Speicher auf die Diskette schreiben, muß er dem DOS den entsprechenden Befehl schicken. Das DOS schreibt dann den Inhalt des angegebenen Pufferspeichers in einen bestimmten Block auf der verlangten Diskette.

Übersicht über die Vorgehensweise:

Im folgenden ist vorausgesetzt, daß der Kommandokanal geöffnet ist.

a. Öffnen eines Kanals auf einen Puffer für direkten Datentransfer in den Pufferspeicher des Floppy.

b. Schreiben der Daten vom Rechner in den Floppy-Puffer durch PRINT§ .

c. Kommando an das Floppy, den Inhalt des Puffers auf Disk zu schreiben (B-W).

2.2 Lesezugriff auf Diskette

Analog zum Schreibzugriff können die Daten nur aus dem Pufferspeicher des DOS gelesen werden. Deshalb muß vorher dem DOS das Kommando geschickt werden, den verlangten Block von der Disk in den Speicher zu lesen.

Übersicht über die Vorgehensweise:

a. Öffnen eines Kanals auf einen Puffer für direkten Datentransfer aus dem Pufferspeicher des Floppy (OPEN ...).

b. Kommando an das Floppy, den Inhalt eines Blocks auf der Disk in den Pufferspeicher zu lesen (B-R).

c. Lesen der Daten aus dem Puffer in den Rechner (INPUT§ oder GET§).

Die Kommandos, mit denen solche Zugriffe organisiert werden können, werden nachfolgend besprochen.

3. Direkt-Zugriff-Kommandos auf Diskette

3.1 OPEN: Öffnen eines Puffers

Durch OPEN wird über einen Kanal ein Puffer geöffnet, auf den dann mittels Blockkommandos zugegriffen wird. Bevor eines der folgenden Blockkommandos ausgeführt werden kann, muß ein Puffer geöffnet werden.

3.1.1 Format

Das Format ist im Rechner-Handbuch beschrieben.

Wichtig ist der Abschnitt über die Sekundäradresse SA. Diese gibt den Datenkanal im Floppy an, über den auf den Puffer zugegriffen wird. Dafüt können die Kanäle 2 bis 14 verwendet werden.

Wird versucht, dem Kanal 15 einen Puffer zuzuordnen, so wird die Fehlerlampe am Floppy angeschaltet und bei Abfrage die Zustandsmeldung SYNTAX ERROR übertragen.

Als Dateiname muß ein '§' (Doppelkreuz) angegeben werden. Dies bedeutet, daß ein **Kanal** einem **Puffer** zugeordnet wird: Der Puffer wird "geöffnet"!

Wird **nur** ein Doppelkreuz angegeben, so wird der nächste freie Puffer vom DOS zugeteilt. Durch eine Zahl dahinter kann ein bestimmter Puffer ausgewählt werden.

Ist ein verlangter Puffer nicht verfügbar, wird die Fehlermeldung NO CHANNELS geschickt. Dies kommt z.B. vor, wenn auf die Puffer 12 bis 15 zugegriffen werden soll. Diese sind schon vom DOS belegt.

Beispiele:

OPEN 2,8,5,"§" ordnet dem Kanal 5 den ersten verfügbaren Puffer zu. Vom Rechner her wird auf diesen Kanal bei späteren Kommandos über die Logische Adresse 2 zugegriffen.

OPEN 3,8,6,"§10" ordnet dem Kanal 6 den Puffer 10 zu.

Die Nummer des zugewiesenen Puffers kann über ein 'GET§' Statement auf die Logische Adresse, auf die der Puffer geöffnet wurde, abgefragt werden. Dieses 'GET§' muß direkt nach dem 'OPEN'auf den Puffer folgen, also bevor eine Schreib- oder Leseoperation mit diesem Puffer durchgeführt wurde. Das dabei übertragene Byte stellt die Puffernummer dar. Es muß vom Programm mit ASC ( ) übersetzt werden.

Beispiel:

```
OPEN 2,8,6,"§"
GET§ 2,G$ : IF G$ = "" THEN G$ = CHR$(0)
PRINT "PUFFERNUMMER" ASC(G$)
```

3.2 CLOSE: Schließen des Puffers

Der CLOSE-Befehl schließt den eröffneten Kanal und schreibt die BAM auf die Diskette, der der Kanal zuletzt zugeordnet war.

