

**FLOPPY DISK-
HANDBUCH
cbm 3040**



Dieses Handbuch wurde gescannt, bearbeitet und ins PDF-Format konvertiert von

Rüdiger Schuldes

schuldes@itsm.uni-stuttgart.de

(c) 2003

Inhaltsverzeichnis

0.	Einführung	5
0.0.	Allgemeine Informationen	5
0.0.1.	Wozu ein Floppy Disk Drive?	5
0.0.2.	Beschreibung	5
0.1.	Zunächst einmal auspacken	6
0.1.1.	Inhalt	6
0.1.2.	Überprüfung	6
0.2.	Wie schließen Sie Ihre Floppy an?	7
0.2.1.	Verbindung der Floppy mit dem Rechner	7
0.2.2.	Der Anschluß	7
0.2.3.	Test bei der Inbetriebnahme	7
0.3.	So sollten Sie Ihre Disketten behandeln	8
0.4.	... und so einlegen	8
0.4.1.	Die Versicherung: Datenschutz	8
1.	Allgemeines	
1.0.	Die Verbindung zum Rechner: Der IEC-Bus	9
1.0.0.	Allgemeines und Anschlußbelegung	9
1.0.2.	So merken die Peripheriegeräte, was der Rechner von ihnen will	11
1.0.3.	Die Statusvariable ST	11
1.0.4.	Wichtige INPUT-OUTPUT-Adressen im Rechner	11
1.1.	Allgemeines zur Arbeit mit Dateien	
1.1.0.	So öffnen Sie eine Datei (file)	11
1.1.0.1.	Der Kommando-/Fehlerkanal	13
1.1.0.2.	Zwei Dinge auf einmal: Befehle im "OPEN"	13
1.1.0.3.	Zuordnen eines Puffers für den Direktzugriff	13
1.1.0.4.	So schließen Sie eine Datei (der CLOSE-Befehl)	14
1.1.2.	Der Weg vom Computer zur Floppy (der PRINT#-Befehl)	14
1.1.2.0.	Allgemeine und mögliche Fehlermeldungen	14
1.1.2.1.	Wichtige Hinweise zu "PRINT#"	14
1.1.3.	So holen Sie Ihre Daten wieder zurück: Der INPUT#-Befehl	15
1.1.3.0.	Allgemeines	15
1.1.3.1.	Der BIP läuft über	15
1.1.3.2.	Abhilfe bei mehr als 80 Zeichen: GET# oder Maschinenprogramm	15
1.1.3.3.	Die Begrenzungszeichen (Delimiter)	16
1.1.4.	Zurück in kleinen Schritten (der GET#-Befehl)	16
1.1.4.0.	Allgemeines	16
1.1.4.1.	Vergleich von "GET#" und "INPUT#"	16
2.	Dateien und Disketten	
2.0.	Was ist eine Datei?	16
2.0.0.	Die PRG-Datei	17
2.0.1.	Die SEQ-Datei	17
2.0.2.	Der Direktzugriff	17
2.1.	So sind Spuren und Blöcke verteilt	18
2.2.	Das Inhaltsverzeichnis und die BAM	18
2.2.0.	Das Schreiben der BAM bei "CLOSE"	18
2.2.1.	So sehen Sie sich das Inhaltsverzeichnis an	18
2.2.2.	... das kann auch ein Programm für Sie tun	19
3.	Die Floppy wird »rumkommandiert«	21
3.0.	Das müssen Sie bei Dateinamen beachten	22
3.0.1.	Die »Jokerzeichen«	22
3.1.	Zuerst muß die Floppy die Diskette kennen	23
3.2.	Sie verdoppeln eine ganze Diskette: D	23
3.3.	... oder nur eine Datei: C	23
3.4.	Eine Datei wird umbenannt: R	23
3.5.	So entfernen Sie Dateien, die Sie nicht mehr brauchen können: S	24
3.6.	Kontrolle und Bereinigen der Diskette	24
3.7.	Das Ende aller Daten, das Löschen: N	25

4.	So speichern Sie ein Programm	25
4.0	Wozu?	25
4.1.	Das Speichern eines BASIC-Programms: "SAVE"	26
4.2.	... und so bekommen Sie Ihr Programm wieder zurück: "LOAD"	27
4.3.	Sicherheitshalber kontrollieren: "VERIFY"	27
4.4.	So speichern Sie ein Maschinenprogramm: .S	27
4.5.	... und laden es wieder: L	28
5.	Die Datenkette: die SEQ-Datei	28
5.0.	Das müssen Sie sich bei SEQ-Dateien alles überlegen	28
5.1.	Der OPEN-Befehl	28
5.2.	Das Schreiben von Daten in eine SEQ-Datei: Print#	29
5.3.	So lesen Sie von einer SEQ-Datei: INPUT#	29
5.4.	Zurück in kleinen Schritten: GET#	30
6.	DOS-Organisation	30
6.0.	Plan der DOS-Organisation	30
6.1.	Pufferorganisation	32
6.2.	Kanäle	32
7.	Für ganz Eilige und Intelligente: Der Direktzugriff	32
7.0.	Was können Sie damit machen?	32
7.0.0.	Anwendungsgebiete im Vergleich zur SEQ-Datei	32
7.0.1.	Die Spur-Sektor-Einteilung	32
7.0.2.	Beispiel für eine einfache Datei	33
7.0.3.	Der Zugriff mit Verweisdatei	33
7.1.	Die Möglichkeiten des Eingriffs	33
7.2.	Das Vorgehen beim Schreibzugriff	34
7.3.	Der Lesezugriff	34
7.4.	Vorgehen beim Ausführen von Maschinenroutinen durch die Floppy	34
7.5.	Beeinflussung der Verbindung Rechner - Puffer	34
7.5.0.	Öffnen eines Puffers	34
7.5.1.	Das Schließen eines Puffers	35
7.5.2.	Das Schreiben in einen Puffer	35
7.5.3.	Das Lesen aus dem Puffer	35
7.5.4.	Das Lesen von einzelnen Zeichen: GET#	35
7.6.	Der Pufferzeiger	35
7.6.0.	So sieht das B-P Kommando aus	35
7.6.1.	Funktion des Pufferzeigers beim Schreiben	35
7.6.2.	Funktion des Pufferzeigers beim Lesen	35
7.6.3.	Anwendungsmöglichkeiten für "B-P"	36
7.7.	Der Eingriff Puffer-Diskette	38
7.7.0.	Schreiben eines Pufferinhaltsverzeichnisses auf Diskette	38
7.7.1.	Lesen eines Diskettenblocks in einen Puffer	38
7.8.	Direktzugriff in die BAM	39
7.9.	Eingriff in den Floppyspeicher	39
7.9.0.	M-W	39
7.9.1.	M-R	40
7.10.	Ausführung von Maschineroutinen durch die Floppy	40
7.10.0.	B-E	40
7.10.1.	M-E	40
7.11.	USER	40
7.11.0.	Allgemein	40
7.11.1.	Die Standartsprungtabelle	41
7.12.	Die Floppyadressen der Puffer	42
7.13.	Lesen von Diskettenname und ID aus der BAM	42
8.	Fehlermeldungen	
8.0.	Was tun, wenn die Floppy spinnt?	42
8.1.	Tabelle der Fehlermeldungen	43
8.2.	Ursachen, Erklärung und Abhelfen	44

0. Einführung

0.0. Allgemeine Informationen

0.0.1. Wozu ein Floppy Disk Drive?

Sie haben mit dem Commodore 3040 Doppellaufwerk einen externen Speicher mit großer Speicherkapazität erworben, der die Möglichkeiten Ihres cbm-Rechners stark erweitern wird.

Bitte lesen Sie die Einführung vollkommen durch, bevor Sie die Floppy auspacken oder in Betrieb nehmen.

0.0.2. Beschreibung

Die Commodore-Floppy ist ein intelligenter, sich selbst kontrollierender Daten-Massenspeicher, der für den Anschluß an den IEC-Bus Ihres Rechners bestimmt ist.

Die technischen Daten entnehmen Sie bitte der Tabelle 1.1.

Tabelle 1.1.

Gehäuse:

Material:	1,25 mm Stahlblech	{18 ga}
Höhe:	16,5 cm	{6,5"}
Breite:	38,1 cm	{15"}
Tiefe:	36,45 cm	{14,35"}

Stromversorgung:

Spannung:	220 V Wechselstrom
Frequenz:	50 Hz
Leistung:	132 W

IC's:

Controller:

6504	Mikroprozessor
6530	I/O, RAM, ROM
6522	I/O, Zeitgeber

Interface:

6502	Mikroprozessor
6532 (2x)	I/O, RAM, Zeitgeber
6332 (2x)	ROM

Außerdem:

6114 (8x)	4x 1 k RAM
-----------	------------

Laufwerke:

Shugart SA 390 (2x)

Disketten:

Standard Mini 5 1/4"

0.1. Zunächst einmal auspacken

Bevor Sie die Floppy auspacken, überprüfen Sie den Transportkarton bitte auf Beschädigungen. Sollte der Karton beschädigt sein, seien Sie bitte bei der Prüfung des Inhalts besonders vorsichtig und aufmerksam. Entfernen Sie nun vorsichtig das gesamte Verpackungsmaterial und nehmen Sie den Inhalt des Kartons heraus.

Es empfiehlt sich, den Karton und das Verpackungsmaterial für einen eventuellen späteren Transport des Gerätes aufzuheben, da nur in der Spezialverpackung ein antistatischer Transport möglich ist!

0.1.1. Inhalt

Bitte überzeugen Sie sich, daß folgende Positionen vorhanden sind:

- 1.) Modell 3040 Floppy Disk Drive
- 2.) Bedienungsanleitung
- 3.) Garantiekarte

Sollte ein Teil fehlen, so wenden Sie sich bitte an Ihren Commodore-Händler.

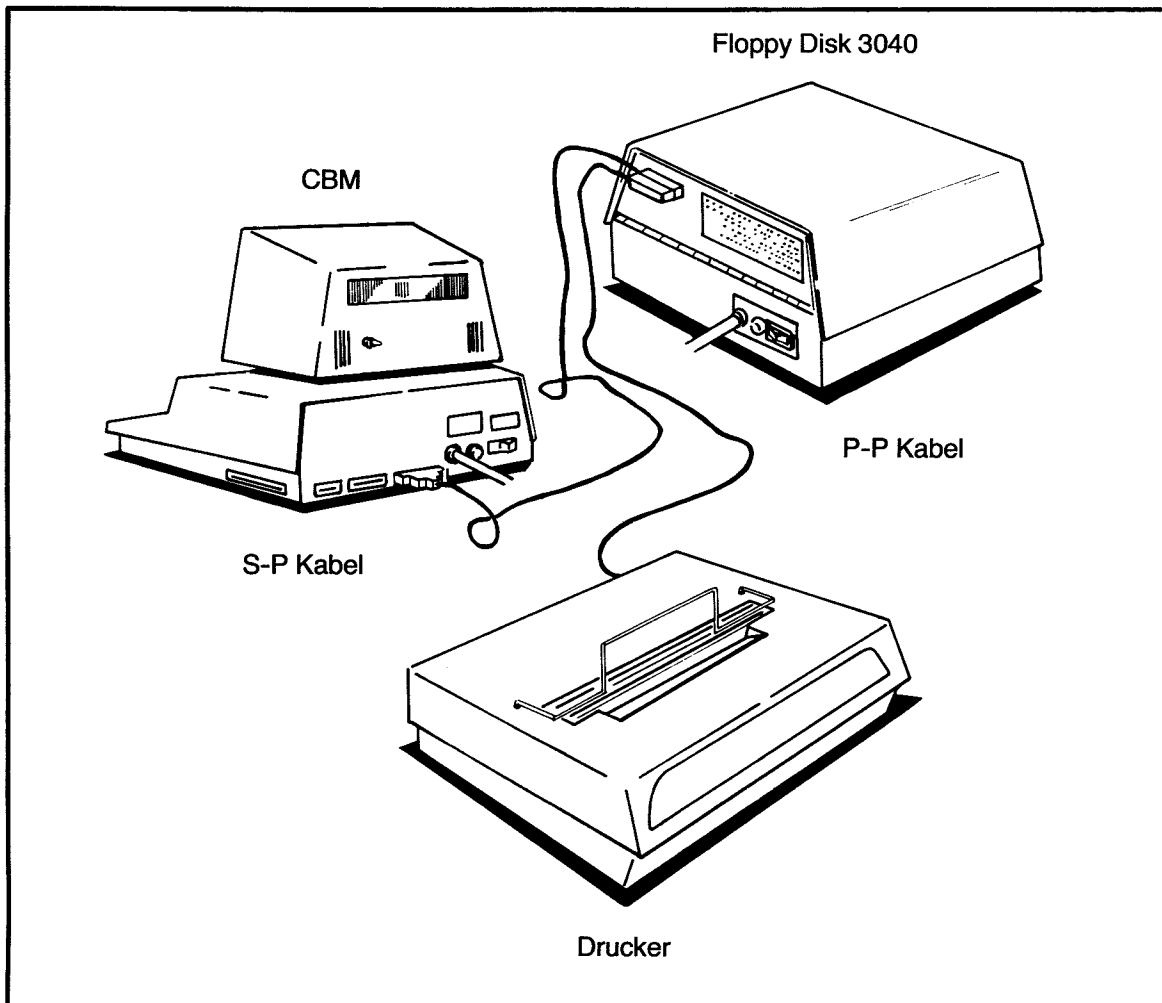


Abbildung 1

0.1.2. Überprüfung

Untersuchen Sie die Floppy bitte auf Zeichen äußerer Beschädigungen (Kratzer etc.). Das Gerät ist vor dem Transport sorgfältig getestet und überprüft worden. Benachrichtigen Sie bitte Ihren Händler, falls Sie eine Beschädigung entdeckt haben!

0.2. Wie schließen Sie Ihre Floppy an?

0.2.1. Verbindung der Floppy mit dem Rechner

Um die Floppy mit dem Rechner zu verbinden, benötigen Sie eines der beiden folgenden Verbindungskabel (Bild 1):

- a.) Rechner nach IEC-Kabel
- b.) IEC nach IEC-Kabel

Das erste Kabel benötigen Sie für Ihr erstes Gerät am IEC-Bus (z. B. Floppy an Rechner).

Das zweite Kabel benötigen Sie für jedes weitere Gerät am IEC-Bus (z. B. Drucker oder weitere Floppys).

Beide Kabel erhalten Sie bei Ihrem Commodore-Händler.

0.2.2. Der Anschluß

Beim Anschluß beachten Sie bitte die folgenden Schritte:

- 1.) Schalten Sie den Rechner ab.
- 2.) Stellen Sie die Floppy in eine geeignete Position so nahe wie möglich an den Rechner.
- 3.) Verbinden Sie das »Rechner nach IEC-Kabel« mit der IEC-Interface Verbindung am Rechner und an der Floppy. Sollten zusätzliche Geräte am IEC-Bus liegen, so muß das »IEC nach IEC-Kabel« verwendet werden.
- 4.) Verbinden Sie nun erst das Netzkabel der Floppy mit der nächsten Steckdose, **SCHALTEN SIE JE-DOCH NOCH NICHT EIN!**

0.2.3. Test bei der Inbetriebnahme

Um den Test durchzuführen, beachten Sie bitte die folgenden Punkte:

- 1.) Verbinden Sie die Floppy mit dem Rechner, wie in Abschnitt 0.2.2. beschrieben.
- 2.) Schalten Sie die Betriebsspannung am Rechner ein und überprüfen Sie, ob der Rechner normal arbeitet.
- 3.) Öffnen Sie beide Laufwerksklappen an der Floppy und stellen Sie bitte sicher, daß sich keine Disketten in den Laufwerken befinden (siehe Bild 2).
- 4.) Schalten Sie nun erst die Betriebsspannung an der Floppy ein (siehe Bild 2).

Sie werden bemerken, daß die beiden Indikatorlampen an den Laufwerken und die Fehlerlampe in der Mitte aufleuchten und nach wenigen Sekunden erlöschen. Keiner der beiden Motoren sollte anlaufen.

Sollte irgendeine Indikatorlampe an der Floppy länger brennen als drei Sekunden, so schalten Sie die Floppy bitte ab, warten Sie 5 Minuten und versuchen Sie es dann noch einmal. Sollten die Lampen immer noch brennen bleiben, wenden Sie sich bitte an Ihren Commodore-Händler!

- 5.) Damit ist der Test bei Inbetriebnahme beendet.

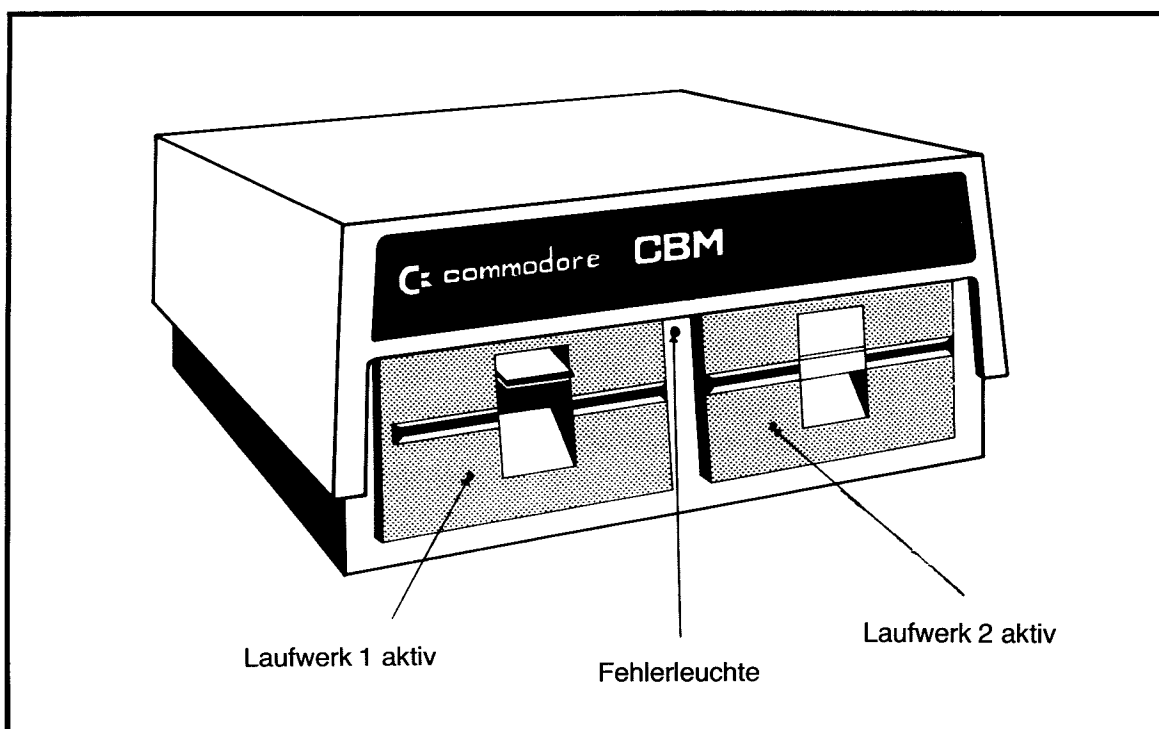


Abbildung 2

0.3. So sollten Sie Ihre Disketten behandeln

An sich sind Disketten recht unempfindliche Datenträger, doch sollten Sie folgende Punkte zur Vermeidung von Schäden oder Fehlern beachten:

- 1.) OBEN ist bei einer Diskette die Seite mit dem Etikett.
- 2.) Auf keinen Fall dürfen die freien Flächen der Diskette in den Öffnungen in der Schutzhülle berührt werden!
- 3.) Um Berührungen der Diskettenoberfläche zu vermeiden, sollten Sie die Disketten immer nur in der Schutzhülle aufbewahren.
- 4.) Die Disketten sollten vor Staub geschützt werden, da sonst sowohl die Disketten als auch die Schreib-Leseköpfe angegriffen werden.
- 5.) Disketten dürfen, wie Kassetten, nicht starken Magnetfeldern ausgesetzt werden, da sonst die Daten verändert oder zerstört werden können.
- 6.) Während des Ein- und Ausschaltens des Gerätes sollten die Disketten aus den Laufwerken genommen werden, auf keinen Fall aber die Klappen der Laufwerke geschlossen sein, damit die Schreib-Leseköpfe nicht auf den Disketten aufliegen.

0.4. und so einlegen

Wenn Sie eine Diskette in ein Laufwerk einlegen wollen, beachten Sie bitte folgende Reihenfolge:

- 1.) Sorgen Sie bitte vor dem Einlegen dafür, daß die Diskette richtig zentriert ist, das heißt, daß der in der Mitte der Diskette sichtbare Teil der Magnetplatte auf allen Seiten des Lochs gleich breit ist.
- 2.) Fassen Sie die Diskette auf der Seite mit dem Etikett an der Schutzhülle an, und schieben Sie sie mit dem Etikett nach oben in das Laufwerk hinein.
- 3.) Schließen Sie das Laufwerk, indem Sie die Laufwerkklappe probeweise andrücken wenn es anläuft und erst dann vollständig schließen.