Es sollte immer nur **ein** Laufwerk gleichzeitig für Direktzugriffskanäle verwendet werden, da sonst leicht ein Durcheinander entstehen könnte. Wenn nämlich vor dem 'CLOSE' auf eine andere Diskette zugegriffen wurde, wird die falsche BAM auf diese Diskette geschrieben. Damit kann der Inhalt von Disketten zerstört werden. Müssen beide Laufwerke benützt werden, sollte unbedingt für jedes Laufwerk ein eigener Kanal geöffnet werden.

Das Format des CLOSE-Befehls ist im Rechner-Handbuch beschrieben.

3.3 "B-R" Block lesen

Das Kommando "B-R" (**B**lock **R**ead) veranlaßt das DOS, den Inhalt eines bestimmten Blocks von der **Diskette** in einen **Puffer** zu **lesen.**

Das Kommando benötigt folgende Parameter in der Reihenfolge:

KA Kanalnummer, über die der Puffer geöffnet wurde
LW Laufwerksnummer der angesprochenen Diskette
SP Spur des benötigten Blocks
SE Sektor des benötigten Blocks

Die Parameter müssen als numerischer Ausdruck dem Kommando direkt folgen.

Beispiel:

Lies den Block auf Laufwerk 0, Spur 15, Sektor 10 in den Puffer des Kanals Nr. 4

```
PRINT§ 15 , "B-R" ; 4 ; 0 ; 15 ; 10
```

oder

```
KA = 4
LW = 0
SP = 15
SE = 10
K$ = "B-R"
PRINT 15 , K$ ; KA ; LW ; SP ; SE
```

Nach Ausführung des Kommandos 'B-R' wird das Byte Nr. 0 des Blocks vom DOS als Zeiger gemerkt. Die Stelle im Puffer, die durch dieses Byte angegeben wird, ist dann beim Senden zum Rechner ('INPUT§' oder 'GET§') ein Endekriterium (siehe Pufferzeiger).

Durch diese Reaktion des DOS ergibt sich, daß man nicht 255 Zeichen aus dem Puffer einlesen kann.

Man kann diese Einschränkung übergehen, indem man statt dem Kommando 'B-R' das Kommando **'U1'** verwendet.

Nach dem Kommando **'U1'** merkt sich das DOS nicht den Inhalt des Bytes 0, sondern setzt stattdessen den Wert auf 255. Dadurch wird beim Einlesen vom Rechner nicht durch diesen gemerkten Wert unterbrochen, sondern nur durch 'Carriage Return' (CR) oder nach dem 255sten Byte.

3.4 "B-W" Block schreiben

Das Kommando "B-W" (**B**lock **W**rite) veranlaßt das DOS, den Inhalt des angegebenen Puffers in einen bestimmten Block auf die Diskette zu **schreiben.**

Das Kommando benötigt folgende Parameter in der Reihenfolge:

KA Kanalnummer, über die der Puffer geöffnet wurde
LW Laufwerksnummer der angesprochenen Diskette
SP Spur des benötigten Blocks
SE Sektor des benötigten Blocks

Die Parameter müssen als numerischer Ausdruck dem Kommando direkt folgen.

Beispiel:

Schreibe den Inhalt des Puffers, der über Kanal 5 geöffnet wurde, in den Block Spur 11, Sektor 14 der Diskette in Laufwerk 1.

```
PRINT§ 15 , "B-W" ; 5 ; 1 ; 11 ; 14
```

oder

```
KA = 5
LW = 1
SP = 11
SE = 14
K$ = "B-W"
PRINT§ 15 , K$ ; KA ; LW ; SP ; SE
```

Bei Ausführung des Kommandos "B-W" wird in das nullte Byte des Puffers der aktuelle Pufferzeiger geschrieben (siehe Pufferzeiger). Dadurch wird also das Byte Nr. 0 vom DOS selbständig verändert.

Dies kann man vermeiden, indem man stattdessen das Kommando 'U2' verwendet. Die Syntax für dieses Kommando ist dieselbe wie bei "B-W".

'U2' verändert also nicht das nullte Byte im Puffer vor dem Schreiben auf die Diskette.

3.5 "B-E" Block ausführen

Das Kommando "B-E" (Block Execute) veranlaßt das DOS, einen bestimmten Block von der Diskette in einen Puffer zu lesen und den Inhalt dieses Puffers sofort als Maschinenprogramm auszuführen.

Das Kommando benötigt folgende Parameter in der Reihenfolge:

KA Kanalnummer, über die der Puffer geöffnet wurde
LW Laufwerksnummer der angesprochenen Diskette
SP Spur des benötigten Blocks
SE Sektor des benötigten Blocks

Die Parameter müssen als numerischer Ausdruck dem Kommando direkt folgen.

Beispiel:

Lies den Block auf Laufwerk 0, Spur 15, Sektor 10 in den Puffer des Kanals Nr. 4 und führe den Inhalt, beginnend mit der Position 0 des Puffers als Maschinenprogramm aus.

```
      PRINT§ 15 , "B-E" ; 4 ; 0 ; 15 ; 10

oder  KA = 4
      LW = 0
      SP = 15
      SE = 10
      K$ = "B-E"
      PRINT§ 15 , K$ ;KA ; LW ; SP ; SE
```

4. Direkt-Zugriff-Kommandos in die BAM

Mit diesen Kommandos kann der Programmierer bestimmte Blöcke in der BAM als belegt bzw. als frei kennzeichnen.

Bei der Ausführung dieser Kommandos wird die BAM entsprechend geändert. Dabei wird die Version der Tabelle geändert, die im DOS-Speicher steht. Diese BAM-Version wird erst nach dem CLOSE-Kommando eines Direkt-Zugriffskanals, nach dem 'CLOSE' einer Schreib-Datei oder 'REL'-Datei oder 'SAVE' (allg. schreibender Zugriff) auf die Diskette zurückgeschrieben.

Diese Kommandos benötigen keinen offenen Direkt-Zugriffskanal, da bei der Ausführung kein Datentransfer über einen solchen Kanal stattfindet. Wenn jedoch die BAM auf der Diskette geändert werden soll, sollten diese Kommandos in Verbindung mit einem Direkt-Zugriffkanal und mit den Block-Kommandos benutzt werden.

4.1 "B-A" Block belegen

Das Kommando "B-A" (**B**lock **A**llocate) veranlaßt das DOS, in der BAM den angegebenen Block als belegt zu kennzeichnen.

Das Belegen eines Blocks bewirkt, daß dieser bei späteren sequentiellen Operationen nicht mehr benützt wird. Damit steht er nur noch für Direkt-Zugriffs-Operationen zur Verfügung.

Wenn der gewünschte Block bereits vorher belegt worden war, zeigt der Fehlerkanal die Fehlermeldung NO BLOCK. Gleichzeitig wird in den zwei Zahlen hinter NO BLOCK der nächste verfügbare Block angegeben, der ab dem verlangten mit aufsteigender Spur- und Sektornummer gefunden wird.

Wenn kein Block über dem gewünschten mehr verfügbar ist, werden Nullen als Spur- und Sektornummer zurückgegeben.

Das Kommando benötigt folgende Parameter in der Reihenfolge:

LW Laufwerknummer der angesprochenen Diskette
SP Spur des gewünschten Blocks
SE Sektor des gewünschten Blocks

Die Parameter müssen als numerischer Ausdruck dem Kommando direkt folgen.

Beispiel:

Kennzeichne den Block Spur 11, Sektor 14 der Diskette in Laufwerk 1 als belegt.

```
      PRINT§ 15 , "B-A" ; 1 ; 11 ; 14

oder  LW = 1
      SP = 11
      SE = 14
      K$ = "B-A"
      PRINT§ 15 , K$ ; LW ; SP ; SE
```

4.2 "B-F" Block freigeben

Das Kommando "B-F" (Block Free) veranlaßt das DOS, in der BAM den angegebenen Block als frei zu kennzeichnen.

Wenn eine Diskette gemischt mit sequentiellem und direktem Zugriff behandelt wird, muß dieses Kommando mit großer Vorsicht gehandhabt werden. Die Blöcke, die von sequentiellen Dateien benützt werden, dürfen auf keinen Fall freigegeben werden, sonst werden solche Dateien bei späteren Schreiboperationen zerstört, da die Blöcke wieder verwendet werden können.