0.4.1. Die Versicherung: Datenschutz

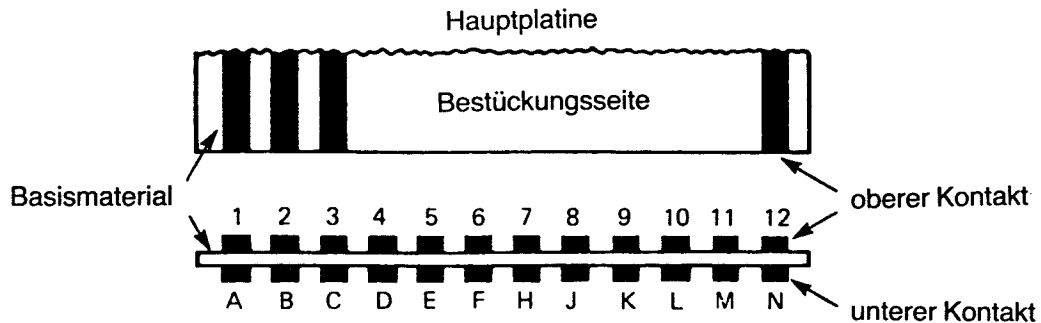
Es ist ratsam, VON JEDER DISKETTE EINE KOPIE ANZUFERTIGEN, damit bei eventuell auftretenden Fehlern wenigstens eine vollständige Version einer Datei oder eines Programms vorhanden ist. Ein Verfahren, das Sie zum Numerieren der verschiedenen Versionen eines Programms verwenden können, ist in der Anleitung Ihres Rechners beschrieben. Bei Dateien können Sie ähnlich vorgehen. Das sicherste Verfahren, das allerdings sehr aufwendig ist, wäre ALLES (also jedes Programm, jede Datei) sofort auf zwei Disketten zu speichern.

1.0. Die Verbindung zum Rechner: Der IEC-Bus

1.0.0. Allgemeines und Anschlußbelegung

Ihr Rechner verfügt über eine genormte IEC-Bus-Schnittstelle. Außer Ihrer Floppy können Sie viele andere Geräte an diesem IEC-Bus verwenden. So zum Beispiel Meßinstrumente, die über Anschlußmöglichkeiten an den IEC-Bus verfügen, oder Sie können über den Bus Drucker oder Steueraggregate betreiben.

Der IEC-Bus verfügt insgesamt über 24 Anschlüsse, die wie folgt belegt sind:



1	DI01	Niederwertigstes Datenbit
2	DI02	Zweitniederwertigstes Datenbit
3	DI03	Drittniederwertigstes Datenbit
4	DI04	Viertniederwertigstes Datenbit
5	EOI	End or identification (Ende oder Identifizierung)
6	DAV	Data valid (Daten gültig)
7	NRFD	Not ready for data (nicht bereit für Daten)
8	NDAC	Data not accepted (Daten [noch] nicht angenommen)
9	IFC	Interface clear (löschen des Busses)
10	SRQ	Service request (Bedienungsaufruf)
11	ATN	Attention (Aufmerksamkeit)
12	GND	Ground (Masse, Chassis und Kabelabschirmung)
A	DI05	Vierthöchstes Datenbit
B	DI06	Dritthöchstes Datenbit
C	DI07	Zweithöchstes Datenbit
D	DI08	Höchstes Datenbit
F	REN	Remote enable (bei Ihrem Rechner stets auf L)
F	GND (DAV)	
H	GND (NRFD)	
J	GND (NDAC)	
K	GND (IFC)	
L	GND(SRQ)	
M	GND (ATN)	
N	GND (DI01-8)	

Bei den Anschlüssen F-N handelt es sich um Erdungsrückleitungen.

Die Übertragung erfolgt nach dem sogenannten »Handshake«-(Händeschütteln-)-Verfahren, kontrolliert durch die drei Leitungen:

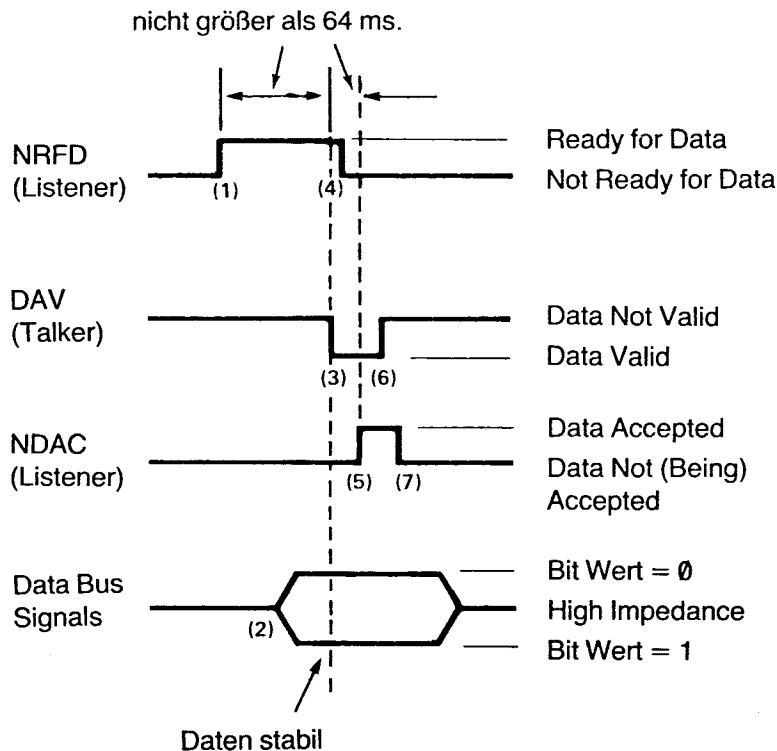
- 1.) DAV (Data valid = Daten gültig)
- 2.) NRFD (Not ready for data = nicht bereit, Daten aufzunehmen)
- 3.) NDAC (Data not accepted = Daten [noch] nicht angenommen)

Das Verfahren der Datenübertragung sei an einem Beispiel erläutert:

Sie wollen Daten vom Rechner auf die Floppy übertragen. Der Rechner ist also Sprecher (engl. talker), die Floppy Hörer (engl. listener). Dann sind NRFD (nicht Daten anzunehmen) und NDAC (Daten [noch] nicht angenommen) als INPUT-Leitungen geschaltet (das heißt, der Rechner empfängt durch diese Leitungen Signale von der Floppy, während DAV (Daten gültig) als OUTPUT-Leitung (das heißt, der Rechner sendet über diese Leitung an die Floppy) geschaltet ist.

ACHTUNG: Der Bus arbeitet mit inverser Logik (das heißt, der Zustand logisch wahr wird durch eine Spannung nahe 0 V realisiert [low oder L], wohingegen der Zustand logisch falsch durch eine Spannung nahe 5 V realisiert wird [high oder H]).

Die Datenübertragung verläuft in sieben Phasen:



- (1) Die Floppy setzt NRFD auf H, signalisiert also, daß sie bereit ist, Daten zu empfangen.
- (2) Der Rechner gibt das nächste Byte auf den Bus und wartet bis die Daten eingeschwungen sind. Das kann vor, während oder nach (1) geschehen.
- (3) Der Rechner testet, ob die Floppy bereit ist, Daten zu empfangen (NRFD = H?). Ist er bereit, setzt er DAV auf L (wahr), sendet also »Daten gültig« an die Floppy.
- (4) Sobald die Floppy »Daten gültig« (DAV = L) erkannt hat, setzt sie NRFD wieder auf L (wahr), sendet also, daß sie im Moment keine weiteren Daten empfangen kann.
- (5) Hat die Floppy die Daten vollständig empfangen, setzt sie NDAC auf H (falsch), gibt also dem Rechner bekannt, daß sie die Daten empfangen hat.
- (6) Hat der Rechner NDAC = (Daten empfangen) von der Floppy erhalten, setzt er DAV auf H (falsch), das heißt, er sendet der Floppy, daß die Daten jetzt nicht mehr gültig sind.
- (7) Sobald die Floppy feststellt, daß DAV = H ist, die Daten also nicht mehr gültig sind), setzt sie NDAC = L und stellt damit den Zustand vor (1) her. Das Handshake ist damit beendet.

Hat die Floppy die Daten vollständig verarbeitet, so setzt sie NRFD auf H, womit das Handshake von vorne beginnt, und das nächste Byte übertragen werden kann.

Achtung: Für die Übertragung gelten folgende zeitliche Beschränkungen:

- 1.) Ist der Rechner Sprecher (wie in obigem Beispiel), erwartet er 64 MS nachdem dem Hörer »bereit für Daten« (NRFD = H, Phase [1]) gegeben hat, das Signal »Daten empfangen« (NDAC = H, Phase [5]).
- 2.) Ist der Rechner Hörer, erwartet er 64 MS nachdem er »bereit für Daten« (NRFD = H) gegeben hat, das Signal »Daten gültig« (DAV = L).

Werden diese Zeiten überschritten, beendet der Rechner die Übertragung bzw. das Empfangen von Daten und setzt die Statusvariable ST auf 1 bzw. 2 (siehe 1.0.3.).

1.02. So merken die Peripherie-Geräte, was der Rechner von ihnen will

Der Management-Bus

Der Management-Bus kontrolliert den Zustand des Daten-Busses und gibt den Peripherie-Geräten bekannt, ob Daten, Adressen oder Befehle am Datenbus anliegen. Es gibt 5 Management-Signale:

- 1.) ATN (Attention = Aufmerksamkeit): ist ATN auf L, so befinden sich periphere Adressen oder Kontrollmitteilungen am Bus. Ist ATN auf H, so können nur vorher zugeordnete Geräte Daten übermitteln.
- 2.) EOI (End or identify = Ende oder Kennzeichnung): ist EOI auf L, so ist das letzte Datenbyte übertragen worden. Der Rechner setzt immer EOI auf L, sobald er (als Sprecher) das letzte Datenbyte übertragen hat, jedoch erzeugen nicht alle Peripherie-Geräte dieses Signal. EOI = L bewirkt, daß die Statusvariable ST (siehe unten) auf 64 gesetzt wird.
- 3.) IFC (Interface clear = löschen des Busses): Ist IFC = L, so wird der Datenbus initialisiert, d. h. die Hörer-Sprechzuleitung wird aufgehoben. Dieses Signal ist analog dem Reset-Signal beim Rechner. Beim Einschalten des Rechners setzt dieser IFC automatisch ca. 100 ms auf L.
- 4.) SRQ (Service request = Bedienungsaufruf): SRQ = L bedeutet, ein Peripherie-Gerät teilt dem Controller mit, daß Bedienung notwendig ist. Dieses Signal wird nicht von Basic verarbeitet, kann aber durch ein geeignetes Programm verwendet werden (siehe Adressenbelegung 1.0.4.)
- 5.) REN (Remote enable): bei Ihrem Rechner stets auf L.

1.0.3. Die Statusvariable ST

ST ist eine BASIC-Variable zur Kontrolle von INPUT - OUTPUT (Eingabe - Ausgabe) Operationen. ST kann Werte zwischen -128 und +127 annehmen.

Für Operationen am IEC-Bus sind folgende Werte von ST von Bedeutung:

Wert	Bedeutung	Erläuterung
1	Zeit beim Hörer überschritten	Das Gerät hat nicht innerhalb von 64 ms »Daten empfangen« gegeben.
2	Zeit beim Sprecher überschritten	Das Gerät hat kein »Daten gültig« innerhalb von 64 ms gegeben.
64	Ende oder Kennzeichnung	EOI = L ist mit dem letzten Datenbyte übermittelt worden. Nicht alle Geräte senden EOI.
-128	Gerät nicht vorhanden	Gerät reagiert nicht auf Adressierung. Es erfolgt Rückkehr nach BASIC und die Fehlermeldung ?DEVICE NOT PRESENT ERROR

1.0.4. Wichtige INPUT - OUTPUT (Eingabe - Ausgabe) Adressen im Rechner

IEC	Betriebsart	Bits	Dez. Adresse	Hex. Adresse
DIO1-8	Eingang	0-7	59424	E820
DIO1-8	Ausgang	0-7	59426	E822
NRDC	Ausgang	3	59425	E821
DAV	Eingang	3	59427	E823
SRQ	-	7	59427	E823
EOI	Eingang	6	59408	E810
NDAC	Eingang	0	59456	E840
NRFD	Ausgang	1	59456	E840
ATN	Ausgang	2	59456	E840
NRFD	Eingang	6	59456	E840
DAV	Ausgang	7	59456	E840

Weitere Informationen über den IEC-Bus finden Sie in der Anleitung Ihres Rechners.

1.1.0. So öffnen Sie eine Datei (file)

Der OPEN-Befehl

Der OPEN-Befehl sieht so aus:

```
OPEN LA,G,SA, DN
```

LA, G und SA sind ganze Zahlen oder ganzzahlige Variable.

DN ist ein String (z. B. "0:PROG,P,R"), eine Stringvariable (z. B. a\$, "ABC27") oder Stringausdruck (z. B. mid\$(a\$+b\$,30,10)).

Die Bedeutung von LA, G, SA und DN:

LA ist die logische Adresse des Peripherie-Gerätes (Filenummer). Der Programmierer kann für die LA jede ganze Zahl zwischen 0 und 255 wählen.

G ist die Gerätenummer des Peripherie-Gerätes. Sie ist im Gerät FEST EINGESTELLT. Für Ihre Floppy beträgt sie normalerweise 8. Sollen mehrere Floppy's an einem Rechner betrieben werden, so kann die Gerätenummer durch den Commodore-Service zwischen 8 und 15 eingestellt werden.

SA ist die Sekundäradresse des Peripherie-Gerätes. Bei Ihrer Floppy dient sie dazu, den jeweiligen Kanal (siehe 6.1.) einzurichten. Die Floppy verfügt über die Kanäle 0 bis 15, wobei die Kanäle 0, 1 und 15 besondere Bedeutung haben:

Kanal 0 und 1 werden bei "LOAD" bzw. "SAVE" verwendet.

Kanal 15 ist der sogenannte Kommando-/Fehlerkanal (siehe 1.1.0.1.).

Datenkanäle können also durch Zahlen von 2 bis 14 für SA eingerichtet werden.

ACHTUNG: Durch Initialisierungen der Diskette werden alle Kanäle außer Kanal 15 geschlossen. Es empfiehlt sich also, Datenkanäle erst nach dem Initialisieren zu öffnen.

DN ist der sogenannte Dateiname. Im Verkehr mit der Floppy hat der DN allerdings eher die Funktion Kommandos an die Floppy zu übermitteln. DN muß im Gegensatz zu LA, G und SA beim OPEN-Befehl nicht angegeben werden. Beim Öffnen des Kanals 15 (Kommando-/Fehlerkanal) kann DN im allgemeinen weggelassen werden. Beim Schreiben oder Lesen von Dateien muß DN allerdings im "OPEN" angegeben werden.

Der OPEN-Befehl stellt die Verbindung zwischen dem Rechner und den Geräten am IEC-Bus (siehe 1.0.) her.

Erst nach dem "OPEN" können über "PRINT#" Daten auf ein Peripherie-Gerät gegeben werden oder durch "INPUT#" und "GET#" von dort empfangen werden. Hinter diesen Befehlen wird nur die LOGISCHE ADRESSE des Peripherie-Gerätes gegeben. Der Rechner speichert durch den OPEN-Befehl die Gerätenummer, und Sekundäradresse, die zu der angegebenen Logischen Adresse gehören. So wird stets das richtige Gerät angesprochen.

Mit einer Logischen Adresse kann stets nur eine Datei eröffnet werden! Wird versucht, mit der selben logischen Adresse eine weitere Datei zu eröffnen, erscheint:

?FILE OPEN ERROR (IN ...) (Datei ist schon eröffnet)

Ihr Rechner kann maximal 10 Dateien (Files) gleichzeitig verwalten. Sind bereits 10 Dateien geöffnet, und stößt der Rechner auf ein weiteres "OPEN", so antwortet er mit:

?TOO MANY FILES ERROR (IN ...) (zu viele Dateien geöffnet)

Gleichzeitig werden alle anderen Dateien geschlossen, und zwar ohne ordnungsgemäßes "CLOSE", was böse Folgen haben kann (siehe 1.1.1.).

Es empfiehlt sich daher, Dateien, die nicht mehr benötigt werden, direkt nach Gebrauch mit "CLOSE" (siehe 1.1.1.) wieder zu schließen. Dies ist auch aus anderen Gründen wichtig, da "CLOSE" auf keinen Fall vergessen werden darf.

Eine Ausnahme bildet der »Kommando-/Fehlerkanal« (Kanal 15). Er sollte stets als erster geöffnet und als letzter wieder geschlossen werden!

Die Floppy kann nur maximal 5 geöffnete Dateien verwalten.

Beispiele:

OPEN15,8,15	LA=15; G=8; SA=15	Hiermit wird Kanal 15 mit der LA=15 geöffnet. Die LA kann auch jede andere ganze Zahl zwischen 1 und 255 sein. LA=15 erhöht jedoch die Übersicht im Programm.
OPEN A,B,C,D\$+E\$	LA=A; G=B; SA=15; DN=D\$+E\$	Anstelle von Zahlen und Strings können auch Variable oder Ausdrücke verwendet werden.

FALSCH:

OPEN8,9,15	LA=8; G=9; SA=15	Durch diesen Befehl wird die Floppy im Normalfall nicht angesprochen, da die angegebene Gerätenummer falsch ist!
OPEN3,8,99	LA=3; G=8; SA=99	Es gibt keinen Kanal 99 auf der Floppy!
OPEN15,8,15,P\$;E\$		Statt ";" muß "+" stehen!
OPEN15,8,A\$,B\$	LA=15; G=8; SA=A\$	SA darf kein String und keine Stringvariable sein!

1.1.0.1. Der Kommando-/Fehlerkanal

Der Kanal 15 der Floppy hat eine besondere Funktion. Über ihn können mit 'PRINT#LA, »Kommando«' (LA = Logische Adresse, mit der der Kommando-/Fehlerkanal geöffnet wurde) Kommandos (siehe 3.) an die Floppy gegeben werden, und die Fehlermeldung kann von der Floppy mit 'INPUT#LA, F, F\$,F2' abgefragt werden (allerdings nur im Programm). Näheres über die Fehlermeldung erfahren Sie in 9. Kanal 15 sollte stets als letzter geschlossen werden, da sein Schließen das unbeabsichtigte Schließen anderer Floppy-Kanäle bewirken kann. Allerdings fehlt diesen Kanälen dann das ordnungsgemäße "CLOSE". Das kann dazu führen, daß Dateien nicht mehr gelesen werden können.

Beispiele:

10 OPEN15,8,15	Kommando-/Fehlerkanal wird geöffnet.
20 PRINT#15,"IO"	Der Floppy wird der Befehl gegeben, die Diskette in Laufwerk 0 zu initialisieren.
30 INPUT#15,F,F\$,F1,F2	Die Fehlermeldung wird gelesen, in F, F\$,F1 und F2 gespeichert . . .
40 PRINT F,F\$,F1,F2	. . . und auf dem Bildschirm angezeigt.

Falsch:

10 ?#15,"IO"	"PRINT#kann weder mit "?#15" noch mit "?15" abgekürzt werden! Verwenden Sie "pR15".
30 OPEN15,8,4	Öffnet Kanal 4
40 PRINT#5,"IO"	Dieser PRINT#-Befehl muß die LA enthalten, mit der Kommando-/Fehlerkanal geöffnet wurde (hier also 15)!

1.1.0.2. Zwei Dinge auf einmal: Befehle im "OPEN"

Gleichzeitig mit dem OPEN-Befehl können auch Kommandos gegeben werden. Der Dateiname des "OPEN" enthält dann das Kommando. Wie es im einzelnen aufgebaut ist, ist in 3. ausführlich erläutert.

So sieht ein Befehl im "OPEN" aus:

OPEN LA, G,SA,"KOMMANDO" Für LA,G,SA siehe 1.1.0.0.

Zwischen der Übermittlung eines Kommandos im "OPEN" und der Übermittlung durch ein: 'PRINT#LA, "KOMMANDO"' auf dem Kommandokanal besteht ein wichtiger Unterschied:

Im zweiten Fall meldet sich der Rechner sofort mit "READY." zurück bzw. setzt das Programm fort.

Dagegen wartet der Rechner bei Kommandos im "OPEN" mit der Fortführung des Programms, bis die Floppy den Befehl ausgeführt hat.

Dieser Unterschied kann unter Umständen wesentlich sein. Die Befehle "D", "N" und "V" (siehe 3.2., 3.7., 3.6.) sollten daher nicht im "OPEN" gegeben werden, da der Rechner sonst bis zu 80 Sekunden warten muß.

Beispiele:

OPEN15,8,15,"IO"	Der Kommando-/Fehlerkanal der Floppy wird mit LA = 15 geöffnet. Gleichzeitig wird der Floppy das Kommando gegeben, die Diskette in Laufwerk 0 zu initialisieren.
OPEN4,8,2,"R1:"+A\$+"="+"1:"+B\$	Kanal 2 wird mit LA = 4 eröffnet. Gleichzeitig erhält die Datei mit dem Namen B\$ des Laufwerks 1 den Namen A\$ ("R" siehe 3.4.).

Falsch:

OPEN4,8,2,"R1:";A\$;"=";"1:";B\$	Die ";" müssen durch "+" ersetzt werden, da der Dateiname ein STRINGAUSDRUCK sein muß. Es handelt sich ja nicht um ein "PRINT".
----------------------------------	---

1.1.0.3. Zuordnen eines Puffers für den Direktzugriff

Das Zuordnen eines Puffers geschieht durch:

a.) OPEN LA,G,KN,"#"

oder durch:

b.) OPEN LA,G,KN,"#P"

LA=logische
Adresse

G=Geräte-
nummer

KN=Kanal-
nummer

P=Puffer-
nummer

Die Gerätenummer ist, wie üblich, normalerweise 8, siehe 1.1.0.0..

Die Kanalnummer muß zwischen 2 und 14 liegen, siehe 1.1.0.0..

Die Puffernummer muß, falls sie angegeben wird (also bei b.), zwischen 0 und 12 liegen.

Der Unterschied zwischen a.) und b.) liegt darin, daß bei a.) der NÄCHSTE FREIE PUFFER dem durch KN angegebenen Kanal zugeordnet wird, während bei b.) ein BESTIMMTER PUFFER zugeordnet wird. Auf ihn kann über "M-R", "M-W" und "M-E" (siehe 7.9. bzw. 7.10.) zugegriffen werden.

Die Puffer 13–15 können nicht belegt werden, da sie vom DOS (siehe 6.) bereits belegt sind.

Die oben angegebenen Befehle bewirken folgendes: Dem durch KN angegebenen Kanal wird ein Puffer zugeordnet. Auf diesem Kanal kann mit der angegebenen logischen Adresse zugegriffen werden.

ACHTUNG: Durch initialisieren werden alle Kanäle bis auf den Kommando-/Fehlerkanal geschlossen!
Es empfiehlt sich also, erst nach dem Initialisieren Kanäle zu eröffnen und Puffer zuzuordnen.

Beispiele:

OPEN 2,8,4,"#10" Dem Kanal 4 wird der Puffer 10 zugeordnet.
OPEN12,8,12,"#" Dem Kanal 12 wird der nächste freie Puffer zugeordnet.

Falsch:

OPEN10,8,15,"#" Dem Kanal 15 kann kein Puffer zugeordnet werden.
OPEN10,8,12,"#13" Die Puffer 13–15 sind vom DOS belegt.

1.1.0.4. So schließen Sie eine Datei (der CLOSE-Befehl)

Der CLOSE-Befehl sieht so aus:

CLOSE LA
LA=logische Adresse

Der CLOSE-Befehl meldet eine Datei beim Rechner wieder ab. Es braucht nur die logische Adresse der Datei, die abgemeldet werden soll, angegeben zu werden.

Wichtige Funktionen von "CLOSE":

Der CLOSE-Befehl erfüllt bei Dateien, die auf Diskette geschrieben werden, drei wichtige Funktionen:

- 1.) Der letzte Datenblock wird vom Puffer auf die Diskette geschrieben.
- 2.) Dieser Datenblock wird mit der EOF (End of file = Ende der Datei)-Markierung versehen.
- 3.) Das Verzeichnis der freien/belegten Blöcke (die BAM, siehe 2.2.) wird auf die Diskette geschrieben.

Wird kein ordnungsgemäßes "CLOSE" gegeben, so fehlt auf der Diskette der letzte Datenblock und das EOF-Zeichen. Die Datei wird nicht in das Inhaltsverzeichnis aufgenommen. Sie kann nicht kopiert werden und wird bei einem »V« (Verify-Befehl der Floppy = Bereinigen der Diskette, siehe 3.6.) vernichtet.

Nicht mehr benötigte Dateien sollten deshalb sofort mit "CLOSE" geschlossen werden.

Das ist auch deshalb günstig, da der Rechner nur 10 Dateien gleichzeitig verwalten kann.

ACHTUNG: Bei "LOAD" und "RUN" »vergißt« der Rechner, welche Gerätenummer und Sekundäradresse zu einer logischen Adresse gehören. Für ihn sind alle Dateien geschlossen. Es wird aber kein ordnungsgemäßes "CLOSE" geben! Deshalb fehlen die EOF-Markierung und die BAM auf der Diskette, was die oben beschriebenen negativen Folgen hat.

1.1.2. Der Weg vom Computer zur Floppy (der PRINT #-Befehl)

1.1.2.0. Allgemeine und mögliche Fehlermeldungen

Der PRINT #-Befehl sieht so aus:

PRINT#LA,VARIABLE;VARIABLE;...;

Statt der Variablen können auch Ausdrücke oder Strings gesendet werden.

Mit PRINT# können Daten oder Kommandos an die Floppy übermittelt werden. "PRINT#" setzt voraus, daß eine Datei mit der entsprechenden logischen Adresse eröffnet ist. Andernfalls erscheint:

?FILE NOT OPEN ERROR (IN...) (Datei nicht eröffnet)

Ist die Datei zwar im Rechner geöffnet, das angesprochene Gerät aber nicht angeschlossen oder nicht eingeschaltet, oder reagiert es aus irgendwelchen anderen Gründen nicht auf die Adressierung, so erscheint:

?DEVICE NOT PRESENT ERROR (IN...) (Gerät nicht vorhanden)

Wird ein "PRINT#" auf einen nicht geöffneten Kanal versucht, so leuchtet an der Floppy die Fehlerlampe auf. Abfrage des Fehlers ergibt:

70 NO CHANNEL 00 00

1.1.2.1. Wichtige Hinweise zu "PRINT#"

Die Übermittlung von Kommandos über "PRINT#" ist in 3. erläutert. Bei der Übermittlung von Daten ist folgendes unbedingt zu beachten:

- 1.) Die einzelnen Variablen sollten stets durch ";" und nicht durch "," getrennt werden.

- 2.) Der PRINT #-Befehl muß immer mit ";" abgeschlossen werden.
- 3.) Sollen die Daten mit "INPUT#" gelesen werden, so dürfen nie mehr als 80 Zeichen hintereinander ohne CR (Carriage return = Wagenrücklauf = CHR\$(13)) gesendet werden!
- Zu 1.) Werden die Variablen durch ";" getrennt, so werden automatisch nach jedem ";" einige Leerzeichen gesendet (analog zu "PRINT" auf dem Bildschirm).
- Zu 2.) Wird der PRINT #-Befehl nicht mit ";" oder ";" abgeschlossen, so wird automatisch ein CR (Carriage Return = Wagenrücklauf = CHR\$(13)) und ein LF (Line Feed = Zeilenvorschub = CHR\$(10)) angehängt. Diese beiden Zeichen werden auch auf der Diskette geschrieben. Beim Einlesen wird CR als Begrenzungszeichen (Delimiter, siehe 1.1.3.3.) erkannt und nicht übertragen. LF wird als erstes Zeichen der nächsten Variablen interpretiert, was im allgemeinen unerwünscht wird.
- Zu 3.) siehe 1.1.3.1.

Will man ein CR (Carriage Return = Wagenrücklauf = CHR\$(13)) senden, so sendet man CHR\$(13) oder setzt zum Beispiel CR\$ = "CHR\$(13)" und sendet CR\$.

Beispiel:

10 OPEN14,8,4	Kanal wird mit logischer Filenummer 14 geöffnet.
20 S\$=A\$+CHR\$(13)+B\$+CHR\$(13)+C\$	In S\$ werden CR eingeschoben, damit nicht mehr als 80 Zeichen ohne CR hintereinander stehen.
30 PRINT#14,S\$;CHR\$(13);D\$;	Die Variablen werden auf Floppy geschrieben.
Falsch:	
10 OPEN14,8,4	Kanal 4 wird mit LA 14 eröffnet.
20 ?#14,A\$?# oder ? sind keine zulässigen Abkürzungen für "PRINT#". Außerdem fehlt ";" am Schluß.
10 OPEN14,8,4:PRINT#14,A\$,B\$,C\$;	Die ";" zwischen den Variablen verursachen die Speicherung unnötiger Leerzeichen auf der Floppy.

1.1.3. So holen Sie Ihre Daten wieder zurück: Der INPUT #-Befehl

1.1.3.0. Allgemeines

Der INPUT #-Befehl sieht so aus:

```
INPUT#LA,VARIABLE,VARIABLE,...
```

Die Variablen müssen ihrer Art (Zahlen-, String- oder Integervariable) nach in derselben Reihenfolge gelesen werden, in der sie geschrieben wurden.

"?FILE NOT OPEN ERROR" "?DEVICE NOT PRESENT ERROR" und Aufleuchten der Fehlerlampe an der Floppy: siehe 1.1.2.0..

Durch den INPUT #-Befehl werden Daten von der Floppy zum Rechner übertragen. Genauer gesagt, werden die Daten solange in den BIP (BASIC-Input Puffer = BASIC - Eingabe - Zwischenspeicher) geladen, bis dem Rechner ein "CR" (Carriage Return = CHR\$(13) = Wagenrücklauf) entdeckt. Dann werden solange Daten aus dem BIP in die Variablen übertragen, bis ein Begrenzungszeichen (Delimiter, siehe 1.1.3.1.) folgt. Die Daten bis zum nächsten Begrenzungszeichen werden jeweils in die nächste Variable geladen. Ist der BIP leer, und sind noch weitere Variable zu laden, wird er bis zum nächsten "CR" wieder geladen.

1.1.3.1. Der BIP läuft über

Der BIP verfügt nur über 80 Byte Speicherkapazität (Adressen 512–591 = \$0200–\$024F des Rechners). Hinter diesen 80 Bytes BIP folgen 30 Bytes zur Speicherung der logischen Adressen, Geräteummern und Sekundäradressen der geöffneten Dateien, danach 10 Byte Tastaturspeicher und darauf 2*192 Byte Recorder – Zwischenspeicher (vergleiche Betriebsanleitung des Rechners, Seite 58).

Stehen auf der Diskette mehr als 80 Zeichen ohne "CR" hintereinander und werden diese Daten mit "INPUT#" gelesen, so wird nicht nur der BIP vollgeschrieben, sondern auch die folgenden Speicherplätze des Betriebssystems.

Der Rechner gerät dann normalerweise in einen unkontrollierbaren Zustand !!!

Deshalb gilt grundsätzlich:

Daten, bei denen mehr als 80 Zeichen ohne "CR" hintereinanderfolgen, nicht mit "INPUT#" lesen!!!

1.1.3.2. Abhilfe bei mehr als 80 Zeichen: GET#

Ist es nicht möglich, beim Abspeichern der Daten nach jeweils maximal 80 Zeichen ein CR zu übermitteln, so gibt es die Möglichkeit die Daten mit "GET#" (siehe 1.1.4.) zu lesen!

1.1.3.3. Die Begrenzungszeichen (Delimiter)

Begrenzungszeichen sind:

- 1.) CR (Carrriage Return = Wagenrücklauf) = CHR\$(13)
- 2.) "," (Komma) = CHR\$(44)
- 3.) ":" (Doppelpunkt) = CHR\$(58)

Der Rechner lädt jeweils die Daten zwischen zwei Begrenzungszeichen in eine Variable. In den BIP werden jedoch alle Daten zwischen zwei "CR" eingeladen, ohne Rücksicht auf Kommata oder Doppelpunkte.

Bei STRINGVARIABLEN gibt es eine wichtige Ausnahme: wenn nach dem letzten CR eine ungerade Zahl von "¶" = CHR\$(34) empfangen wurde, so werden "," und ":" als DATEN und NICHT als BEGRENZUNGSZEICHEN interpretiert. Sie werden also wie jedes andere Zeichen in den String gebracht.

Beispiele:

10 OPEN4,8,4,"0:DATEN,S,R"	Öffnen einer Datei, um die Datei "DATEN" von der Diskette in Laufwerk 0 zu lesen.
20 INPUT#4,A\$,B\$,C\$	Lesen der ersten drei Strings in A\$,B\$ und C\$.
10 OPEN5,8,5,"#"	Dem Kanal 5 wird der nächste freie Puffer zugeordnet.
20 PRINT#5,A\$;CHR\$(13);B%;CHR\$(13);A;CHR\$(13);A\$,B% und A werden in den Puffer geschrieben.	
30 INPUT#5,Z\$,Y%,J	Der Inhalt des Puffers wird in Z\$, Y% und J gelesen.

1.1.4. Zurück in kleinen Schritten (der GET#-Befehl)

1.1.4.0. Allgemeines

Der GET#-Befehl sieht so aus:

```
GET#LA,STRINGVARIABLE
```

Fehlerlampe siehe 1.1.2.0.

Statt der Stringvariablen kann notfalls auch eine Zahlvariable verwendet werden, wenn nur Zahlen von 0-9, "+", "-", " ", ":", und "." von der Floppy gesendet werden. Es empfiehlt sich aber, stets eine Stringvariable zu verwenden. Sie kann ja durch "VAL" leicht in eine Zahl verwandelt werden.

"GET#" holt immer genau ein Zeichen von der Floppy, und zwar ohne Rücksicht auf Begrenzungszeichen (Delimiter). Begrenzungszeichen werden, wie jedes andere Zeichen, als Daten empfangen.

Achtung! "GET#" übernimmt CHR\$(0) als leeren String ("").

Beispiel:

```
10 OPEN 4,8,7
20 A$=CHR$(0)
30 A=LEN(A$)
40 PRINT#4,A$;
50 GET#4,G$
60 G=LEN(G$)
```

A (= Länge von CHR\$(0)) hat den Wert 1; G (= Länge von " ") den Wert 0.

Wenn es also auf STELLENRICHTIGE WIEDERGABE ankommt, oder die mit "GET#" gelesenen Zeichen in ASCII-Codes umgewandelt werden sollen, muß folgende Routine angewandt werden:

```
50 GET#4,G$:IFG$="" THENG$=CHR$(0)
```

1.1.4.1. Vergleich von "GET#" und "INPUT#"

"GET#" ist, da jeweils nur ein Zeichen empfangen werden kann und meistens noch logische Entscheidungen (s. o.) nötig sind, um eine bis zwei Größenordnungen langsamer als "INPUT". Mit "GET#" können jedoch Datensätze gelesen werden, bei denen zwischen zwei "CR" mehr als 80 Zeichen stehen.

2.0. Was ist eine Datei?

Hier soll nicht der Begriff im Sinne der Informatik definiert werden. Es reicht für das Verständnis der Vorgänge in der Floppy, wenn man unter einer Datei eine Zusammenfassung von Daten versteht; Daten können hierbei Zahlen, alphanumerische Variablen oder auch, wenn es sich um ein Programm handelt, Interpreter-Codes sein.

Allen Arten von Daten ist gemeinsam, daß ihre Grundeinheit das Byte ist, das, will man nicht noch in Bits (ein Bit kennt nur den Zustand 1 oder 0) unterteilen, die Grundeinheit des Rechners und der Floppy darstellt. Ein Byte kennt 256 Werte, nämlich 0-255. Eine Zahlenvariable besteht aus 5 Bytes, eine Ganz-

zahlvariable aus 2 Bytes und eine Textvariable benötigt pro Buchstabe ein Byte.

In dieser kleinsten Einheit geschieht auch die Verarbeitung von Daten. Sie werden der Floppy Byte für Byte übergeben und von dieser Byte-weise abgespeichert.

Es gibt drei Arten von Dateien, die sich prinzipiell unterscheiden:

2.0.0. Die PRG-Datei

Bei einer PRG-Datei handelt es sich, rein technisch gesehen, um einen zusammenhängenden Speicherbereich. Meistens handelt es sich hierbei um Programme, die im Rechner ab der Speicherstelle 1025 (\$0401) abgespeichert werden. Genausogut kann aber auch jeder andere Speicherbereich über den .S-Befehl des Monitors auf die Floppy übertragen werden.

Das Charakteristikum der PRG-Datei ist, daß sie bei ihrem Laden genau wieder an der Speicherstelle steht, von der aus sie auch abgespeichert wurde. Dies wird dadurch erreicht, daß zu Beginn der Datei zunächst zwei Bytes abgespeichert werden, die die Adresse des Arbeitsspeicherbereiches enthalten, ab der die Datei im Arbeitsspeicher stand.

Beim Laden der Datei werden nun die einzelnen Bytes dieser Datei ab der geladenen Adresse wieder in den Arbeitsspeicher des Rechners geschrieben.

2.0.1. Die SEQ-Datei

Bei einer SEQ-Datei (SEQ = sequentiell) handelt es sich im Gegensatz zu einer PRG-Datei um eine Datei, in der konkrete Daten gespeichert sind, meistens die Inhalte von Zahlen- oder Textvariablen. Sie stehen in der Datei sequentiell, das heißt hintereinander. Um solche Daten auf Diskette zu schreiben, verwendet man den BASIC-Befehl "PRINT#" (siehe 1.1.2.).

Um in derselben Reihenfolge wieder abzurufen, benötigt man die BASIC-Befehle "INPUT#" (siehe 1.1.3.) oder "GET#" (siehe 1.1.4.), wobei "INPUT#" die Daten so ausgibt, wie sie geschrieben wurden, während "GET#" die Datei byteweise liest.

Wegen des sequentiellen Aufbaus der Datei ist es oft nicht sinnvoll, die Dateien zu lang werden zu lassen. Will man namentlich Daten abrufen, die ganz am Ende der Datei stehen, so stehen diese erst zur Verfügung, wenn alle in der Datei vorher stehenden Daten bereits abgerufen sind.

Es empfiehlt sich deshalb immer, statt einer umfangreichen Datei mehrere kleine Dateien zu schreiben.

Bei beiden Dateiarten (SEQ und PRG) ist folgendes zu beachten: Da die Diskette in Blöcke unterteilt ist, gilt ein Block, sobald er »angebrochen« ist, als belegt, so daß der unbenutzte Teil dieses Blocks nicht mehr für die nächste Datei zur Verfügung steht. Sie kann erst im nächsten Block beginnen.

Soll die Diskette voll ausgenutzt werden, ist es unter Umständen sinnvoll, durch entsprechende Anpassung der Dateien an die Blockstruktur ein paar Blöcke einzusparen.

2.0.2. Der Direktzugriff

Außer den durch die Floppy verwalteten Zugriffsarten (SEQ und PRG) gibt es auch noch die Möglichkeit, auf die Diskette (fast) direkt zuzugreifen. Der gesamte Speicherbereich einer Diskette ist in 690 Blöcke eingeteilt (Näheres siehe 2.1.). Es gibt nun die Möglichkeit, einzelne Blöcke in einen Puffer (Kommando "B-R", siehe 5.4.25.3) zu geben. Aus diesem Puffer kann man nun den Inhalt des gelesenen Blocks durch "INPUT#" (siehe 1.1.3.) oder "GET#" (siehe 1.1.4.) in den Rechner gebracht werden. Umgekehrt kann durch "PRINT#" (siehe 1.1.2.) in den Pufferspeicher geschrieben werden, dessen Inhalt dann (mit "B-W" siehe 17.4.21.) direkt auf die Diskette übertragen werden kann.

Die Adresse eines zu lesenden Blocks ergibt sich aus seiner Spur- und seiner Sektornummer (siehe 2.1.). Die Länge der Blöcke ist konstant und beträgt 255 Byte.

Der Vorteil des Direktzugriffs liegt in der niedrigen Zugriffszeit (0,1–2 Sekunden), und darin, daß die Anzahl der Dateien nicht nur 152, wie bei der SEQ-Datei, sondern 670 pro Diskette betragen kann.

Der Nachteil liegt darin, daß eine eigene Verwaltung für die Blöcke vom Programm aufgebaut werden muß, da die Dateien nicht mehr von der Floppy im Inhaltsverzeichnis verwaltet und abgerufen werden können. Außerdem ist man an die Blocklänge von 255 Zeichen gebunden.

2.1. So sind Spuren und Blöcke verteilt

Der gesamte Speicherplatz einer Diskette ist in 690 Blöcke eingeteilt; jeder Block enthält 256 Bytes, wobei das erste Byte vom Puffer-Zeiger (siehe 17.4.8.21.) belegt ist, so daß 255 Bytes zur Verfügung stehen. Die 690 Blöcke sind auf 35 Spuren verteilt, wobei die Anzahl der Blöcke pro Spur von außen nach innen kleiner wird. Es ergibt sich folgende Verteilung der Blöcke auf die Spuren:

Spur	Sektor	Anzahl der Blöcke
01–17	00–20	je 21
18–24	00–19	je 20
25–30	00–17	je 18
31–35	00–16	je 17

Die gesamte Spur 18 wird für das Inhaltsverzeichnis und die BAM (siehe 2.2.) gebraucht, auch wenn keine Datei im Inhaltsverzeichnis steht, also die Diskette nur für den Direktzugriff benutzt wird. Demnach stehen nur 670 Blöcke in diesem Fall für den Direktzugriff zur Verfügung. Sollte man dennoch gezwungen sein, eine Diskette sowohl für PRG- und SEQ-Dateien als auch für den Direktzugriff verwenden, so kann man mit den Befehlen "B-A" und "B-F" (siehe 3.3.9.) verhindern, daß die beiden Dateiformen sich gegenseitig überschreiben.

2.2. Das Inhaltsverzeichnis und die BAM

Jede Diskette enthält außer den auf ihr gespeicherten Daten noch drei weitere Informationen:

a. Den Diskettennamen und die ID (Identität):

Der Diskettenname spielt eine untergeordnete Rolle. Er wird zusammen mit der ID beim Abrufen des Inhaltsverzeichnisses ausgedruckt, und soll dem Programmierer das Erkennen einer falschen Diskette erleichtern.

Die ID wird beim Initialisieren (siehe 3.1.) der Diskette zusammen mit der BAM (siehe unten) in den Arbeitsspeicher der Floppy geladen und dient dazu, neu eingelegte, aber noch nicht initialisierte Disketten als solche zu erkennen (die ID der eingelegten Diskette wird mit der im Arbeitsspeicher abgelegten ID verglichen und bei Nichtübereinstimmen eine Fehlermeldung ausgegeben (siehe 8.1.)).

b. Die BAM (Block availability map = Block-Verfügbarkeitstabelle):

Sie enthält die Information, welche Blöcke der Diskette belegt sind. Die BAM wird beim Initialisieren zusammen mit der ID (siehe oben) in den Arbeitsspeicher der Floppy geladen.