Das Kommando benötigt folgende Parameter in der Reihenfolge:

LW Laufwerknummer der angesprochenen Diskette
SP Spur des gewünschten Blocks
SE Sektor des gewünschten Blocks

Beispiel:

Kennzeichne den Block Spur 3, Sektor 15 auf der Diskette in Laufwerk 1 als frei.

```
PRINT§ 15 , "B-F" ; 1 ; 3 ; 15

oder  LW = 1
      SP = 3
      SE = 15
      K$ = "B-F"
      PRINT§ 15 , K$ ; LW ; SP ; SE
```

5. Direkt-Zugriff-Kommandos in den Puffer

5.1 "B-P" Puffer-Zeiger setzen

5.1.1 Erklärung des Puffer-Zeigers

Das DOS verwaltet für jeden Puffer einen sogenannten **Puffer-Zeiger.**

Es gibt zwei Auswirkungen dieses Zeigers, die unabhängig voneinander auf Lese- und Schreiboperationen wirken.

Schreiben in den Puffer:

Beim Schreiben in den Puffer wird der Zeiger auf das Byte gesetzt, das dem letzten geschriebenen Byte folgt. Das bewirkt, daß Daten, die nacheinander in den Puffer geschrieben werden, im Puffer auch hintereinander abgespeichert werden.

Setzt man den Zeiger durch ein Kommando auf einen bestimmten Wert, so werden die Daten, die dann in den Puffer geschrieben werden, ab der Zeigerposition abgespeichert. Dadurch soll der Benutzer die Möglichkeit erhalten, die Abspeicherposition von Daten selbst zu bestimmen.

Wird der Inhalt des Puffers mit dem Kommando "B-W" auf die Diskette geschrieben, dann wird der aktuelle Pufferzeiger in Position Null des Blocks geschrieben. Dieser Zeiger, der auf Diskette geschrieben worden ist, wird im weiteren als **Schreibzeiger** bezeichnet. Durch das Kommando "U2" wird das Byte Null nicht verändert.

Will man den Puffer-Zeiger durch ein Zeiger-Kommando auf den Wert 0 oder 255 setzen, wird er durch das DOS auf den Wert 1 gesetzt.

Lesen aus dem Puffer:

Beim Lesen aus dem Puffer gibt es zwei Zeiger, die verschieden wirken:

- **Der Schreibzeiger** (gilt nur bei Verwendung des Kommandos "B-R").

Dieser wurde beim Abspeichern des Puffers mit "B-W" in Position Null des Blocks geschrieben. (Siehe oben).

Er bewirkt, daß beim Einlesen vom Puffer mit 'INPUT§' der Datentransfer nach dieser Stelle unterbrochen wird. Dieser Zeiger soll also das Ende der gültigen Information in dem Block kennzeichnen.

Beim Lesen mit 'GET§' wird das Byte, das nach dem durch den Schreibzeiger angegebenen folgt, nicht richtig eingelesen. Stattdessen kommt der Wert, der in Position 1 des Blocks steht.

- **Der Lesezeiger**

Dieser Zeiger kann durch ein Kommando gesetzt werden, wird aber auch durch fortlaufendes Lesen aus dem Block hochgezählt.

Wie der Name andeuten soll, wird beim Lesen aus dem Puffer ab der Position gelesen, die durch den Lesezeiger angegeben wird.

5.1.2 "B-P" Puffer-Zeiger setzen

Das Kommando "B-P" (Buffer Pointer) veranlaßt das DOS, den Puffer-Zeiger des angegebenen Puffers auf den entsprechenden Wert zu setzen.

Das Kommando benötigt folgende Parameter in der Reihenfolge:

KA Kanalnummer, über die der Puffer geöffnet wurde
PZ Pufferzeigerposition

Beispiel:

Setze den Puffer-Zeiger des Puffers von Kanal 4 auf den Wert 12.

```
PRINT§ 15 , "B-P" ; 4 ; 12
```

oder
```
KA = 4
PZ = 12
K$ = "B-P"
PRINT§ 15 ,K$ ; KA ; PZ
```

Nach der Ausführung dieses Kommandos werden :

- bei Lese-Operationen die Daten ab der Position 12 gelesen
- bei Schreib-Operationen Daten ab der Position 12 abgespeichert.

5.2 PRINT§ Schreiben in den Puffer

Dieser BASIC-Befehl ist im Rechner-Handbuch erklärt.