Die ID und die BAM sind in Spur 18, Sektor 0, abgelegt.

c. Das Inhaltsverzeichnis:

Spur 18, Sektor 1–19, enthalten das Inhaltsverzeichnis der Diskette. In ihm stehen sämtliche Dateinamen mit der Information, ob es sich um PRG- oder SEQ-Dateien handelt, und, für den Programmierer nicht abrufbar, in welchen Blöcken die Dateien zu finden sind.

Wird eine Datei abgerufen, so »sieht« die Floppy zuerst im Inhaltsverzeichnis nach, wo die gesuchte Datei zu finden ist, um sie dann von dort zu holen.

Wird eine Datei gespeichert, so wird gleichzeitig das Inhaltsverzeichnis auf der Diskette ergänzt. Außerdem werden die betreffenden Blöcke in der BAM im Arbeitsspeicher der Floppy als belegt gekennzeichnet.

WICHTIG: Die neue BAM wird außer bei "SAVE" und ".S" erst nach dem Schließen der Verbindung zur Floppy (siehe 2.2.0.) auf die Diskette geschrieben, während das Inhaltsverzeichnis sofort auf der Diskette ergänzt wird.

2.2.0. Das Schreiben der BAM bei "CLOSE"

Die während der Arbeit mit der Diskette in dem Arbeitsspeicher der Floppy geänderte BAM wird erst beim Schließen des Datenkanals auf die Diskette übertragen.

Deshalb darf der CLOSE-Befehl für geöffneten Files auf keinen Fall vergessen werden!

Anmerkung: Beim Speichern von Programmen mit "SAVE" oder ".S" wird das Öffnen und Schließen der Verbindung mit der Floppy vom Rechner übernommen, so daß hier die BAM automatisch auf die Diskette übertragen wird.

2.2.1. So sehen Sie sich das Inhaltsverzeichnis an

Das Inhaltsverzeichnis einer Diskette (die natürlich vorher initialisiert worden sein muß [siehe 3.1.]) wird mit dem Befehl 'LOAD"\$L",8' (L=Laufwerksnummer) geladen. Das \$-Zeichen weist die Floppy an, anstelle eines Programms das Inhaltsverzeichnis auszugeben. Es wird als Programm in den BASIC-Speicher des Rechners geschrieben, und läßt sich durch "LIST" ausdrucken.

Schreibweise:

```
LOAD"$L:NAME",G
```

L = Laufwerksnummer NAME = Programmname G = Gerätenummer

Die Laufwerksnummer ist entweder 0 oder 1.

Die Gerätenummer für die Floppy ist normalerweise 8, sie kann aber geändert werden.

Der Programmname kann, muß aber nicht gegeben werden. Er dient dazu, nur Dateien mit bestimmten Dateinamen aufzulisten. Es können zum Beispiel alle Dateien mit einem bestimmten Wortanfang ausgegeben werden. Natürlich können Sie "*" und "?" als »Joker« verwenden.

Außerdem kann noch nach Dateityp, also SEQ oder PRG aussortiert werden. In diesem Fall schreiben Sie den Befehl so:

```
LOAD"$L:NAME = T",G
```

L = Laufwerksnummer NAME = Programmname T = Typ G = Gerätenummer

Für den Dateinamen gilt die Rechtschreibung wie in 3.0. beschrieben.

Der Typ ist entweder:

S für SEQ (sequentiell)

P für PRG (Programm)

Beispiele:

```
LOAD"$0",8                    Lade das gesamte Inhaltsverzeichnis von Laufwerk 0 (rechts)
```

```
LOAD"$0:*",8                 wie vor
```

```
LOAD"$0:DAT*",8             wie vor, aber nur alle Dateien, die mit "DAT" beginnen
```

```
LOAD"$0:DAT*=P",8          wie vor, aber nur PRG-Dateien, die mit "DAT" beginnen
```

Geben Sie nun "LIST" ein, worauf der Rechner das Inhaltsverzeichnis auf den Bildschirm ausgibt.

Die Kopfzeile gibt die Laufwerksnummer, den Namen und die ID der Diskette an.

Darauf folgen die Dateinamenzeilen, in denen die Anzahl der belegten Blöcke, der Dateiname und der Dateityp steht.

Die letzte Zeile gibt die Anzahl der noch nicht belegten Blöcke an.

HINWEIS: "LIST 50-100" ist hier sinnlos, da die Zeilennummern nicht wie in einem BASIC-Programm nach aufsteigenden Zeilennummern geordnet sind.

ACHTUNG: Durch das Laden des Inhaltsverzeichnisses in den Rechner, wird ein eventuell im BASIC-Speicher abgelegtes Programm überschrieben und dadurch unbrauchbar gemacht.

Dies können Sie mit dem im nächsten Kapitel beschriebenen Programm vermeiden.

2.2.2. ... das kann auch ein Programm für Sie tun

Durch das 'LOAD"\$L:N",G' geladene Inhaltsverzeichnis wird wie ein Programm in den BASIC-Speicher des Rechners geladen. Dadurch wird eine Analyse des Inhaltsverzeichnisses durch ein Programm unmöglich.

Es gibt jedoch die Möglichkeit, sich das Inhaltsverzeichnis mittels GET# von der Floppy übergeben zu lassen. Die Bytes können dann in Variablen gespeichert werden, und so vom Programm analysiert werden. Es kann zum Beispiel anhand der Kopfzeile überprüft werden, ob die richtige Datendiskette eingelegt wurde, es kann überprüft werden, ob noch genügend Blöcke für eine weitere SEQ-Datei vorhanden sind, oder ob ein bestimmtes Programm auf einer Diskette vorhanden ist.

Für die Übergabe des Inhaltsverzeichnisses müssen der Floppy zuerst folgende Anweisungen gegeben werden:

```
OPEN 15,8,15 (falls noch nicht geschehen)
PRINT#15,"m-e"chr$(212)chr$(237)
OPEN FN,8,0,"$" + LN$ + ":" + V$
FN = Filenummer; LN$ = Laufwerksnummer
```

Der Zusatz von ":" + V\$ bewirkt, daß nur die Dateien übergeben werden, die dem in V\$ angegebenen Muster entsprechen.

Nun kann das Inhaltsverzeichnis bytewise über GET#FN (FN = Filenummer wie bei OPEN FN,8,0,"\$" + LN\$ + ":" + V\$) abgerufen werden. Dabei werden pro Zeile des Inhaltsverzeichnisses folgende Bytes übergeben:

1. 2 Bytes ohne Bedeutung
2. 2 Bytes, die in LOW/HIGH-Darstellung eine Zahl ergeben
3. 1–3 Blanks
4. Maximal 18 Bytes, die den Dateinamen enthalten. Das erste und letzte Byte sind Anführungsstriche (chr\$(34))
5. 1 oder mehrere Blanks bis zur Typenbezeichnung
6. 3 Bytes Typenbezeichnung (PRG oder SEQ)
7. 2–4 Blanks
8. 1 Leerstring als Ende der Zeile.

1. Die Kopfzeile

Sie beginnt mit weiteren 2 Bytes, die ebenfalls ohne Bedeutung sind.

Zu 2. Die in LOW/HIGH-Darstellung stehende Zahl hat hier keine Bedeutung.

Zu 3. Hier steht nur ein RVS-ON (CHR\$(18)).

Zu 6. Hier steht anstelle der Typenbezeichnung die ID der Diskette (zwei Zeichen plus ein Blank).

2. Die Dateinamen-Zeile

3. Die Schlußzeile

Zu 2. Die Zahl gibt die Anzahl der insgesamt noch freien Blöcke an.

Zu 3. Die Blanks entfallen; dies kann als Kennzeichen der Schlußzeile verwendet werden.

4.-8. Entfällt; es steht nur der Text "BYTES FREE.", dem 13 Blanks und drei Leerstrings folgen (falls Sie beide Inhaltsverzeichnisse abgerufen haben, so folgen beim ersten Inhaltsverzeichnis 15 Blanks und ein Leerstring).

Jede Zeile ist so mit Blanks (siehe 3. und 7.) aufgefüllt, daß sie genau 32 Zeichen umfaßt.

Die Zahl unter 2. muß mit Hilfe der ASC-Funktion gesondert berechnet werden. Da der Rechner jedoch ein String mit dem Wert »0« wie ein Leerstring behandelt, und die ASC-Funktion für den Leerstring nicht definiert ist ("ILLEGAL QUANTITY ERROR"), muß der Leerstring vorher »abgefangen« werden.

```
10 GET#1,B$
20 Z = 0
30 IF B$<>" " THEN Z=ASC(B$)
40 GET#1,B$
50 IF B$<>" " THEN Z=Z+256*ASC(B$)
```

Beispielprogramm zum Laden des Inhaltsverzeichnisses vom Programm; das Inhaltsverzeichnis wird zeilenweise in A\$(I) gespeichert:

```
1 REM****LADEN DES INHALTSVERZEICHNISSES
2 REM****
3 REM****
4 REM****
5 K=0:DIMA$(152)
6 REM****
7 PRINT"LAUFWERK?"
8 GET LN$: IF LN$="" THEN 8
9 INPUT "MUSTER";V$
10 OPEN15,8,15
20 PRINT#15,"M-E" CHR$(212) CHR$(237)
30 OPEN2,8,0,"$" + LN$ + ":" + V$
50 GET#2,A$:GET#2,A$
60 L=1
```

```

100 GET#2,A$: GET#2,A$
105 A=0
110 GET#2,A$: IFA$<>""THEN A=ASC(A$)
115 GET#2,A$: IFA$<>""THEN A=A+256*ASC(A$)
120 A$(L)=STR$(A)+" "
130 GET#2,A$:A$(L)=A$(L)+A$:IFA$<>"" ANDL<>1 THEN K=1
150 GET#2,A$:A$(L)=A$+A$
160 IFA$<>""THEN 150
200 IFK<>1 THEN L=L+1:GOTO 100
210 CLOSE2:CLOSE 15
220 REM***
230 REM***AUSDRUCK DES INHALTSVERZEICHNISSES***
240 REM***
300 FOR I=1 TO L
310 PRINT A$(I)
320 NEXT I
330 END
READY.

```

3. Die Floppy wird rumkommandiert

Außer "SAVE" und "LOAD" stehen noch einige andere Beispiele zur Vergügung, die dem Kopieren, Löschen und Umbenennen von Dateien und ganzen Disketten dienen.

Diese Befehle werden über "PRINT#" über den Kommando-/Fehlerkanal an die Floppy gegeben.

Beispiel:

```

OPEN 15,8,15
PRINT# 15,"I0"
CLOSE 15
"Initialisieren der Diskette in Laufwerk 0"

```

Zunächst eine Zusammenfassung der Befehle mit kurzen Erläuterungen und Hinweis auf den erklärbaren Abschnitt:

- 3.1. I Initialisieren der Diskette.
- 3.2. D Kopieren der Diskette.
- 3.3. C Kopieren von Dateien.
- 3.4. R Ersetzen eines Dateinamens durch einen anderen.
- 3.5. S Löschen einer Datei.
- 3.6. V Überprüfen und bereinigen der Diskette, BAM aktualisieren und alle mit "*" gekennzeichneten Dateien löschen.
- 3.7. N Löschen und Umbenennen der Diskette; wenn die Identifikation angegeben ist auch neu formatieren.

Bei den meisten Befehlen folgen noch die Dateinamen, auf die das Kommando angewendet werden soll.

Beispiel:

```
"C0:NEUDATEI=1:ALTDATI"
```

Bei allen Befehlen an die Floppy ist zu beachten, daß der gesamte Befehl (mit Dateinamen etc., also der Kommandostring) die Länge von 40 Zeichen nicht überschreiten darf!

Sollten Sie ein Kommando an die Floppy geben, das länger als 40 Zeichen ist, so wird die Fehlerlampe aufleuchten, und Sie können vom Kommando-/Fehlerkanal abrufen:

```
32 SYNTAX ERROR 0000
```

Näheres erfahren Sie in 8.

Außerdem dürfen Sie in einem "PRINT#" nicht mehr als ein Floppy-Kommando unterbringen.

3.0. Das müssen Sie bei Dateinamen beachten

Die folgenden Erklärungen beziehen sich auf die Kapitel 3.3. bis 3.5., aber auch auf alle folgenden Kapitel, in denen es um das Laden, Speichern und Überprüfen von Dateien geht.

Für Dateien dürfen Sie alle Zeichen verwenden, bis auf folgende Sonderzeichen:

- = trennt Teile eines Befehls
- : trennt Dateinamen
- , trennt hinzugefügte Teile vom Befehl
- ;
- ;
- ;

und die »Jokerzeichen«:

- ? Steht für einen beliebigen Buchstaben.
- * Steht für ein beliebiges Wortende.

Sollten Sie ein derartiges Zeichen in einem Dateinamen verwenden, so wird die Fehlerlampe aufleuchten, und Sie können vom Kommando-/Fehlerkanal folgende Fehlermeldung abrufen:

33 SYNTAX ERROR 0000

Es kann jedoch auch geschehen, daß die Floppy das Inhaltsverzeichnis durcheinanderbringt, zum Beispiel Dateien im Inhaltsverzeichnis erzeugt werden, die weder existieren, noch gelöscht werden können!

3.0.1. Die »Jokerzeichen«

Bei einigen Befehlen können Sie die Dateinamen mit dem »Jokerzeichen« abkürzen. Es gibt folgende Jokerzeichen:

- ? Steht für genau ein beliebiges Zeichen (auch für " ", also das Leerzeichen).
- * Steht für ein beliebiges Wortende, unabhängig von dessen Länge.

"?" paßt nur auf ein vorhandenes Zeichen, "*" paßt auch auf ein nicht vorhandenes Wortende. "*" darf nur am Ende des Dateinamens stehen.

Beispiele:

Muster	entspricht:	entspricht nicht:
"DAT*"	"DATEI" "DATEN" "DAT"	"DA" "DAS" " DATEI"
"AB??EF*"	"ABCDEF" "ABCDEF*" "ABXYEFAAA"	"AB" "ABXXEFG"
"AB??EF"	"ABCDEF" "ABXxEF" "AB EF"	"ABCDEF " " ABCDEF"
"DAT??*"	"DATEI" "DATEN" "DAT PROG*." "DATEI PROG*." "DAT "	"DAT" "DATE" " DAT"
"DAT??"	"DATEN" "DATEI"	"DATEN1" "DAT"

Bei den verschiedenen Floppybefehlen ergeben sich zwei Interpretationsmöglichkeiten durch die Floppy:

a) Der Befehl wirkt auf jede Datei, die dem angegebenen Namensmuster entspricht. Dies geschieht bei den Befehlen:

LOAD"\$..." Inhaltsverzeichnis laden.
PRINT#15,"S..." Scratch

b) Der Befehl wirkt nur auf die zuerst gefundene Datei, die dem angegebenen Namensmuster entspricht. Dies geschieht bei folgenden Befehlen:

VERIFY"... " Vergleichen eines Programms im Rechner mit einem Programm auf der Diskette.
LOAD"... " Laden eines Programms von einer Diskette in den Rechner.
.L"...",... Wie "LOAD", aber vom Monitor.

'LOAD"*",8' und 'VERIFY"*",8' haben eine besondere Bedeutung, die Sie aus 4.2. bzw. 4.3. entnehmen können.

In den übrigen Befehlen dürfen Sie keine »Jokerzeichen« verwenden:

R	Rename
C	Copy
N	New
SAVE"..."	Speichern von Programmen.
.S"..."	Wie "SAVE", aber vom Monitor.

3.1. Zuerst muß die Floppy die Diskette kennen

Bevor die Floppy mit einer Diskette arbeiten kann, muß die Diskette initialisiert werden.

Beim Initialisieren geschieht folgendes:

1. Der Schreib-Lesekopf wird auf Spur 1 gefahren und auf sie eingestellt.
2. Die BAM und die ID (siehe 2.2., a.),) werden gelesen und in den Arbeitsspeicher der Floppy geschrieben.

Damit ist die Diskette gewissermaßen der Floppy »vorgestellt«.

Wenn Sie auf eine Diskette in der Floppy zugreifen, die noch nicht initialisiert wurde, so leuchtet die Fehlerlampe auf, und Abfragen der Fehlermeldung ergibt:

```
29 DISK ID MISMATCH SP SE
```

Es gibt nur folgende drei Möglichkeiten:

- "I0" Initialisiere die Diskette in Laufwerk 0
- "I1" Initialisiere die Diskette in Laufwerk 1
- "I" Initialisiere die Diskette in beiden Laufwerken.

ACHTUNG: Das Initialisieren schließt alle bereits geöffneten Kanäle (bis auf Kanal 15) in der Floppy! Öffnen Sie daher erst nach dem Initialisieren!

3.2. Sie verdoppeln eine ganze Diskette: D

Mit dem Befehl »D« können Sie eine vollständige Kopie einer Diskette anfertigen.

Obwohl der Kopiervorgang ca. 80 Sekunden dauert, ist es ratsam, von jeder Diskette, mit der gerade gearbeitet wird, eine Kopie anzufertigen, da Sie bei einem Fehler dann nur die zuletzt gemachte Arbeit zu rekonstruieren brauchen, während sonst unter Umständen eine ganze Programmbibliothek verloren gehen kann.

Hinweis: Damit eine Verwechslung von zu kopierender und Kopierdiskette nicht die zu kopierende Diskette überschreibt, ist es ratsam, die zu kopierende Diskette mit dem SCHREIBSCHUTZ zu schützen.

Es handelt sich hierbei um die selbstklebenden, 1 x 2 cm großen Etiketten, die Sie bei Ihrem Commodore-Händler erhalten können. Sie werden über den Einschnitt an der Seite der Disketten geklebt und bewirken, daß diese Diskette von der Floppy nur gelesen, aber nicht mehr beschrieben werden kann.

Schreibweise:

Es gibt nur folgende zwei Möglichkeiten:

- "D0=1" Schreibe eine Kopie der Diskette in Laufwerk 1 auf die Diskette in Laufwerk 0.
- "D1=0" Schreibe eine Kopie der Diskette in Laufwerk 0 auf die Diskette in Laufwerk 1.

3.3. ... oder nur eine Datei: C

Mit dem Befehl »C« können Sie Kopien von Dateien anfertigen. Dabei kann die Kopie sowohl auf die Diskette, auf der die Originaldatei steht, als auch auf die andere Diskette geschrieben werden.

Schreibweise:

```
"C L:Kopie=L:Original"
```

L=Laufwerksnummer Kopie=Name der zu L=Laufwerksnummer Original=Name der bereits
der zu erstellenden Datei erstellenden Datei der bereits vorhand. Datei vorhandenen Datei

ACHTUNG: Der gesamte Kommandostring darf nicht länger sein als 40 Zeichen! Keiner der Dateinamen darf ein »Jokerzeichen« enthalten (siehe 3.0.1.)!

3.4. Eine Datei wird umbenannt: R

Mit dem Befehl R können Sie den Namen einer Datei ändern. Diese Operation ändert nur das Inhaltsverzeichnis.

Schreibweise:

```
"R L: neuer Name = L: alter Name"
```

Obwohl die Laufwerksnummern natürlich gleich sind, müssen beide angegeben werden.

Beispiele:

```
"R0:DATEI=0:PROBE"
```

Falls bereits eine Datei mit dem Namen "DATEI" existiert, leuchtet an der Floppy die Fehlerlampe auf, und eine Abfrage des Fehlers ergibt:

```
63 FILE EXISTS 00 00
```

3.5. So entfernen Sie Dateien, die Sie nicht mehr brauchen können: S

Mit dem Befehl S können Sie eine oder mehrere Dateien löschen. Normalerweise löschen Sie, indem Sie den (die) vollen Dateinamen angeben. Falls Sie Jokerzeichen (siehe 3.0.1.) verwenden, so werden alle Daten, die dem angegebenen Namensmuster entsprechen, gelöscht!

Die Floppy löscht die Dateien, indem sie den (die) Namen im Inhaltsverzeichnis löscht und alle von ihr (Ihnen) belegten Blöcke in der BAM freigibt. Der Inhalt der Datei selbst wird nicht gelöscht, er könnte mit den Mitteln des Direktzugriffs noch gelesen werden.

Es kommt vor, daß beim Löschen nicht alle betroffenen Blöcke freigegeben werden. Dies kann mit dem Befehl V nachträglich vollendet werden.

Schreibweise:

```
"S L:Dateiname"
```

L=Laufwerksnummer

Jede weitere zu löschende Datei wird durch ", " angeschlossen:

```
"S L:1.Dateiname,L:2.Dateiname,L:3.Dateiname"
```

ACHTUNG: Der Kommandostring darf nicht länger als 40 Zeichen sein.

Beispiele:

```
OPEN15,8,15
```

```
PRINT#15, "SØ: TEST"
```

Lösche die Datei "TEST" im Laufwerk 0

```
Print#15,"S0:TEST,1:ALTDAT,0:PROBE"
```

Wie oben, jedoch zusätzlich "ALTDAT" auf Laufwerk 1 und PROBE auf Laufwerk 0.

```
Print#15,"S1:DAT*"
```

Lösche ALLE Dateien, deren Name mit "DAT" beginnt.

ACHTUNG:

```
Print#15,"S0:*"
```

löscht nicht die erste, sondern ALLE Dateien auf Laufwerk 0.

3.6. Kontrolle und Bereinigen der Diskette

Um eine Diskette, auf der sich Fehler eingeschlichen haben, zu überprüfen, und zu bereinigen, verwenden Sie den Befehl V. Er ändert das Inhaltsverzeichnis und die BAM so, daß sie mit dem Inhalt der Diskette übereinstimmen. Außerdem werden alle Dateien, die wegen eines Fehlers beim Speichern oder Löschen mit einem "*" gekennzeichnet wurden.