Mit diesem Befehl können Daten in einen Puffer geschrieben werden. Dabei sind dieselben Regeln zu beachten wie beim Schreiben in eine Datei vom Typ SEQ.

Die angegebenen Daten werden ab der Stelle im Puffer abgespeichert, auf die der aktuelle Puffer-Zeiger zeigt. Der Puffer-Zeiger wird nach dem Datentransfer auf die Stelle gesetzt, die dem letzten übertragenen Byte folgt.

Beispiel:

Schreibe den Inhalt der Stringvariablen P$ in den Puffer, der mit der Logischen Adresse 3 geöffnet worden ist. Dahinter soll kein Delimiter geschrieben werden.

```
PRINT§ 3 , P$ ;
```

5.3 GET§ Lesen aus dem Puffer

Dieser BASIC-Befehl ist im Rechner-Handbuch erklärt.

Beim Lesen aus einem Puffer wird das Zeichen, auf das der aktuelle Lesezeiger zeigt, aus dem Puffer in eine Variable übertragen.

Das Zeichen hinter dem Byte, auf welches der Schreibzeiger zeigt, wird nicht mit dem richtigen Wert übertragen. Dafür wird der Wert des Bytes Nr. 1 vom Floppy gesendet.

Beispiel:

Lies das Byte Nr. 50 aus dem Puffer, der über Kanal 3 und die Logische Adresse 5 geöffnet wurde.

```
PRINT§ 15 , "B-P" ; 3 ; 50
GET§ 5, G$
```

Danach steht der Inhalt des Bytes Nr. 50 in G$ und der Lesezeiger (Pufferzeiger) zeigt auf Position 51.

5.4 INPUT§ Lesen aus dem Puffer

Dieser BASIC-Befehl ist im Rechner-Handbuch erklärt.

Dabei werden die Daten ab dem aktuellen Lesezeiger in die gewünschte Basic-Variable übertragen.

Für das Einlesen in die Variable gelten dieselben Trennzeichen wie sie in 3.6 angegeben sind. Zusätzlich wirkt der **Schreibzeiger** als Endekriterium für die Datenübertragung. Bei Verwendung von "U1" muß dieser allerdings nicht berücksichtigt werden.

Es werden also folgende Daten aus dem Puffer in die Variable übertragen:

Beginn: Position des Lesezeigers

Ende: CR (Carriage Return) = CHR$(13)
',' (Komma) = CHR$(44)
':' (Doppelpunkt) = CHR$(58)
Position des Schreibzeigers

Dabei wird beim Erreichen der Schreibzeiger-Position, oder eines CR der Datentransfer zum Rechner unterbrochen.

Wird vor einem 'INPUT§' der Lesezeiger hinter dem Schreibzeiger aufgesetzt, kann der Rechner "sterben", wenn in demselben Block kein Delimiter mehr steht.

Beim Entwickeln des Datenformats in einem Block muß der Programmierer folgendes beachten:

Wie in 3.6 erklärt wurde, können mit einem normalen 'INPUT§' maximal 80 Zeichen eingelesen werden. Wenn in einem Datensatz mehr als 80 Zeichen stehen, muß bei Verwendung des normalen 'INPUT§' spätestens nach jeweils 80 Zeichen ein Carriage Return (ASC 13) stehen.

Beispiel:

Lies die Daten ab Byte Nr. 10 aus dem Puffer, der über Kanal 12 und die Logische Adresse 5 geöffnet wurde.

```
PRINT§ 15 , "B-P" ; 12 ; 10
INPUT§ 5 , P$
```

Danach steht in P$ der Inhalt des Puffers ab Byte Nr. 10 bis zum nächsten Trennzeichen.

5.5 Schreiben und Lesen über denselben Puffer

Macht man über denselben Puffer Schreib- und Lesezugriffe auf Diskette, ergibt sich folgender DOS-Fehler:

Folgt auf einen Schreibzugriff auf eine Diskette sofort eine Lesezugriff, dann wird dieser nicht richtig ausgeführt. Bei einem 'INPUT' in den Puffer wird nämlich nur ein Byte übertragen, also nicht bis zum nächsten Delimiter. Dieser Fehler tritt sowohl bei Verwendung von "B-R" als auch von "U1" auf.