Der Befehl V initialisiert automatisch die Diskette, so daß eine noch nicht initialisierte Diskette vorher nicht initialisiert werden muß.

Schreibweise:

```
"V L"
```

L = Laufwerksnummer

Falls Sie die Laufwerksnummer weglassen, wird die Diskette bereinigt, auf die zuletzt zugegriffen wurde.

Beispiele:

```
OPEN 15,8,15 (falls nötig)
```

```
Print#15,"V"
```

Überprüfe und berichte die Diskette, auf die zuletzt zugegriffen wurde.

```
Print#15,"V1"
```

Überprüfe und berichte die Diskette in Laufwerk 1.

Der Schreibschutz darf »nicht« geklebt sein, da "V" die BAM und das Inhaltsverzeichnis auf der Diskette berichtigen muß.

ACHTUNG: Jeder Lese- oder Schreibfehler unterbricht den Berichtigungsvorgang. Dadurch entstehen unter anderem Differenzen zwischen der BAM im Speicher der Floppy und der BAM auf der Diskette. Die BAM wird namentlich zuerst im Speicher der Floppy berichtigt und erst nach Abschließen der Überprüfung auf die Diskette übertragen.

Initialisieren Sie bei Auftreten eines Fehlers auf jeden Fall neu, damit die Differenzen zwischen beiden BAM's aufgehoben werden und nicht bei "CLOSE" die unvollständig berichtigte BAM aus dem Speicher der Floppy auf die Diskette geschrieben wird.

Nach dem Initialisieren können Sie dann noch einmal die Überprüfung wiederholen.

3.7. Das Ende aller Daten, das Löschen: N

Für den Befehl N gibt es zwei Anwendungen:

- a. Sie können eine ganze Diskette löschen,
- b. Noch nicht benutzte Disketten können zusätzlich zum Löschen noch neu formatiert werden. Das heißt, alle Blöcke werden mit (neuen) Blockköpfen (blockheader) versehen. Die Blockheader dienen der Floppy zu eindeutiger Identifikation jedes Blockes auf der Diskette.

Sie löschen eine Diskette, indem Sie nach dem »N« nur Laufwerksnummer und den neuen Diskettennamen, getrennt durch einen Doppelpunkt, angeben. Der neue Diskettenname ersetzt den alten, die alte ID (siehe 2.2.a.) bleibt erhalten.

Schreibweise:

"N L:Diskettenname"

Beispiel:

```
OPEN 15,8,15  
PRINT#15,"N0:LAGERDATEN"
```

Lösche die Diskette in Laufwerk 0 und gib ihr den Namen "LAGERDATEN". Die alte ID bleibt erhalten.

Das Löschen ohne Formatieren dauert ca. 20 Sekunden.

Sie löschen und »formatieren« eine Diskette, indem Sie nach dem N und dem Diskettennamen die ID eingeben. Sie ist zwei Zeichen lang. Falls Sie nur ein Zeichen eingeben, so ist das zweite automatisch "CR" (chr\$(13)).

Falls die Diskette schon einen Namen und ID hatte, so werden sie durch die im N-Befehl angegebenen ersetzt.

Schreibweise:

"N L:Diskettenname,ID"
L = Laufwerksnummer

Beispiel:

```
OPEN 15,8,15  
PRINT#15,"N0:PERSONALDATEN,PD"
```

Lösche und formatiere die Diskette im Laufwerk 0, versehe sie mit dem Namen "PERSONALDATEN" und der ID "PD".


Löschen mit Formatieren dauert ca. 80 Sekunden.

WICHTIGER HINWEIS: Der Inhalt einer mit N gelöschten Diskette ist unwiderbringbar verloren. Wenden Sie deshalb den N-Befehl mit äußerster Vorsicht an!

Falls Sie feststellen, daß Sie aus Versehen "N" gegeben haben, öffnen Sie SOFORT die Laufwerksklappen an der Floppy. Dadurch entsteht kein Schaden an Diskette oder der Floppy. Falls Sie Glück haben, hatte die Floppy den Löschvorgang noch nicht begonnen. Ansonsten können Sie versuchen, die Blöcke, die noch nicht gelöscht wurden, mit "b-r" (siehe 7.7.) zu lesen.

So speichern Sie ein Programm

4.0. Wozu?

Der Normalfall einer PRG-Datei ist ein BASIC-Programm. Es wird nach seiner Erstellung einmal abgespeichert, es kann aber auch mit  immer wieder verändert werden, z. B., wenn es seit seiner letzten Speicherung verbessert worden ist. Ist ein Programm erst einmal auf Diskette gespeichert, so kann man darauf in Sekunden zugreifen, ohne lange Suchzeiten, wie beim Kassettenrecorder, in Kauf nehmen zu müssen.

Sollte die Diskette jedoch schon sehr stark belegt sein, vor allem mit verstreuten, durch Direktzugriff belegten Blöcken, so kann sich die Zeit zum Abspeichern oder Laden eines Programms stark erhöhen, da der Schreib-/Lesekopf viele Diskettenspuren »abgrasen« muß, bis er freie Blöcke für das Programm

findet. Sollte die Zeit für das Lesen oder Schreiben eines Programms zum Problem werden, so hat man zwei Möglichkeiten:

1. Möglichkeit:

Man verteilt die Daten dieser Diskette auf mehrere andere Disketten und löscht dann auf der Ursprungsdiskette alle nun kopierten Dateien. Dieses Verfahren ist das wirkungsvollere, es erhöht aber den Aufwand an Disketten und kann dazu führen, daß inhaltlich zusammengehörende Dateien (z. B. eine Sammlung von Programmen, die zur Lösung desselben Problems dienen) getrennt und die Sammlung dadurch unübersichtlich wird.

2. Möglichkeit:

Man legt alle durch Direktzugriff belegten Blöcke nahe zusammen. Vor allem ist dabei zu empfehlen, die ersten Spuren einer Diskette zu verwenden (z. B. reichen für 50 Blöcke, die im Direktzugriff benutzt werden sollen, die Spuren 1–3). Dieses Verfahren sollte vor allem verwandt werden, wenn viele Blöcke im Direktzugriff benutzt werden sollen, aber außerdem auf dieser Diskette noch Programme gespeichert werden müssen.

Allgemein sollte man jedoch eine Diskette ENTWEDER für PRG- und SEQ-Dateien, ODER für Speicherung von Daten im DIREKTZUGRIFF verwenden, da z. B. bei einem falschen Direktzugriff-Schreibbefehl unter Umständen ein ganzes Programm wertlos gemacht werden kann, wenn der Anfang des Programms dabei überschrieben worden ist.




4.1. Das Speichern eines BASIC-Programms: "SAVE"

Mit dem BASIC-Kommando "SAVE" können Sie ein BASIC-Programm auf Diskette abspeichern. Sie können dabei wählen, auf welches Laufwerk Sie das Programm bringen wollen, und ob Sie mit dem Programm ein bereits auf der Diskette gespeichertes Programm gleichen Namens ersetzen lassen wollen. Dabei ist es gleichgültig, ob das zu speichernde Programm kürzer oder länger als das bereits auf der Diskette abgespeicherte Programm ist. Die Floppy sucht sich selbständig freie Blöcke der Diskette, auf denen sie das Programm abspeichern kann.

Die Form des "SAVE"-Befehls:

SAVE"L:NAME",G


zum Überschreiben L = Laufwerksnummer NAME = Programmname G = Gerätenummer

Das  kann weggelassen werden. Es dient dazu, der Floppy mitzuteilen, daß dieses Programm ein anderes Programm mit gleichem Namen überschreiben soll. Findet die Floppy bei einem  kein Programm dieses Namens, so tritt keine Fehlermeldung auf, sondern das Programm wird abgespeichert, als ob kein  vor der Laufwerksnummer gegeben worden wäre.

Die Nummer des Laufwerks sollte auf jeden Fall angegeben werden! Geschieht das nicht, so wird die Floppy im Normalfall das Programm auf das Laufwerk schreiben, das zuletzt benutzt worden ist. Dabei kann es jedoch vorkommen, daß bei einem nachfolgenden Zugriff die Inhaltsverzeichnisse der Disketten verwechselt werden, was dazu führt, daß mindestens eine der beiden Disketten dann nicht mehr ohne Löschen verwendbar ist!

Der Programmname muß nach den bekannten Regeln für Dateinamen (siehe 3.0.0.) gebildet werden. Dabei können Sie die »Jokerzeichen« ? und * nicht verwenden, da die Floppy ja dann nicht weiß, wie das Programm genannt werden soll.

Die Gerätenummer ist hier die Kenn-Nummer der Floppy, sie ist normalerweise 8 (siehe 1.1.0.1.1.).

Beim Gebrauch des "SAVE"-Kommandos kann es vorkommen, daß Sie versuchen, ein Programm ohne den Gebrauch des  abzuspeichern, obwohl bereits ein Programm gleichen Namens auf der Diskette existiert. In diesem Fall wird die Fehlerlampe an der Floppy aufleuchten, und Sie können über den Kommando/Fehlerkanal die Fehlermeldung "FILE EXISTS" (Datei bereits vorhanden) abrufen. Näheres über die Fehlermeldungen können Sie in Kapitel 9 nachlesen.

Sollten während des Abspeicherns eines Programms Fehler auftreten oder das Abspeichern unterbrochen werden, so erscheint das Programm im Inhaltsverzeichnis der Diskette, obwohl das Programm nicht oder nur teilweise abgespeichert wurde. In diesem Fall ist die Information über den Dateityp mit einem Stern gekennzeichnet. Sie lautet also: "*PRG". Mit dem "VERIFY"-Kommando der Floppy (siehe 3.5.) können Sie in diesem Fall das Inhaltsverzeichnis berichtigen und evtl. gespeicherte Teile des Programms löschen.

Damit dabei das Programm nicht verlorengeht, empfiehlt es sich, es vorher auf einer anderen Diskette abzuspeichern!

4.2. ... Und so bekommen Sie Ihr Programm wieder zurück: "LOAD"

Im vorigen Kapitel haben Sie gelernt, wie Sie ein Programm, das Sie in den Rechner eingegeben haben, auf eine Diskette abspeichern können. Das tun Sie natürlich nur, um das Programm auch wieder in den Rechner laden zu können, um es dann zu verwenden. Das geschieht mit dem BASIC-Befehl "LOAD". Ein LOAD-Befehl ist so aufgebaut:

```
LOAD"L:NAME",G
```

L = Laufwerksnummer, G = Gerätenummer Name = Programmname

Beim LOAD-Befehl kann die Laufwerksnummer entfallen. Die Floppy sucht das Programm automatisch auf der zuletzt benutzten Diskette. Steht es dort nicht im Inhaltsverzeichnis, wechselt die Floppy automatisch das Laufwerk und sucht im Inhaltsverzeichnis der dort befindlichen Diskette.

Das vereinfachte "LOAD" sieht dann so aus:

```
LOAD"NAME",G
```

Für die Gerätenummer gilt das schon im SAVE-Befehl erläuterte (siehe auch Kapitel 1.1.0.1.1.).

Natürlich können Sie beim "LOAD" die Zeichen '?' und '*' als »Joker« verwenden. Ihre Wirkung ist in Kapitel 3.0.0. erklärt.

Eine weitere Vereinfachung ist der Befehl LOAD"*",G. Er führt, wenn er als erster Befehl nach dem Einschalten gegeben wird, folgendes durch:

1. Diskette 0 wird initialisiert (vgl. 3.0.).
2. Von Diskette 0 wird das erste Programm, das im Inhaltsverzeichnis steht, geladen.

Sollten Sie versuchen, ein Programm zu laden, das nicht im Inhaltsverzeichnis der Diskette steht, also nicht gespeichert worden ist, so leuchtet die Fehlerlampe auf, und Sie können vom Fehler-/Kommando-kanal die Meldung: "FILE NOT FOUND" (Datei nicht gefunden) einlesen. Dafür ist in den meisten Fällen ein Tippfehler beim Programmieren oder die Vertauschung von Disketten verantwortlich. Näheres erfahren Sie in Kapitel 8.

4.3. Sicherheitshalber kontrollieren: "VERIFY"

Der VERIFY-Befehl vergleicht das im Rechner gespeicherte Programm mit einem auf der Diskette befindlichen Programm. Er dient vor allem dazu, nach dem Abspeichern eines Programms mit "SAVE" zu überprüfen, ob das Programm auch absolut richtig gespeichert worden ist. Man kann mit ihm aber auch feststellen, ob mehrere Versionen eines Programms identisch sind. Dabei geschieht folgendes: Der Rechner vergleicht das gespeicherte Programm Byte für Byte mit dem Programm auf der Diskette. Sind alle Bytes identisch, d. h. die Programme stimmen vollkommen überein, so erscheint auf dem Bildschirm die Meldung: "OK READY."

Sollte auch nur ein Byte unterschiedlich sein, so gibt der Rechner die Meldung: "?VERIFY ERROR"!

Das heißt, daß beim Vergleich zweier Programme eine vollkommen unwichtige Änderung, z. B. das Einfügen eines Leerzeichens, schon zum "VERIFY ERROR" führt.

So wenden Sie "VERIFY" an:

```
VERIFY"L:NAME",G
```

L = Laufwerksnummer, G = Gerätenummer, NAME = Programmname

Wie bei "LOAD" kann auch hier die Laufwerksnummer entfallen. Sie können hier analog zu "LOAD" die Zeichen "?" und ":" als Joker verwenden.

Der Befehl VERIFY"*",G hat jedoch eine besondere Wirkung: Er vergleicht das Programm im Rechner mit dem zuletzt abgespeicherten Programm.

Diesen Befehl sollten Sie nach jedem "SAVE"-Befehl anwenden!

4.4. So speichern Sie ein Maschinenprogramm: .S

Der Befehl .S des RESIDENT-Monitors ist in der Lage, einen zusammenhängenden Speicherbereich auf einer Diskette zu speichern. Dazu müssen Sie zunächst den Monitor mit SYS 1024 ansprechen.

Der .S-Befehl sieht so aus:

```
.S" L:NAME",G,A,E
```

Zum Über-	L=Laufwerks-	NAME=Pro-	G=Geräte-	A=Anfangs-	E=End-
schreiben	nummer	grammname	nummer	adresse	adresse

Das @ bewirkt wie beim BASIC-Kommando SAVE, das heißt, ein bereits bestehendes Programm dieses Namens wird überschrieben. Auch die Fehlermeldungen treten genau wie nach dem SAVE-Kommando auf.

Auch bei der Laufwerksnummer gleichen sich .S und SAVE: sie sollte auf jeden Fall angegeben werden. Und natürlich dürfen Sie im Programmnamen nicht "*" und "?" als »Jokerzeichen« verwenden.

Die Gerätenummer muß zweistellig und hexadezimal sein, das heißt, sie ist im Normalfall 08.

Die Anfangsadresse gibt an, ab welcher Speicherzelle der Rechner speichern soll, sie muß vierstellig und hexadezimal sein. Der Rechner hört mit dem Speichern beim letzten Byte vor der Endadresse auf. Also müssen Sie die Endadresse um 1 größer wählen, als die echte Länge des Speicherbereichs. Wollen Sie z. B. ein Maschinenprogramm speichern, das von 0600 bis 083B (natürlich hexadezimal) reicht, so muß die Anfangsadresse 0600 und die Endadresse 083C sein.

4.5. ... und laden es wieder: .L

Das Laden eines Maschinenprogramms geschieht wie beim normalen LOAD-Befehl, seine Syntax ist in 4.2. erklärt.

Sie können aber auch den L.-Befehl des Monitors verwenden. Er sieht so aus:

```
.L "LW:Name",G
```

LW = Laufwerksnummer Name = Programmname G = Gerätenummer (normalerweise 8)

Der Dateiname sieht genauso aus, wie im normalen "LOAD" (siehe 4.2.).

5. Die Datenkette; die SEQ-Datei

Eine sequentielle Datei ist eine Folge von Variableninhalten (zum Beispiel Daten, die Sie in einem Programm errechnet haben). Wie der Name schon besagt, haben diese Daten eine bestimmte Reihenfolge (lat.: sequi = folgen), die Sie nicht umgehen können.

Im Gegensatz zu einer PRG-Datei handelt es sich also nicht um zusammenhängende Datenblöcke (Programmname), die beim Laden zusammenhängend ab einer bestimmten Position in den Programmspeicher des Rechners gesetzt werden, sondern um eine Kette aus einzelnen Daten, die Sie in Variable einlesen müssen.

5.0. Das müssen Sie sich bei SEQ-Dateien alles überlegen

Schon beim Öffnen einer Datei müssen Sie sich entscheiden, ob Sie

- eine PRG- oder eine SEQ-Datei eröffnen wollen. Dabei ist der Dateityp meist sinnlos, da PRG-Dateien viel einfacher mit den Befehlen "LOAD" und "SAVE" behandelt werden können.
- auf eine SEQ-Datei schreiben oder von ihr lesen wollen.
- eine Datei gleichen Namens überschreiben wollen (nur bei Print#)

5.1. Der OPEN-Befehl...

... sieht dann folgendermaßen aus:

```
OPEN LA,G,KN," L:Name,T,M"
```

LA = logische Adresse, unter der die Datei eröffnet werden soll

G = Gerätenummer (bei Ihrer Floppy normalerweise 8)

KN = Kanalnummer (Sie können Ihrer Datei einen Kanal zwischen 2 und 14 zuordnen)
= zum Überschreiben

L = Laufwerksnummer (0 oder 1)

Name = Dateiname

T = Typ (PRG oder SEQ)

M = Modus (READ oder WRITE = Lesen oder Schreiben)

Zu LA: Die logische Adresse, unter der Sie eine Datei im Rechner eröffnen, können, kann zwischen 0 und 255 liegen (siehe 1.1.0.0.).

Zu G: Mit der Gerätenummer entscheiden Sie, welches Peripheriegerät Sie ansprechen wollen. Ihre Floppy hat normalerweise die Gerätenummer 8.

Zu KN: Sie müssen der Datei auch in der Floppy einen Kanal zuordnen. KN muß zwischen 2 und 14 liegen (siehe 1.1.0.0.).

Zu: ☉ Wollen Sie auf eine Datei schreibend zugreifen, die bereits existiert, so müssen Sie mit dem ☉ -Zeichen der Floppy mitteilen, daß die alten Daten überschrieben werden sollen.

Zu L: L ist die Nummer des Laufwerks, auf das Sie zugreifen wollen.

Zu Name: Name ist der Name der Datei, auf die Sie zugreifen wollen. Näheres zu den Regeln von Dateinamen erfahren Sie in 3.

Zu T: In T wird angegeben, ob es sich um eine PRG- oder SEQ-Datei handelt. PRG können Sie mit P und SEQ mit S abkürzen.

Zu M: Eröffnen Sie eine SEQ-Datei, so müssen Sie angeben, ob Sie auf die Datei lesend oder schreibend zugreifen wollen. Wollen Sie lesend zugreifen, so müssen Sie in M "R" wie "READ" (lesen) angeben, im anderen Fall "W" wie "WRITE" (schreiben).

Folgende Punkte müssen Sie unbedingt beachten:

1. Sowohl beim lesenden, als auch beim schreibenden Zugriff muß der Dateityp angegeben werden! Wird er weggelassen oder falsch angegeben, so kann die Floppy die Datei nicht finden.
2. Nur beim lesenden Zugriff dürfen Sie im Dateinamen die Jokerzeichen (siehe 3.0.1.) verwenden.
3. Sie können zwar mit mehreren Dateien lesend auf eine Datei zugreifen, aber nur mit einer Datei schreibend!

Im Fehlerfall leuchtet die Fehlerlampe der Floppy auf, und über den Kommando-/Fehlerkanal kann die Meldung:

```
60 WRITE FILE OPEN 00 00  
abgerufen werden.
```

4. Achten Sie unbedingt darauf, in der Floppy nicht mehr als 5 Datenkanäle offenzuhalten! Bei Überschreiben dieser Zahl leuchtet an der Floppy die Fehlerlampe auf und über den Kommando-/Fehlerkanal können Sie die Fehlermeldung:

```
70 NO CHANNEL 00 00  
abfragen.
```

Schließen Sie jede Datei sofort nach ihrer Verwendung mit "CLOSE". Wenn Sie vergessen, eine SEQ-Datei zu schließen, wird sie in der Regel unbrauchbar.

5.2. Das Schreiben von Daten in eine SEQ-Datei: PRINT#

Nachdem Sie eine Datei als »Schreibdatei« eröffnet haben, können Sie mit "PRINT#" darauf schreibend zugreifen. Der PRINT#-Befehl ist in 1.1.2. genau erläutert. Hier soll nur eine kurze Zusammenfassung gegeben werden.

Der PRINT#-Befehl sieht so aus:

```
PRINT#LA,Variable;CHR$(13);Variable;CHR$(13);...;CHR$(13);
```

LA ist die logische Adresse der Datei, die Sie im Rechner eröffnet haben.

"Variable" ist die Variable, deren Inhalt Sie auf Diskette schreiben wollen. Es können Strings-, Zahlen- oder Integervariable geschrieben werden.

Die CHR\$(13) (= CR) zwischen den Variablen sind notwendig, um die Variablen beim Einlesen zu trennen. Die Semikoli können nicht durch Kommata ersetzt werden, da sonst unnötige Leerzeichen auf der Diskette gespeichert werden.