Beispiel:

```
OPEN 2,8,2 "§"
   .
   .
PRINT§ 15,"U2",KA,LW,SP,SE      Schreibzugriff
PRINT§ 15,"U1",KA,LW,SP,SE      Lesezugriff
PRINT§ 15,"B-P",KA,1            Pufferzeiger setzen (Lesezeiger)
INPUT§ 2,A$                     hier wird nur ein Byte übertragen
```

Abhilfe:

Man sollte generell für Schreib- und Lesezugriff verschiedene Puffer verwenden. Es muß also im Programm je ein Direktzugriffskanal für Schreiben und für Lesen geöffnet werden.

6. Direkt-Zugriff-Kommandos in den DOS-Speicher

Mit den folgenden Kommandos kann der Benutzer direkt auf den Speicher des Floppys zugreifen.

Dadurch ergeben sich unter anderem folgende Möglichkeiten:

Man kann die Pufferbereiche des DOS direkt lesen. Der Programmierer hat damit ein Hilfsmittel, um den Disknamen, ID und die BAM mit schnellem Zugriff zu lesen.

Für die Ausführung von Hilfsroutinen kann ein Prozessor des Floppys verwendet werden. Die Routinen werden im Speicher des DOS abgelegt und laufen ab, während im CBM ein anderes Programm läuft.

Unterroutinen des DOS können aufgerufen werden und laufen parallel zum Programm im cbm.

Parameter:

Die Parameter müssen hinter dem Kommando als **String**-Variable oder Konstante folgen. Die Werte der Parameter müssen also durch den Befehl CHR$( ) in Strings mit dem entsprechenden Kode umgewandelt werden.

6.1 "M-W" In den Speicher schreiben

Dieses Kommando veranlaßt das DOS, Daten in bestimmte Bereiche des DOS-Speichers abzulegen. Man kann maximal 34 Bytes mit einem Aufruf des Kommandos abspeichern.

Es sind folgende **Parameter** in der Reihenfolge notwendig:

1. **Speicheradresse,** ab der die angegebene Bytefolge abgespeichert werden soll. Diese Adresse muß in zwei Bytes aufgeteilt werden (höherwertiger und niederwertiger Teil). Die Bytes werden in der Reihenfolge

   CHR$(niederwertigem Byte) CHR$(höherwertigem Byte)

benötigt.

2. **Anzahl Bytes,** die mit diesem Schreib-Kommando geschickt werden.

3. Angegebene Anzahl Bytes als **Bytefolge,** die ab der obigen Speicheradresse abgespeichert werden sollen.

Beispiel:

Es sollen die fünf Bytes (dezimal) 44,59,96,32,78 ab der Adresse 4620 abgespeichert werden.

Die Adresse 4620 muß in zwei Teile zerlegt werden. Das ergibt

```
INT (4620 / 256)   = 18   (höherwertiger Teil)
4620 - 18 * 256    = 12   (niederwertiger Teil)
```

Das Kommando kann also wie folgt aussehen:

```
PRINT§ 15, "M-W"  CHR$(12) CHR$(18) CHR$( 5) CHR$(44) CHR$(59)
                  CHR$(96) CHR$(32) CHR$(78)

                        oder

B$ =  CHR$(12) + CHR$(18) + CHR$(5) + CHR$(44) + CHR$(59)
   +  CHR$(96) + CHR$(32) + CHR$(78)
K$ = "M-W"
PRINT$15 , K$ B$
```

6.2 "M-R" Aus dem Speicher lesen

Dieses Kommando zeigt auf das Byte, das durch die Adresse hinter dem Kommando bestimmt wird. Ab diesem Byte kann nach dem Kommando "M-R" durch 'GET§' oder 'INPUT§' vom Kommandokanal gelesen werden.

Parameter:

1. Die **Speicheradresse** des ersten zu lesenden Bytes. Diese Adresse muß in zwei Bytes aufgeteilt werden (höherwertiger und niederwertiger Teil). Die Bytes werden in der Reihenfolge

CHR$(niederwertiges Byte) CHR$(höherwertiges Byte)

benötigt.

2. Die **Anzahl Bytes,** die eingelesen werden soll, wird durch ein Zeichen angegeben. Es sind also maximal 255 Bytes möglich.