Eine Variable darf nicht mehr als 80 Zeichen enthalten, wenn die Daten mit "INPUT#" gelesen werden sollen. Das ist wichtig für die Übertragung von Stringvariablen. Gegebenenfalls muß eine zu lange Stringvariable mittels "MID\$" und "LEFT\$" in mehrere kurze Variablen aufgeteilt werden.

Beispiel:

```
OPEN3,8,3,"0:PRIMZAHL,S,W"  
PRINT#3,A;CHR$(13);B;CHR$(13);C;CHR$(13);D.CHR$(13);
```

5.3. So lesen Sie von einer SEQ-Datei: INPUT#

Auch hier nur eine kurze Zusammenfassung des in 1.1.3. beschriebenen INPUT#-Befehls: Dieser Befehl sieht so aus:

```
10 INPUT#LA,Variable,Variable,...
```

LA ist die logische Adresse der im Rechner eröffneten »Lesedatei«.

Zur Trennung der Variablen muß ", " verwendet werden. Andere Zeichen sind nicht zulässig.

Die hinter "INPUT#" angegebenen Variablen müssen dem Typ (String-, Zahl-, Integervariable) nach in derselben Reihenfolge stehen, in der sie vorher mit "PRINT#" geschrieben worden sind; allerdings können Zahlenvariable auch in Stringvariable eingelesen werden. Wird jedoch versucht, Stringvariable in Zahlenvariable einzulesen, so erscheint:

```
?FILE DATA ERROR (IN...)
```

Es hat sich als günstig erwiesen, alle Daten zunächst in Stringvariablen einzulesen. Wenn nötig, können diese leicht mit VAL in Zahlenvariable umgewandelt werden.

Beispiel:

```
OPEN 4,8,4,"0:PRIMZAHL,S,R"  
INPUT#4,A$,B$,C$,D$  
A=VAL(A$):B=VAL(B$):C=VAL(C$):D=VAL(D$)
```

5.4. Zurück in kleinen Schritten: GET#

"GET#" werden Sie normalerweise nur verwenden, wenn Sie die Struktur der Daten, die Sie lesen wollen, nicht kennen; also nicht sicher sind, daß nach jeweils spätestens 80 Zeichen ein CR (Carriage return = CHR\$(13)) steht.

Es ist nach dem Einlesen durch "GET#" leicht möglich, die gelesenen Zeichen so zu Variablen zusammenzusetzen, wie es dem Format Ihrer Daten entspricht.

Die genaue Beschreibung von "GET#" finden Sie in 1.1.4., im Prinzip sieht der GET#-Befehl so aus:

```
GET#LA,Stringvariable
```

LA = logische Adresse, unter der die Lesedatei im Rechner eröffnet worden ist.

Statt der Stringvariablen kann unter Umständen auch eine Zahlenvariable verwendet werden (näheres siehe 1.1.4.).

WICHTIGER HINWEIS: CHR\$(0) wird als der leere String übernommen. Diese Eigenart kann durch folgende Routine vermieden werden:

```
10 GET#LA,G$:IFG$=""THEN G$=CHR$(0)
```

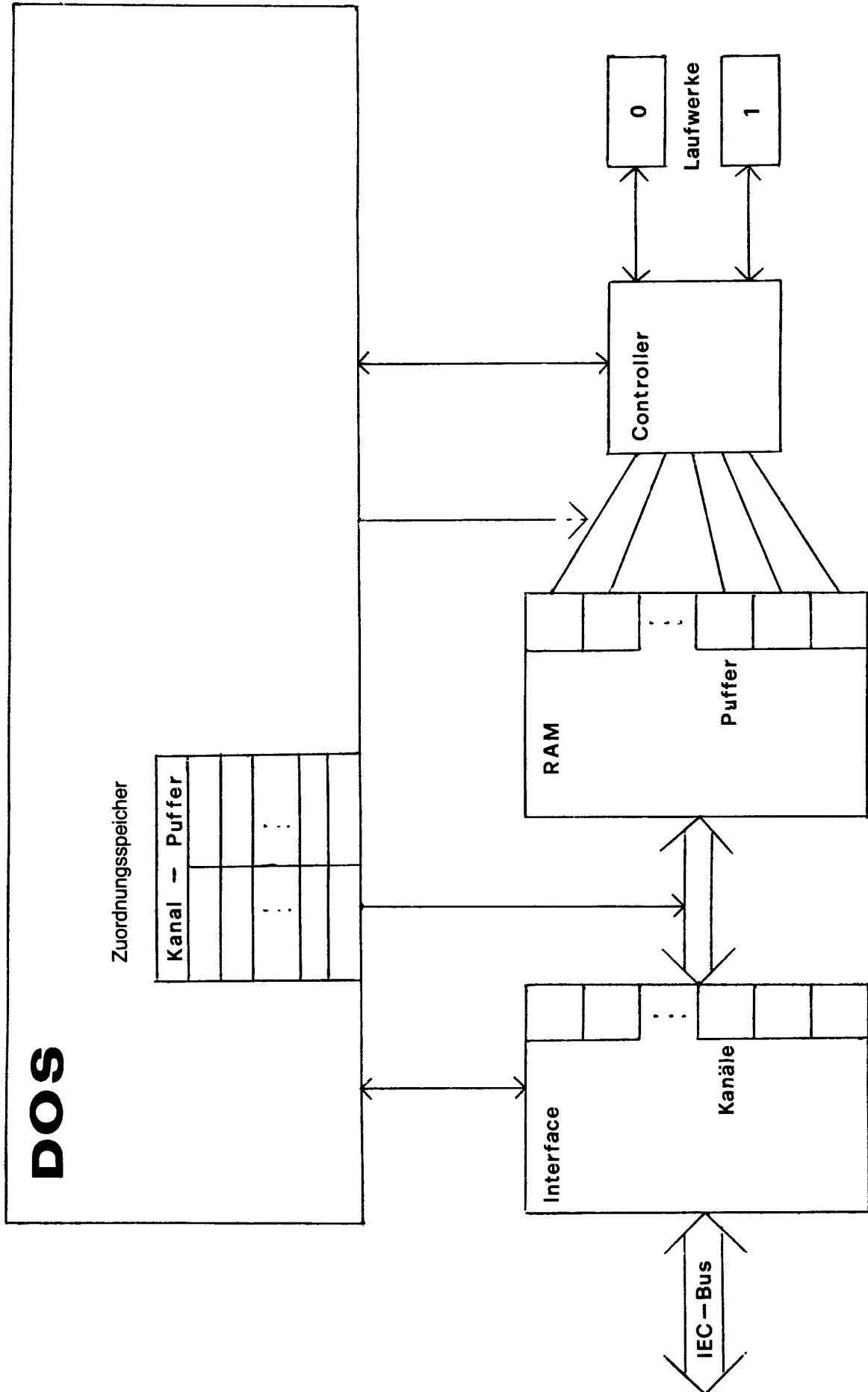
6. DOS-Organisation

In diesem Kapitel soll nur der grobe Aufbau der Floppy erklärt werden, so, wie er für das Verständnis der Vorgänge in der Floppy von Bedeutung ist.

6.0. Plan der DOS-Organisation

Die Floppy besteht im Groben aus folgenden Teilen:

Das DOS ist die zentrale Steuereinheit. Es kontrolliert das Interface, die Kanäle, die Verbindung vom Puffer zum Controller und den Controller selbst.



6.1. Pufferorganisation

Als interne Speicher verwendet die Floppy 16 Puffer (mit je 255 Bytes Länge), die Teil des RAM's der Floppy sind (siehe Plan). Puffer 0–12 werden als Zwischenspeicher für Daten, die vom Rechner auf Diskette, oder umgekehrt übertragen werden sollen (in Puffer 13 und 14 steht die aktuelle BAM von Laufwerk 0 und 1, Puffer 15 wird vom DOS verwendet).

Sollen Daten (sowohl aus PRG-, als auch aus SEQ-Dateien) auf Diskette geschrieben werden, so werden sie zunächst blockweise in den Puffer geschrieben und von dort auf die Diskette übertragen.

Auf dem umgekehrten Weg, also beim Laden von Daten in den Rechner, werden diese blockweise in den Puffer abgelegt und an den Rechner übertragen.

Wenn Sie mit den Befehlen des Direktzugriffs (siehe 7.) Daten auf Diskette speichern wollen, so müssen Sie denselben Weg benutzen. Beim Speichern schreiben Sie Ihre Daten zunächst in einen Puffer und lassen Sie von dort auf Diskette übertragen ("B-W", siehe 7.2.).

Beim Laden lassen Sie Ihre Daten zunächst von Diskette in einen Puffer schreiben, um sie dann von dort in den Rechner zu lesen.

6.2. Kanäle

Die Verbindung zwischen dem Rechner und den Puffern in der Floppy geschieht über Kanäle. Die Kanalstruktur der Floppy entspricht der Ihres Rechners: Das DOS speichert unter der Kanalnummer die im "OPEN" genannte Puffernummer (siehe 1.1.0.3.). Im Rechner wiederum ist die Kanalnummer unter der im "OPEN" genannten Filenummer gespeichert. Wenn Sie auf diesen Puffer zugreifen, so ruft der Rechner die unter der im Befehl angegebenen Filenummer gespeicherte Kanalnummer ab und gibt diese an die Floppy weiter. Die Floppy verwendet nun den unter dieser Kanalnummer abgespeicherten Puffer.

7. Für ganz Eilige und Intelligente: Der Direktzugriff

7.0. Was können Sie damit machen?

7.0.0. Anwendungsgebiete im Vergleich zur SEQ-Datei

Beim Direktzugriff kann ein Datensatz (bis zu 255 Bytes) direkt auf einen bestimmten Block der Diskette geschrieben und von dort wieder gelesen werden. Im Gegensatz zur SEQ-Datei brauchen beim Direktzugriff, wenn ein bestimmter Datensatz gelesen werden soll, nicht alle in der Datei vor diesem Satz stehende Daten gelesen zu werden. Man erreicht also durch den Direktzugriff eine wesentliche Verkürzung der Zugriffszeit.

Allerdings muß der Programmierer eine eigene Organisation seiner Datei aufbauen, was ein intelligentes Programm erfordert.

Anwendungsmöglichkeiten für den Direktzugriff liegen bei der Speicherung und Verwaltung von Adressen, bei der Lagerbestandsverwaltung und ähnlichen Problemen, bei denen einzelne Datensätze von beschränkter Länge ohne Rücksicht auf vorstehende Daten direkt abgerufen und geändert werden müssen.

7.0.1. Die Spur-Sektor-Einteilung

Der Programmierer muß beachten, daß die Blöcke auf der Diskette durch Spur- und Sektornummer gekennzeichnet sind. Die Blockanzahl pro Spur nimmt von außen nach innen ab und es ergibt sich folgende Verteilung:

Spurnummer	Sektornummer	Fortlaufende Numerierung	Fortlaufende Numerierung bei Auslassung von Spur 18
1–17	0–20	0–356	0–356
18–24	0–19	357–496	357–476
25–30	0–17	497–604	477–584
31–35	0–16	605–689	585–669

Eine einfache Umrechnung von durchlaufender Numerierung in Spur-Sektor-Numerierung für 357 Blocks pro Diskette ist durch:

$$SP = \text{INT}(N/21) + 1$$

$$SE = N - (SP - 1) * 21$$

möglich. Dabei werden alle Sektoren der Spuren 1–17 verwendet.

Eine Umrechnung, die alle Blöcke der Diskette (natürlich bis auf Spur 18, die das Inhaltsverzeichnis enthält), also 670 ausnutzt, ist im Beispielprogramm "Direktzugriff" (siehe unten).

ACHTUNG: Spur 18 darf nicht im Direktzugriff verwendet werden, da sonst das Inhaltsverzeichnis und die BAM (Verzeichnis der freien/belegten Blöcke) überschrieben werden. Spätestens beim Schließen eines Datenkanals leuchtet dann die Fehlerlampe auf. Aber auch schon vorher können Fehler auftreten, die nicht bemerkt werden und unter Umständen dazu führen, daß bestimmte Blöcke oder auch die ganze Diskette nicht mehr kontrolliert gelesen oder beschrieben werden können.

7.0.2. Beispiel für eine einfache Datei

Es soll eine Adressendatei angelegt werden. Der Stammdatensatz einer Adresse wird jeweils auf einem Block gespeichert.

Jeder Adresse wird eine Zahl zugeordnet. Diese Zahl gibt an, im wievielten Block (in durchlaufender Nummerierung) die Daten zu der entsprechenden Adresse stehen.

Sollen die Daten einer Adresse aufgerufen werden, so braucht nur die fortlaufende Nummer eingegeben zu werden. Der Rechner kann daraus (siehe Beispiel Programm "Direktzugriff") Spur und Sektor berechnen. Durch Direktzugriff kann dieser Block dann in den Rechner gebracht werden, und die Daten können angezeigt, ausgedruckt oder auch geändert werden. Die geänderten Daten können dann auf Diskette geschrieben werden.

7.0.3. Der Zugriff mit Verweisdatei

Die oben aufgeführte Datei hat noch einen Nachteil. Sollen zum Beispiel alle Adressen einer bestimmten Stadt ausgedruckt werden, so muß jeder Block gelesen werden und es muß überprüft werden, ob die Postleitzahl der Adresse mit der der gesuchten Postleitzahl übereinstimmt. Das Durchsuchen der ganzen Datei kann unter Umständen sehr lange dauern.

Abkürzen läßt sich die Suchzeit durch das Anlegen von »Verweisdateien«. In unserem Beispiel könnte man zusätzlich zur Hauptdatei eine Datei (evtl. sogar eine SEQ-Datei) anlegen, in der nur die laufende Nummer und die Postleitzahl jeder Adresse gespeichert sind. Jetzt muß nur diese Datei (die natürlich viel kürzer ist als die Hauptdatei) durchsucht werden. Wird festgestellt, daß die Postleitzahl mit der gesuchten übereinstimmt, so wird dann auf den Block mit dieser laufenden Nummer zugegriffen.

Es sind auch ineinander verschaltete Verweisdaten möglich, das heißt, eine Verweisdatei zeigt nicht auf die Hauptdatei, sondern auf die nächste Verweisdatei, die dann auf die Hauptdatei zugreift. Natürlich kann man auch mehr als zwei verschachtelte Verweisdateien verwenden. Hier ist jedoch in den meisten Fällen die Grenze des Sinnvollen erreicht.

Dieser Zugriff mit Verweisdatei verkürzt die Suchzeit bei geschickter Wahl der Verweisdateien erheblich. Allerdings erfordert er ein kompliziertes Programm.

Außerdem muß sich der Programmierer schon bei der Erstellung des Programms im Klaren sein, welche Daten einmal als Suchkriterien in Frage kommen, für welche Kriterien er also Verweisdaten anlegen muß.

ANMERKUNG: Ist die Verweisdatei kurz (5000–10000 Byte), so ist es oft günstiger, sie im Speicher des Rechners anzulegen, da der Rechner wesentlich schneller darauf zurückgreifen kann als auf die Floppy.

7.1. Die Möglichkeiten des Eingriffs

Der Programmierer hat folgende Einflußmöglichkeiten:

1. Beeinflussung der Verbindung Rechner – Puffer durch die Befehle:
"OPEN", "CLOSE", "PRINT#", "INPUT#" und "GET#" (siehe 7.5.).
2. Setzen des Pufferzeigers durch den Befehl:
"B-P" (siehe 7.6.).
3. Schreiben eines Pufferinhalts auf Diskette oder Lesen eines Diskettenblocks in einen Puffer durch die Befehle:
"B-W" bzw. "B-R" (siehe 7.7.) und "U1" bzw. "U2" (siehe 7.11.1.).
4. Belegen oder freigeben eines Blocks in der BAM (Blockverfügbarkeitsliste) durch die Befehle:
"B-A" bzw. "B-F" (siehe 7.8.).
5. Lesen oder beschreiben einer beliebigen Speicherstelle »der Floppy« durch die Befehle:
"M-W" bzw. "M-R" (siehe 7.9.).
6. Ausführen von Maschinenroutinen durch die Floppy durch die Befehle:
"B-E" bzw. "M-E" (siehe 7.10.).
7. User (siehe 7.11.).

Lediglich im Fall 1. kann der Befehl vom Rechner direkt gegeben werden. In den Fällen 2. – 7. muß das entsprechende Kommando durch:

```
PRINT#15,"KOMMANDO"
```

an das DOS (Disk operating system) = Floppy-Betriebssystem gegeben werden. Das DOS übernimmt die Ausführung des Befehls.

7.2. Das Vorgehen beim Schreibzugriff

Der Schreibzugriff (wenn also Daten auf Diskette geschrieben werden sollen) wird wie folgt ausgeführt:

1. Öffnen des Kommando-/Fehlerkanals (siehe 1.1.0.3.), sofern nicht bereits geschehen.
2. Öffnen eines Puffers (siehe 1.1.0.5.), sofern nicht bereits geschehen.
3. Schreiben der Daten in den Puffer (siehe "PRINT#" 1.1.2. und 7.5.2.).
4. Befehl an das DOS (s. o.), den Pufferinhalt auf Diskette zu schreiben (siehe "B-W" 7.7.).

Vergessen Sie bitte auf keinen Fall nach Beenden des Schreibzugriffs den geöffneten Puffer wieder zu schließen!

7.3. Der Lesezugriff

Der Lesezugriff (wenn Sie also Daten von Diskette lesen wollen) wird wie folgt ausgeführt:

1. Öffnen des Kommando-/Fehlerkanals (siehe 1.1.0.3.), sofern nicht bereits geschehen.
2. Öffnen eines Puffers (siehe 1.1.0.5.), sofern nicht bereits geschehen.
3. Lesen der Daten von Diskette in den Puffer (siehe "B-R"; siehe 7.7.).
4. Lesen der Daten in den Rechner (siehe "INPUT#" und "GET#" 1.1.3. bzw. 1.1.4. und 1.5.3. und 7.5.4.).

7.4. Vorgehen beim Ausführen von Maschinenroutinen durch die Floppy

1. Möglichkeit: Mit "B-E" (siehe 7.10.).

1. Phase:

- a.) Umwandeln der Maschinencodes mit "CHR\$" und Zusammensetzen der Codes zu einem String (die 80 Zeichen-Grenze braucht nicht beachtet zu werden).
- b.) Schreiben dieses Strings in einen Puffer (mit "PRINT#").
- c.) Setzen des Pufferzeigers (siehe 7.6.) auf 235.
- d.) Schreiben des Pufferinhalts auf Diskette (mit "B-W" siehe 7.7.). Dabei wird in Byte 0 des Diskettenblocks 234 (= NOP = no Operation = keine Operation) abgespeichert.

2. Phase:

Aufrufen der Routine durch "B-E" (siehe 7.10.)

2. Möglichkeit: (mit "M-W" und "M-E" (siehe 7.9. bzw. 7.10.).

1. Schreiben des Programms ab einer bestimmten Speicherstelle mit "M-W" (siehe 7.9.).
2. Ausführen der Routine durch "M-E" (siehe 7.10.).

3. Möglichkeit: Mit "USER" (siehe 7.11.).

1. Siehe 2. Möglichkeit.

2. Schreiben der Startadresse der Sprungtabelle in Byte 00DD (=dez.221) und 00DE (=dez. 222) in der Reihenfolge LOW Byte und HIGH Byte.

3. Schreiben der Startadresse des Programms an eine Position (zum Beispiel Byte 11 und 12) LOW-Byte vor HIGH-Byte.

4. Aufruf der Routine durch Angabe der Sprungtabelleposition, im obigen Beispiel durch "6".

Die erste Möglichkeit empfiehlt sich besonders, wenn die Routinen häufig verwendet werden, allerdings ist die Länge des Maschinenprogramms auf 255 Bytes beschränkt.

7.5. Beeinflussung der Verbindung Rechner-Puffer

7.5.0. Öffnen eines Puffers

Das Öffnen eines Puffers geschieht durch:

1. OPENLA,G,K,"#"

oder:

2. OPENLA,G,K,"#ZAHL"

LA = logische Adresse, G = Gerätenummer, K = Kanalnummer

1. ordnet den nächsten freien 2. den durch die Zahl angegebenen Puffer zu.

Im Fall 1. können Sie die Nummer des zugeordneten Puffers durch folgende Routine erfahren:

```
10 GET#LA,PN$:IFPN$=""THENPN$=CHR$(0)
20 PN=ASC(PN$):PRINT"PUFFERNUMMER ";PN
```

LA = logische Adresse, über die der Puffer geöffnet wurde.

Allerdings muß die Routine angewendet werden, bevor ein "PRINT#" oder "INPUT#" auf diesen Puffer gegeben wurde, am besten gleich nach dem "OPEN". Näheres über das Öffnen eines Puffers erfahren Sie in 1.1.0.5..

7.5.1. Das Schließen eines Puffers

... geschieht durch:

CLOSE LA

LA = logische Adresse, mit der der Puffer eröffnet wurde.

Näheres über den CLOSE-Befehl erfahren Sie in 1.1.1..

7.5.2. Das Schreiben in einen Puffer

... geschieht durch:

PRINT#LA,Variable;...;

LA = logische Adresse, mit der der Puffer eröffnet wurde.

Die nähere Erläuterung dieses Befehls finden Sie in 1.1.2.

ANMERKUNG: Soll der Puffer, in den Sie schreiben, später mit "b-w" (siehe 7.7.) auf Diskette geschrieben werden, so ist es oft zweckmäßig, nicht mehr als 253 Byte in den Puffer zu schreiben.

7.5.3. Das Lesen aus dem Puffer

... geschieht durch:

INPUT#LA, Variable,....,

LA = Logische Adresse, mit der der Puffer eröffnet wurde.

Die nähere Beschreibung dieses Befehls finden Sie in 1.1.3.

ACHTUNG! Lesen Sie niemals Daten mit "INPUT", wenn Sie nicht völlig sicher sind, daß nach spätestens 80 Zeichen ein CR folgt!

7.5.4. Das Lesen von einzelnen Zeichen: GET#

... geschieht durch:

GET#LA,String-Variable

Diesen Befehl finden Sie in 1.1.4. erklärt.

Wichtiger Hinweis: Wurde durch ein "GET#" das Byte gelesen, auf das der "Letztes-Zeichen-Zeiger" (siehe 7.6.) zeigt, so wird der Lesezeiger auf 1 gesetzt. Das heißt, ein weiteres "GET#" liest wieder das erste Byte.

7.6. Der Pufferzeiger

Für jeden Puffer gibt es im DOS (Disk Operating System = Floppy-Betriebssystem) einen Pufferzeiger (englisch: buffer-pointer = B-P). Dieser Pufferzeiger hat beim Schreiben in und beim Lesen von dem Puffer verschiedene Funktionen.

7.6.0. So sieht das "B-P"-Kommando aus

PRINT#15,"B-P";KN;PZ

KN = Kanalnummer unter der der Puffer zugeordnet wurde.

PZ = neue Pufferzeiger-Position

"B-P" setzt den Pufferzeiger auf den angegebenen Wert (unabhängig von seinem alten Wert).

7.6.1. Funktion des Pufferzeigers beim Schreiben

Nach einem PRINT#-Befehl zeigt der Pufferzeiger auf das Byte, das dem letzten Geschriebenen im Puffer folgt. Weitere Daten werden ab der Position des Pufferzeigers gespeichert. So wird gewährleistet, daß Daten, die hintereinander in den Puffer geschrieben werden, auch hintereinander im Puffer abgespeichert werden.

Der Anwender kann vor einem PRINT#-Befehl den Pufferzeiger setzen, um Daten ab einer bestimmten Position im Puffer zu speichern. Dies kann von Bedeutung sein, wenn mehrere kurze Datensätze an definierten Stellen des Puffers (und damit auch später des Diskettenblocks) gespeichert werden sollen.

Wird das Kommando "B-W" (also Pufferinhalt auf Diskette schreiben) gegeben, so wird der Pufferzeiger um 1 erniedrigt (er zeigt also jetzt auf das zuletzt geschriebene Byte). Dieser erniedrigte Pufferzeiger wird in Position 0 des Diskettenblocks geschrieben. Er zeigt auf die Position des zuletzt geschriebenen (und somit des letzten gültigen) Bytes im Diskettenblock. Dieser Wert wirkt als »Letztes-Zeichen-Zeiger« (englisch: last character pointer). Nach Ausführung des Schreibbefehls wird der Pufferzeiger auf 1 gesetzt.

7.6.2. Funktion des Pufferzeigers beim Lesen

Beim Lesen gibt der Pufferzeiger an, ab welcher Position Daten mit "INPUT#" oder "GET#" gelesen werden. Durch das Kommando "B-R" (Block von Diskette in den Puffer lesen) wird der »Letztes-Zeichen-Zeiger« des Puffers, in den der Block geschrieben werden soll, auf den Wert gesetzt, der in Byte 0 des gelesenen Blocks steht.

Werden Daten gelesen, wird der Pufferzeiger nach jedem gelesenen Byte um 1 erhöht. Er zeigt also stets auf das nächste, zu lesende Byte. Daher wird er hier auch als »Lesezeiger« bezeichnet.

Stimmt der Wert des »Lesezeigers« mit dem des »Letzten-Zeichen-Zeigers« überein, so wird mit dem entsprechenden Byte EOI = L (End of identification, Ende der Identifikation) übertragen. Dadurch wird die Datenübertragung beendet. Der Pufferzeiger wird wieder auf 1 gesetzt. Wenn also Daten mit "GET#" gelesen werden, wird, nachdem das letzte gültige Zeichen gelesen worden ist, wieder ab Byte 1 gelesen.

Durch das Setzen des Pufferzeigers vor "GET#" und "INPUT#" hat der Programmierer die Möglichkeit, bestimmte Stellen des Puffers auszulesen.

ACHTUNG: Wird der Pufferzeiger vor einem "INPUT#" hinter den »Letztes-Zeichen-Zeiger« gesetzt und folgt kein Begrenzungszeichen mehr, so gerät die Floppy in einen völlig unkontrollierbaren Zustand. Es kann sogar passieren, daß auch der Rechner nicht mehr kontrolliert ansprechbar ist.

7.6.3. Anwendungsmöglichkeiten für "B-P"

Der Block-Puffer-Befehl ist sehr nützlich, wenn man kurze Datensätze hat, und deshalb mehrere Datensätze in einem Block speichern will.

Beispiel:

In einem einfachen Lagerbuchaltungs-Programm sollen nur Preis und Lagerbestand gespeichert werden. Setzt man für den Preis 10 Byte und für ein Lagerbestand 8 Byte an, und berücksichtigt, daß hinter jeder gespeicherten Variablen ein "CR" = CHR\$(13) stehen muß, damit es beim Lesen mit "INPUT#" keine Probleme gibt, ergibt sich folgende Abspeicherungsmöglichkeit:

Byte

1–10	Preis des 1. Artikels
11	CR
12–19	Lagerbestand des 1. Artikels
20	CR
21–30	Preis des 2. Artikels
31	CR
32–39	...
40	...
41–50	...
...	

Numeriert man die Artikel von 1–12 durch, ergeben sich folgende Formeln für die Position, an der die Abspeicherung beginnt:

Preis: Position = (Artikel-Nr. – 1)*20 + 1

Lagerbestand: Position = (Artikel-Nr. – 1) *20 + 12

B-P Demonstrationsprogramm

```
0 REM*** B-P DEMO
10 OPEN15,8,15,"I0":REM*** Öffnen des Kommando-/Fehlerkanals und
11 REM*** Initialisieren der Diskette in Laufwerk 0
20 OPEN10,8,10,"#":REM*** Öffnen eines Datenkanals
25 :
26 :
27 :
28 :
29 REM*** Abfrage ob lesen oder speichern von Daten erwünscht
30 INPUT "Wollen Sie Daten speichern";A$
40 IFA$="JA"ORA$="ja"THEN100
50 INPUT "Wollen Sie Daten lesen";A$
60 IFA$="JA"ORA$="ja"THEN700
70 IFA$<>"JA"ANDA$<>"ja"THEN30
95 :
96 :
97 :
98 :
99 REM*** Eingabe von Preis und Lagerbestand
100 INPUT"Artikelnummer";AN
105 AN=INT(AN):IFAN<10RAN>12THEN100
110 INPUT "Preis";P$
115 IFLEN(P$)>10THEN110
```

```

120 INPUT "Lagerbestand";LB$ : IFLEN(LB$)>8THEN120
197 :
198 :
199 REM*** Berechnung der Position, ab der der Preis gespeichert wird
200 PZ=(AN-1)*20+1
209 REM*** Setzen des Pufferzeigers von Kanal 10 auf PZ
210 PRINT#15,"B-P";10;PZ
296 :
297 :
298 REM*** Schreiben von P$ in den Puffer von Kanal 10 und zwar ab Position des
299 REM Pufferzeigers
300 PRINT#10,P$;CHR$(13);:REM*** Das CHR$(13) ist nötig, damit INPUT# die Variablen
301 REM trennen kann.
397 :
398 :
399 REM*** Berechnung der Position, ab der der Lagerbestand gespeichert wird.
400 PZ= (AN-1)*20+12
409 REM*** Setzen des Pufferzeigers von Kanal 10 auf PZ
410 PRINT#15,"B-P";10;PZ
496 :
497 :
498 REM*** Schreiben von L$ in den Puffer von Kanal 10 und zwar ab Position des
499 REM*** Pufferzeigers
500 PRINT#10,LB$;CHR$(13);:REM*** Das CHR$(13) ist nötig, damit INPUT#die Variablen
501 REM trennen kann.
595 :
596 :
597 :
598 REM*** Schreiben des Puffers von Kanal 10 auf Spur 1, Sektor 19 der Diskette
599 REM In Laufwerk 0
600 PRINT#15,"B-W";10;0;1;19
610 GOTO30:REM*** Rücksprung in die Abfrage
695 :
696 :
697 :
698 :
699 REM*** Auslesen von Daten
700 INPUT"Artikelnummer";AN
705 AN=INT(AN):IFAN<10RAN>12THEN700
796 :
797 :
798 REM*** Lesen von Spur 1, Sektor 19 der Diskette in Laufwerk 0 in den Puffer
799 REM*** von Kanal 10
800 PRINT#15,"B-R";10;0;1;19
897 :
898 :
899 REM*** Berechnung der Position, ab der der Preis gespeichert ist.
900 PZ=(AN-1)*20+1
909 REM*** Setzen des Pufferzeigers von Kanal 10 auf PZ
910 PRINT#15,"B-P";10;PZ
996 :
997 :
998 REM*** Lesen von P$ in den Puffer von Kanal 10 und zwar ab Position des
999 REM*** Pufferzeigers
1000 INPUT#10,P$
1097 :
1098 :
1099 REM*** Berechnung der Position, ab der der Lagerbestand gespeichert ist.
1100 PZ=(AN-1)*20+12
1109 REM*** Setzen des Pufferzeigers von Kanal 10 auf PZ
1110 PRINT#15,"B-P";10;PZ
1196 :
1197 :

```

```

1198 REM*** Lesen von LB$ in den Puffer von Kanal 10 und zwar ab Position des
1199 REM*** Pufferzeigers.
1200 INPUT#10,LB$
1300 PRINT:PRINT"Preis: ";P$:PRINT:PRINT"Lagerbestand : ";LB$
1320 GOTO30:REM*** Rücksprung in die Abfrage.
READY.

```

7.7. Der Eingriff Puffer-Diskette

Wichtige Hinweise:

Werden beide Disketten im Direktzugriff benutzt, so muß für jedes Laufwerk ein eigener Kanal und ein eigener Puffer verwendet werden. Sonst kann es unter Umständen passieren, daß die Floppy die beiden Disketten verwechselt.

Bei der Anwendung des B-W-Befehls ist darauf zu achten, daß nicht auf einen Block geschrieben wird, der für eine PRG- oder SEQ-Datei benutzt wird. Durch einen falschen "B-W" kann ein ganzes Programm oder eine ganze SEQ-Datei zerstört werden!

7.7.0. Schreiben eines Pufferinhaltsverzeichnisses auf Diskette

Der Befehl an das DOS, einen bestimmten Pufferinhalt auf Diskette zu schreiben, sieht so aus:

```
PRINT#15,"B-W";KN;LW;SP;SE
```

KN = Kanalnummer,
mit der Puffer
zugeordnet wurde

LW = Laufwerks-
nummer

SP = Spurnummer

SE = Sektor-
nummer

Der gesamte Pufferinhalt (Byte 1 – 255) wird auf den angegebenen Diskettenblock geschrieben. In Byte 0 des Diskettenblocks wird der »Letztes-Zeichen-Zeiger« (siehe 7.6.) gespeichert.

Beispiel:

```
PRINT#15,"B-W";10;1;10;7
```

Schreibt den Inhalt des Puffers, der Kanal 10 zugeordnet worden ist, auf Spur 19, Sektor 7 der Diskette, die sich im Laufwerk 1 befindet.

ACHTUNG: Vor dem "B-W"-Befehl sollte der Pufferzeiger nicht höher als 254 sein (es sollten also nicht mehr als 253 Bytes in den Puffer geschrieben werden), da es sonst beim Lesen des Blocks (siehe 7.7.1.) Schwierigkeiten geben kann!

7.7.1. Lesen eines Diskettenblocks in einen Puffer

Der Befehl an das DOS einen Diskettenblock in einen Puffer zu lesen sieht so aus:

```
PRINT#15,"B-R";KN;LW;SP;SE
```

KN = Kanalnummer unter der der Puffer zugeordnet wurde.

LF,SP,SE: Laufwerksnummer, Spur und Sektor des Diskettenblocks, der gelesen werden soll.

Beispiel:

```
PRINT#15,"B-R";80;9;1
```

Spur 9, Sektor 1 der Diskette, die sich in Laufwerk 0 befindet wird in den Puffer, der Kanal 8 zugeordnet wurde, gelesen.

ACHTUNG: Wurden auf den zu lesenden Block mehr als 252 Zeichen (einschließlich der CR) geschrieben (oder ist der »Letztes-Zeichen-Zeiger« größer als 255), so kann es beim ersten Lesen nach dem Schreiben vorkommen, daß nur das erste Zeichen des Blocks übertragen wird. Normalerweise werden beim zweiten Lesen alle Daten übertragen. Ist die Datensatzlänge auf allen Blocks gleich, so kann sicherheitshalber folgende Routine verwendet werden:

```

900 ...
950 ...
1000 PRINT#10,A$,CHR$(13);B$,CHR$(13);...
1010 L=LEN(A$+B$+...)
1020 PRINT#15,"B-W";10;LF;SP;SE
1030 ...
1040 ...
1900 ...
1950 ...
2000 PRINT#15,"B-R";10;LF;SP;SE
2005 PRINT#15,"B-P";10;1
2010 INPUT#10,A$,B$,...
2020 IFL<>LEN(A$+B$+...)THEN2000

```

Natürlich müssen im Programm vorher die nötigen OPEN-Befehle gegeben worden sein.

7.8. Direktzugriff in die BAM

Der Direktzugriff in die BAM erfolgt durch die Kommandos "B-A" und "B-F". Die entsprechenden Befehle sehen so aus:

```
PRINT#15,"B-A";LW;SP;SE  
bzw.  
PRINT#15,"B-F";LW;SP;SE
```

SP und SE sind Spur- und Blocknummer des Blocks, der belegt (bzw. freigegeben) werden soll.

LW gibt das Laufwerk an, in dem sich die Diskette befindet, auf der ein Block belegt (bzw. freigegeben) werden soll.

Mit "B-A" und "B-F" kann ein bestimmter Block in der BAM (Verzeichnis der freien/belegten Blöcke siehe 2.2.) als belegt bzw. als frei gekennzeichnet werden. Wird ein Block als belegt gekennzeichnet (bei Blöcken, die von PRG- oder SEQ-Dateien benutzt werden, geschieht dies automatisch), so benutzt das DOS ihn nicht mehr zum Abspeichern von neuen PRG- oder SEQ-Dateien. Ein Direktzugriff (auch Schreibzugriff) auf einen belegten Block ist weiterhin möglich.

Die durch "B-A" und "B-F" geänderte BAM wird erst auf Diskette geschrieben, wenn ein Direktzugriffskanal oder eine Schreibdatei geschlossen werden. Ein weiterer Grund, warum auf keinen Fall ein ordnungsgemäßes "CLOSE" vergessen werden darf.

Wird versucht, einen Block zu belegen, der schon belegt ist (durch voriges "B-A" oder durch PRG- oder SEQ-Datei), so leuchtet an der Floppy die Fehlerlampe auf und Abfragen der Fehlermeldung (siehe 8.) ergibt:

```
65 NO BLOCK SP SE
```

SP und SE geben den nächsten freien Block (in aufsteigender Numerierung von SP und SE) an. Ist hinter dem verlangten Block kein Block mehr frei, so werden SP und SE als 00 angegeben.

Beispiele:

```
PRINT#15,"B-A";1;12;16
```

Der Block auf Spur 12, Sektor 16 oder Diskette in Laufwerk 1 wird als belegt gekennzeichnet.

```
PRINT#15,"B-F";0;24;7
```

Der Block auf Spur 24, Sektor 7 der Diskette in Laufwerk 0 wird als frei gekennzeichnet.

WICHTIGE HINWEISE:

Wird eine Diskette im Direktzugriff und für PRG- oder SEQ-Dateien gleichzeitig verwendet, so ist das B-F-Kommando mit äußerster Vorsicht anzuwenden. Wird ein Block, der Teil einer PRG- oder SEQ-Datei ist, freigegeben, so kann es vorkommen, daß dieser Block vom DOS beim Schreiben neuer Dateien überschrieben wird. Dadurch kann die ganze Datei wertlos werden!

7.9. Eingriff in den Floppyspeicher

Mit "M-W" und "M-R" können Daten in den Floppyspeicher geschrieben oder aus ihm gelesen werden. Bei diesen beiden Befehlen muß die entsprechende Adresse in zwei Bytes zerlegt werden: Hohes Byte und Niedriges Byte.

Beispiel: Die Adresse dezimal 4437 soll zerlegt werden.

$$4437 = 17 * 256 + 85$$

Hohes Byte: 17, Niedriges Byte: 85

7.9.0. M-W

Der M-W-Befehl sieht so aus:

```
PRINT#15,"M-W"CHR$(NB)CHR$(HB)CHR$(A)CHR$(....)CHR$(....)
```

HB = Hohes Byte, NB = Niedriges Byte der Adresse, ab der in den Floppyspeicher geschrieben werden soll.

A = Anzahl der zu schreibenden Bytes (max. 34).

Zu beachten ist, daß sowohl nach "M-W" als auch nach den einzelnen CHR\$'s kein Trennungszeichen stehen darf!

Das Kommando kann auch so übermittelt werden:

```
10 K$= "M-W"  
20 D$=CHR$(NB)+CHR$(HB)+CHR$(A)+CHR$(....)+CHR$(....)  
30 PRINT#15,K$D$
```

Beispiel:

```
PRINT#15,"M-W"CHR$(1)CHR$(17)CHR$(3)CHR$(5)CHR$(77)CHR$(76)
```

Ab Adresse $17 * 256 + 1$ werden die 3 Bytes 5, 77 und 76 in den Floppyspeicher geschrieben.

7.9.1. M-R

Der M-R-Befehl sieht so aus:

```
PRINT#15,"M-R"CHR$(NB)CHR$(HB)
```

HB = Hohes Byte, NB = Niedriges Byte der Adresse, die gelesen werden soll.

Auch hier ist zu beachten, daß hinter "M-R" und zwischen den CHR\$'s kein Trennzeichen steht.

"M-R" speichert das adressierte Byte im Puffer des Kommando-/Fehlerkanals. Durch ein GET# (INPUT# geht NICHT!) auf die logische Adresse des Kommando-/Fehlerkanals kann das Byte in den Rechner gebracht werden und mit ASC kann sein Wert bestimmt werden.

Beispiel: Es soll der Inhalt der Speicherzelle 4437 (= $17 \cdot 256 + 85$) gelesen werden:

```
10 OPEN 15,8,15
20 PRINT#15,"M-R"CHR$(85)CHR$(17)
30 GET#15,A$:IFA$=""THEN A$=CHR$(0)
40 PRINT"Wert:";ASC(A$)
```

7.10. Ausführung von Maschinenroutinen durch die Floppy

ACHTUNG: Jede Maschinenroutine muß mit dem Befehl RTS (return from subroutine, Code = 96 = \$60) abschließen, damit nach Ausführung der Routine die Kontrolle an das DOS zurückgegeben wird.

Das allgemeine Vorgehen beim Ausführen von Maschinenroutinen ist in 7.4. beschrieben. Hier sind nur die Befehle B-E und M-E erklärt.

7.10.0 B-E

Der B-E-Befehl sieht so aus:

```
PRINT#15,"B-E";KN;LW;SP;SE
```

KN = Kanalnummer des Kanals, dem der Puffer zugeordnet wurde, in den der angegebene Block gelesen wird.

LW, SP und SE: Laufwerksnummer, Spur und Sektor des auszuführenden Blocks.

"B-E" liest den angegebenen Block in den Puffer des angegebenen Kanals und führt den Block ab Byte 0 als Maschinenprogramm aus.

In Byte 0 steht normalerweise der »Letztes-Zeichen-Zeiger«. Der Programmierer muß dafür sorgen, daß dieser Zeiger auf 234 (= NOP = keine Operation) steht. Wie dies geschehen kann, ist in 7.4. erklärt.

Beispiel:

```
PRINT#15,"B-E",7,1,31,5
```

Der Block auf Spur 31, Sektor 5 der Diskette in Laufwerk 1, wird in den Puffer von Kanal 7 gelesen und ab Byte 0 ausgeführt.

Das B-E-Kommando empfiehlt sich für kurze (bis zu 255 Bytes) Maschinenroutinen, die häufig verwendet werden. Diese Routinen brauchen nur einmal auf Diskette geschrieben zu werden und können dann jederzeit abgerufen werden.

7.10.1. M-E

Der M-E-Befehl sieht so aus:

```
PRINT#15,"M-E"CHR$(LB)CHR$(HRB)
```

LB und HB sind niedriges bzw. hohes Byte der Startadresse des Maschinenprogramms.

"M-E" bewirkt, daß das DOS einen Speicherbereich, beginnend mit der angegebenen Adresse, als Maschinenprogramm ausführt.

Beispiel:

```
PRINT#15,"M-E"CHR$(85)CHR$(17)
```

Ab Adresse 4437 = $17 \cdot 256 + 85$ führt die Floppy den Speicherbereich als Maschinenroutine aus.

7.11. USER

7.11.0. Allgemein

User ermöglicht das Aufrufen von Maschinenroutinen in der Floppy über eine Sprungtabelle. In ihr können bis zu 15 Adressen gespeichert (Reihenfolge LOW Byte, HIGH Byte) und abgerufen werden.