Beim Lesen mit INPUT§ ist zu beachten, daß INPUT§ beim Auftreten eines Kodes 0 oder 13 abbricht. Tritt in dem zu lesenden Speicherbereich dieser Kode auf, wird nicht die gewünschte Anzahl Zeichen eingelesen. Ansonsten wird die richtige Anzahl eingelesen, auch wenn als letztes Zeichen kein CR kommt.

Beispiel:

Es sollen 10 Bytes ab der Speicheradresse 4620 gelesen werden.

Die Adresse 4620 muß in zwei Teile zerlegt werden. Das ergibt

```
INT (4620 / 256)    = 18    (höherwertiger Teil)
4620 - 18 * 256     = 12    (niederwertiger Teil)
```

Das Einlesen kann also wie folgt aussehen:

```
PRINT§ 15, "M-R" CHR$(12) CHR$(18) CHR$(10)
INPUT§ 15, G$

oder

B$ = CHR$(12) + CHR$(18) + CHR$(10)
K$ = "M-R"
PRINT§ 15, K$B$
INPUT§ 15, G$
```

Danach stehen in G$ der Inhalt des Speichers von 4620 bis 4629.

6.3 "M-E" Speicherroutinen ausführen

Dieses Kommando veranlaßt das DOS, den Speicherinhalt ab der angegebenen Adresse als Maschinenroutine auszuführen.

Damit können Routinen ausgeführt werden, die der Benutzer vorher in den DOS-Speicher geschrieben hat oder die fest im DOS stehen. Diese Routinen müssen mit dem Maschinenbefehl RTS (Return from Subroutine $60) abschliessen, da sonst das DOS "abstürzt" und nicht mehr definiert angesteuert werden kann.

Als Parameter ist die Speicheradresse des zu lesenden Bytes notwendig. Diese Adresse muß in zwei Bytes aufgeteilt werden (höherwertiger und niederwertiger Teil). Die Bytes werden in der Reihenfolge

CHR$(niederwertiges Byte) CHR$(höherwertiges Byte)

benötigt.

Beispiel:

Führe den Code ab der Speicheradresse 4620 aus.

Die Adresse 4620 muß in zwei Teile zerlegt werden. Das ergibt

```
INT (4620 / 256)   = 18   (höherwertiger Teil)
4620 - 18 * 256    = 12   (niederwertiger Teil)
```

Das Kommando kann also wie folgt aussehen:

```
PRINT§ 15, "M-E" CHR$(12) CHR$(18)
```

oder

```
B$ = CHR$(12) + CHR$(18)
K$ = "M-E"
PRINT§ 15, K$B$
```

6.4 "U" Maschinenroutine über Sprungliste

Mit "U" (USER) kann der Anwender eine Verbindung zu Maschinenroutinen über eine Sprungliste herstellen. Auf diese Sprungliste wird durch einen Zeiger gezeigt. Dieser Zeiger steht in $00DD oder 221 im DOS-Speicher. Das zweite Zeichen im Kommando wird zur Indizierung der Tabelle benutzt. Dazu können die ASCII-Zeichen 0 (Null) bis 9 und A bis O verwendet werden.

In einer Sprungliste sind bis zu 15 Eintragungen möglich. Die einzelnen Eintragungen werden über das Indexzeichen angesprochen. Dabei wirken Ziffern und Buchstaben in gleicher Weise. "U1" und "UA" springen zum Beispiel beide an die Speicheradresse, die in den ersten beiden Bytes der Sprungliste steht.

Diese Standardtabelle ist unten aufgeführt.

Das Kommando "U0" setzt den Anwender-Zeiger auf die Standard-Sprungliste, die die Verbindung zu speziellen Routinen enthält, die unten aufgeführt sind. Der Standardzeiger hat den Wert 65514.

Der Benutzer kann mit Speicher-Schreibbefehlen den Zeiger auf einen anderen Wert setzen. Dann können über die USER-Kommandos Routinen angesprungen werden, die selbst durch den Benutzer definiert sind.

Beispiel:

Setze den Anwenderzeiger auf die Standardsprungliste des DOS und springe auf die Speicheradresse, die als dritte in der Liste angegeben ist.

```
PRINT§ 15,"U0"
PRINT§ 15,"U3"   oder PRINT§ 15,"UC"
K$ = "M-E"
PRINT§ 15, K$B$
```

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

Commodore AG
Aeschenvorstadt 57
CH-4010 Basel

Commodore GmbH
Fleschgasse 2
A-1130 Wien

81f-183