Falls Sie mehrere Sprungtabellen brauchen, haben Sie außerdem die Möglichkeit, den USER-Zeiger, der auf die Anfangsadresse der Tabelle zeigt, zu ändern. Er steht an den Stellen \$00DD (= dez. 221) und \$00DE (= dez. 222) in der Reihenfolge LOW Byte – HIGH Byte.

Beispiel: Sie haben ab Adresse \$1201 (= dez. 4609) im Floppyspeicher eine Sprungtabelle mit den Adressen \$1300, \$1303, \$FF26 und \$FF65: Der Floppyspeicher sieht dann so aus:

Speicherstelle	Wert	(=dez.)		
\$00DD	1	01	niedriges Byte	
\$00DE	16	12	hohes Byte	der Startadresse der Sprungtabelle
...				
...				
\$1201	0	00	niedriges Byte	
\$1202	19	13	hohes Byte	der 1. Sprungadresse
\$1203	3	03	niedriges Byte	
\$1204	19	13	hohes Byte	der 2. Sprungadresse
\$1205	38	26	niedriges Byte	
\$1206	255	FF	hohes Byte	der 3. Sprungadresse
\$1207	101	65	niedriges Byte	
\$1208	255	FF	hohes Byte	der 4. Sprungadresse

Soll jetzt die Maschinenroutine, die bei \$ 1303 beginnt, aufgerufen werden, so geschieht dies, weil es sich um die 2. Adresse der Tabelle handelt, durch:

```
PRINT#15,"U2"
```

Die Adressen der Sprungtabelle sind von 1 bis 9 oder aber von a – o durchnummeriert. Sie können damit oben auch schreiben:

```
PRINT#15,"UB"
```

Dabei ergibt sich kein Unterschied, außer daß Sie bei den Buchstaben 15 Adressen ansprechen können. "U1" und "UA" rufen somit dieselbe Adresse, nämlich die erste der Sprungtabelle, auf.

ACHTUNG: Die Maschinenroutinen, die mit USER aufgerufen werden, müssen mit RTS (= return from subroutine) abschließen, damit nach Beendigung der Routine die Kontrolle wieder an das DOS zurückgegeben wird.

7.11.1. Die Standardsprungtabelle

Nach dem POWER-ON-RESET-Zyklus (ausgelöst durch Einschalten der Floppy oder des Rechners) zeigt der USER-Zeiger auf eine Standardsprungtabelle. Falls Sie ihn versetzt haben, können Sie ihn jederzeit durch "U0" wieder auf die Standardsprungtabelle zurücksetzen lassen.

In der Standardsprungtabelle stehen die Adressen einiger DOS-Routinen. Besondere Bedeutung haben die Routinen, die mit "U1" und "U2" aufgerufen werden können.

```
PRINT#15,"U1";KN;LW;SP;SE
```

(KN = Kanalnummer, LW = Laufwerksnummer, SP und SE = Spur- und Sektornummer des Blocks.) bewirkt die Ausführung eines modifizierten "B-R"-Kommandos: der Block wird in den Puffer des angegebenen Kanals gelesen; der »Letztes-Zeichen-Zeiger« wird auf 255 (= \$FF) und der Pufferzeiger auf 0 gesetzt (siehe 7.6.). Es kann jetzt ab Byte 0 aus dem Puffer gelesen werden.

```
PRINT#15,"U2";KN;LW;SP;SE
```

(KN, LW, SP, SE siehe oben)

bewirkt ein modifiziertes "B-W". Es wird der GESAMTE Pufferinhalt (also ab Byte 0) auf Diskette geschrieben. In Byte 0 des Diskettenblocks wird also nicht der »Letztes-Zeichen-Zeiger«, sondern der Wert, der in Byte 0 des Puffers steht, gespeichert.

U3 – U5 bewirken Sprünge zu den undefinierten Routinen. U3 bewirkt einen Sprung nach \$1300, U4 nach \$1303 und U5 nach 1306.

U6 – U9 führen zu Sprüngen in nicht leere ROM-Bereiche. Sie dürfen nicht angewendet werden, solange die entsprechenden ROM's nicht vorhanden sind.

UJ – UK dürfen bei der Standardsprungtabelle nicht verwendet werden, da sie hier nicht definiert sind.

7.12. Die Floppyadressen der Puffer

Die Puffer sind jeweils auf einer Speicherseite angeordnet. Zum Lesen oder Beschreiben des gesamten Puffers muß das niedrige Byte von 0 bis 255 laufen. Die folgende Tabelle gibt daher nur jeweils das hohe Byte an.

Puffernummer	hohes Byte	hex.
0	17	11
1	12	12
2	19	13
3	32	20
4	33	21
5	34	22
6	35	23
7	43	30
8	49	31
9	50	32
10	61	33
11	64	40
12	65	41
13	66	42 belegt durch BAM LW 0
14	67	43 belegt durch BAM LW 1
15	30	50 belegt durch DOS

7.13. Lesen von Diskettenname und ID aus der BAM

In den Bytes 144–161 von Puffer 13 bzw. 14 steht der Name der Diskette in Laufwerk 0 bzw. 1. In Byte 162 und 163 des entsprechenden Puffers steht die ID der betreffenden Diskette.

Das Auslesen von Name und ID kann durch folgendes Programm geschehen:

```
10 OPEN 15,8,15
12 :
15 REM Eingabe des Laufwerks und Umrechnen auf die Puffernummer
17 :
20 ?"Laufwerk";
30 INPUT A$
40 IFA$="0" THEN L$=CHR$(66):GOTO 100
50 IFA$="1" THEN L$=CHR$(67):GOTO 100
60 GOTO 30
70 :
80 Einlesen des entsprechenden Pufferbereiches
90 :
100 N$=""
110 FOR I=144 TO 163
115 I$=CHR$(I)
116 :
117 REM Abrufen des I'TEN Bytes
118 :
120 PRINT#15,"M-R"L$I$
130 GET#15,G$:IF G$="" THEN G$=CHR$(0)
140 N$=N$+G$
150 NEXT I
160 :
170 REM Ausdruck von Name und ID
180 :
200 PRINT: Print "Name und ID und Laufwerk";A$;":":N$
```

8.0 Was tun, wenn die Floppy spinnt?

Beim Betrieb der Floppy können, wie auch beim Rechner, Fehler auftreten. Die Floppy unterbricht dann NICHT das laufende Programm, sondern sie meldet sich nur über die Fehlerlampe und zeigt so dem Benutzer, daß ein Fehler aufgetreten ist.

Die Fehlermeldung kann vom Rechner aus abgerufen werden. Das geschieht über den Kommando-/ Fehlerkanal.

Die Abfrage kann folgendermaßen geschehen:

```
10 OPEN 15,8,15
20 INPUT#15,A,A$,A1,A2
30 PRINT A$;A1;A2
```

Auf dem Bildschirm erscheint nun die Nummer der Fehlermeldung (siehe Tabelle), danach der Text der Fehlermeldung und dann zwei Zahlen, die angeben, in welcher Spur und in welchem Sektor der Fehler aufgetreten ist (falls ein Fehler dort auftritt). Sind Dateien gelöscht worden, gibt die erste Zahl nach dem Fehlertext die Anzahl der gelöschten Blöcke an. Bei Fehlern, die nicht direkt mit der Diskette zu tun haben, sind diese beiden Zahlen = 0.

Das OPEN im Beispiel ist nicht mehr nötig, wenn der Kommando-/Fehlerkanal bereits geöffnet war. Die Fehlermeldung kann dann sofort abgefragt werden. Die Abfrage muß aber in einem kleinen Programm geschehen.

Um das Weiterarbeiten von Programmen nach einer Fehlermeldung der Floppy zu verhindern, sollte man nach jedem oder spätestens vor dem nächsten Zugriff auf die Floppy den Kommando-/Fehlerkanal abfragen.

Soll ein eventuell auftretender Fehler nicht unbedingt ausgedruckt werden, so genügt es, den ersten Teil der Fehlermeldung, den Fehlercode, abzufragen und zu testen, ob sein Wert = 0 ist oder nicht. Ist er = 0, so liegt kein Fehler vor, und das Programm kann weiterarbeiten. Ist der Fehlercode nicht = 0, so ist es nötig, daß das Programm auf den Fehler reagiert.

Beispiel:

```

10 OPEN 15,8,15
20 INPUT#15,A
30 IF A<>0THEN 1000
40 REM LAUFENDES PROGRAMM
50 ...
.. ...
999 END
1000 REM VERARBEITUNG DES FEHLERS
.... ...

```

WICHTIG: Die Verarbeitung von Daten wird bei einer Fehlermeldung der Floppy NICHT unterbrochen! Es ist also unbedingt nötig, in einem Programm, das auf die Floppy zugreift, vor oder nach jedem Kommando an die Floppy den Kommando-/Fehlerkanal zu lesen und zu testen, ob ein Fehler vorliegt!

Es folgt nun eine Tabelle der auftretenden Fehlermeldungen. Diese sind im nachfolgenden Text soweit wie möglich erklärt. Diese Erläuterungen können hier nicht jede mögliche Erscheinungsform von Fehlern berücksichtigen, Sie werden aber alle Fehler, die beim normalen Betrieb auftreten, erläutert finden.

8.1. Tabelle der Fehlermeldungen

00	OK	00	00	Es liegt kein Fehler vor
01	FILES SCRATCHED	A	00	A = Anzahl der gelöschten Dateien
20	READ ERROR	SP	SE	Block Header fehlt
21	READ ERROR	SP	SE	Synchronisationszeichen fehlt
22	READ ERROR	SP	SE	Datenblock nicht gefunden
23	READ ERROR	SP	SE	Prüfsummenfehler im Datenblock
24	READ ERROR	SP	SE	Byte wurde falsch dekoriert
25	WRITE ERROR	SP	SE	Schreib-/Prüffehler
26	WRITE PROTECT ON	SP	SE	Schreibschutz der Diskette geklebt
27	READ ERROR	SP	SE	Prüfsummenfehler im Header
28	WRITE ERROR	SP	SE	Datenblock zu lang
29	DISK ID MISMATCH	SP	SE	Diskettenidentität stimmt nicht
30	SYNTAX ERROR	00	00	Allgemeine Syntax falsch
31	SYNTAX ERROR	00	00	Ungültiges Kommando
32	SYNTAX ERROR	00	00	Kommandostring zu lang
33	SYNTAX ERROR	00	00	Dateiname falsch
34	SYNTAX ERROR	00	00	Dateiname fehlt
60	WRITE FILE OPEN	00	00	Datei schon zum Schreiben offen
61	FILE NOT OPEN	00	00	Datei noch nicht geöffnet
62	FILE NOT FOUND	00	00	Datei nicht gefunden
63	FILE EXISTS	00	00	Datei existiert bereits
64	FILE TYPE MISMATCH	SP	SE	Falscher Dateityp
65	NO BLOCK	SP	SE	Kein Block mehr frei
70	NO CHANNEL ERROR	00	00	Kanal zu oder Puffer belegt
71	DIR ERROR	SP	SE	Fehler im Inhaltsverzeichnis
72	DISK FULL	00	00	Diskette voll belegt

8.2. Ursachen, Erklärung und Abhelfen

00 OK 00 00:

Alle Operationen sind ordnungsgemäß durchgeführt worden. Es liegt kein Fehler vor.

01 FILES SCRATCHED A 00:

Diese Meldung tritt auf, wenn Dateien gelöscht worden sind, also nach dem S-Kommando. Es handelt sich also nicht um einen Fehler! A gibt an, wieviele Dateien gelöscht worden sind.

Die Lesefehler:

Die LOAD ERROR-Meldungen können mehrere Ursachen haben:

- a. Es befindet sich keine Diskette im Laufwerk, oder die Laufwerksklappe ist geöffnet.
- b. Sie verwenden eine Diskette, die noch nicht formatiert worden ist.
- c. Die Diskette, die Sie beschreiben, ist defekt.
- d. Die Floppy selber ist defekt.

Zu c. In diesem Fall kann man versuchen, mehrmals den gleichen Befehl ausführen zu lassen. Eventuell kann man die Diskette herausnehmen, neu zentrieren und dann neu einlegen.

Auf jeden Fall aber sollte die defekte Diskette Datei für Datei kopiert und dann aus dem Verkehr gezogen werden!

Zu d. Tritt ein Lesefehler in einem Laufwerk auffallend oft auf, so kann es sein, daß Ihre Floppy defekt ist.

Wechseln Sie dann bitte das Laufwerk und prüfen Sie, ob der Fehler auch hier auftritt. Ist das nicht der Fall, so sollten Sie sich an Ihren Commodore-Händler wenden und das fehlerhafte Laufwerk nicht mehr verwenden!

20 READ ERROR SP SE:

Der Block-Header fehlt.

Der Block-Header entspricht dem Dateikopf (siehe Rechner-Bedienungsanleitung). Um den Fehler zu beseitigen, formatieren Sie die Diskette am besten mit »N« (löschen und formatieren! Siehe 3.6.).

21 READ ERROR SP SE:

Synchronisationszeichen fehlt.

Das Synchronisationszeichen ist nötig, damit die Floppy den Beginn eines Blocks einwandfrei erkennen kann. Fehlt dieses Zeichen, so ist dieser Block für die Floppy nicht mehr auffindbar. Sie meldet sich mit der hier genannten Fehlermeldung.

22 READ ERROR SP SE:

Datenblock nicht vorhanden.

Der Block, auf den sich diese Fehlermeldung bezieht, ist noch nicht mit dem Befehl »N« formatiert worden.

23 READ ERROR SP SE:

Prüfsummenfehler im Datenblock.

Auf den Datenblock sind beim Schreiben zur Kontrolle zusätzlich die Summe aller im Datenblock enthaltenen Bytes geschrieben. Beim Lesen bildet die Floppy die Summe aller gelesenen Bytes und vergleicht sie mit der auf der Diskette gespeicherten Prüfsumme. Stimmen die beiden Summen nicht überein, so gibt die Floppy die obige Fehlermeldung aus.

24 READ ERROR SP SE:

Byte ist falsch dekodiert worden.

Die Fehlermeldung wird gegeben, wenn es beim Lesen zu Dekodierungsfehlern kommt.

27 READ ERROR SP SE:

Prüfsummenfehler im Header.

Die Erkennung dieses Fehlers ist im Prinzip die gleiche wie bei »23 READ ERROR«, nur daß hier die Summe aller Bytes im Header falsch war.

Die Schreibfehler:

25 WRITE ERROR SP SE:

Schreib-/Prüffehler.

Die Floppy prüft bei einem schreibenden Zugriff auf die Diskette, ob der von ihr geschriebene Block mit dem im Puffer gespeicherten übereinstimmt. Dazu liest sie den geschriebenen Block, und vergleicht

seinen Inhalt mit dem des Puffers. Tritt bei diesem Test eine Ungleichheit auf, so wird sie den Block noch einmal zu schreiben versuchen. Sollte auch das nicht erfolgreich sein, so gibt sie diesen Fehler aus. Vermutlich verwenden Sie eine schadhafte Diskette, zum Beispiel mit Kratzern in der Magnetfolie. Wechseln Sie diese Diskette bitte aus. Auch wenn dieser Fehler nicht auftritt, ist es sinnvoll, jede geschriebene Datei entweder mit "VERIFY" zu überprüfen, bzw. noch einmal zur Probe zu lesen.

26 WRITE PROTECT ON SP SE:

Schreibschutz an der Diskette.

Dieser Fehler meldet Ihnen, daß Sie versucht haben, auf eine Diskette zu schreiben, die mit dem Schreibschutz geschützt ist. Wenn Sie wirklich auf diese Diskette schreiben wollen (und nicht nur einfach die Disketten vertauscht hatten), so müssen Sie den Schreibschutz entfernen. Der Schreibschutz ist in 3.2. erläutert.

28 WRITE ERROR SP SE:

Datenblock zu lang.

Wenn Sie versucht haben, einen Datenblock zu schreiben, der zu lang ist (also mehr als 252 Zeichen enthält), so kann diese Fehlermeldung auftreten, was jedoch nicht geschehen muß! Sie sollten auf jeden Fall vermeiden, zu lange Datenblöcke zu schreiben.

Die Syntaxfehler:

Die Fehlermeldungen 30 bis 34 sind »SYNTAX ERROR«-Meldungen, das heißt, Sie haben in einem Kommando einen Fehler gemacht. Tritt ein solcher Fehler auf, so überprüfen Sie das an die Floppy gegebene Kommando. Im einzelnen unterscheidet die Floppy die folgenden Syntaxfehler:

30 SYNTAX ERROR 0 0:

Allgemeine Syntax falsch.

Sie haben einen grundsätzlichen Fehler im Aufbau des Kommandostrings gemacht. Mögliche Ursachen für eine solche Fehlermeldung können sein:

- a. Sie haben kein ":" zwischen Laufwerksnummer und Dateiname geschrieben.
- b. Fehlendes "=" zwischen Ziel- und Quelldateiname.
- c. Fehlendes "," zwischen mehreren Dateinamen bei »S« oder »N«. Näheres dazu in den Beispielen für die jeweiligen Kommandos.

31 SYNTAX ERROR 0 0:

Ungültiges Kommando.

Hier wird gemeldet, daß Sie ein nicht definiertes Kommandozeichen gegeben haben. Häufig wurde einfach vergessen, ein Kommandozeichen (zum Beispiel »S« oder »N«) anzugeben, wodurch die Laufwerksnummer als Kommandozeichen interpretiert wurde.

32 SYNTAX ERROR 0 0:

Zu langes Kommando.

Die Anzahl der Zeichen im Kommando ist auf 40 begrenzt! Wahrscheinlich sind zu viele Dateinamen angegeben worden.

33 SYNTAX ERROR 0 0:

Falscher Dateiname.

Hier wurde gegen die Regeln für die Bildung von Dateinamen verstoßen. Wie Sie Dateinamen bilden müssen, können Sie in 3.0. nachsehen.

34 SYNTAX ERROR 0 0:

Sie haben vergessen, alle nötigen Dateinamen anzugeben. Überprüfen Sie Ihr Kommando bitte darauf, ob alle nötigen Ziel- und Quelldateinamen angegeben worden sind.

Weitere Fehler:

Die folgenden Fehler beziehen sich alle auf die Benutzung von Dateien, also nicht mehr auf die gegebenen Kommandos selbst.

29 DISK ID MISMATCH SP SE:

Diskettenidentität stimmt nicht.

Hier ist vergessen worden, zu initialisieren; entweder haben Sie vergessen, die Diskette zu initialisieren, oder Sie haben seit dem letzten Initialisieren eine andere Diskette eingelegt. Diese Fehlermeldung kann vermeiden, daß Sie auf eine falsche Diskette zugreifen.

30 WRITE FILE OPEN 0 0:

Datei als Schreibdatei bereits geöffnet.

Sie können zwar eine Datei über mehrere Kanäle lesen, aber nur über einen Kanal schreiben. Versuchen Sie eine Datei zu öffnen, um auf eine bereits als geöffnete zu schreiben, so tritt diese Fehlermeldung auf.

61 FILE NOT OPEN 0 0:

Datei nicht geöffnet.

Sie können nur auf eine Datei zugreifen, die Sie geöffnet haben. Wahrscheinlich hat die Floppy durch aufgetretene Fehler die geöffneten Dateien vergessen, oder aber Sie haben die Floppy zwischendurch ausgeschaltet. Schließen Sie bitte die Datei im Rechner und öffnen Sie sie dann wieder.

ACHTUNG: Erst wenn Sie alle Dateien ordnungsgemäß geschlossen haben, können Sie sicher sein, daß die BAM richtig auf die Diskette geschrieben worden ist.

62 FILE NOT FOUND 0 0:

Datei nicht gefunden.

Die Floppy hat den von Ihnen angegebenen Dateinamen nicht im Inhaltsverzeichnis der Diskette gefunden. Meist handelt es sich um einen Tippfehler beim Dateinamen. Schauen Sie den richtigen Namen im Inhaltsverzeichnis der Diskette nach.

63 FILE EXISTS 0 0:

Datei existiert bereits.

Sie haben hier versucht, eine Datei auf Diskette zu schreiben, deren Name bereits im Inhaltsverzeichnis steht. Entweder müssen Sie einen anderen Dateinamen angeben, oder einen " " zum Überschreiben der bereits vorhandenen Datei einfügen. Die alte Datei geht beim Überschreiben natürlich verloren.

64 FILE TYPE MISMATCH 0 0:

Falscher Dateityp.

Die verschiedenen Dateitypen (SEQ und PRG) dürfen nicht durcheinanderkommen. Bei diesem Fehler haben Sie entweder zwei Dateien gleichen Namens, aber verschiedenen Dateityps verwechselt, oder einfach bei der Aufgabe des Typs einen Fehler gemacht.

65 NO BLOCK SP SE:

Block nicht mehr frei.

Sollten Sie versuchen, einen Block zu belegen, der bereits belegt ist, so erscheint die obige Fehlermeldung. SP und SE geben Spur und Sektor des nächsten freien Blocks an. Ist hinter dem Block kein freier Block mehr zu finden, so stehen BL und SE auf 0.

70 NO CHANNEL 0 0:

Kanal nicht mehr frei.

Diese Fehlermeldung tritt auf, wenn Sie versucht haben, einen Kanal oder einen Puffer zu belegen, der bereits belegt ist. Haben Sie im "OPEN" zum Öffnen des Puffers als Puffernummer nur ein #, so bedeutet diese Fehlermeldung, daß überhaupt kein Puffer mehr frei ist.

71 DIR ERROR SP SE:

Fehler im Inhaltsverzeichnis.

Hier handelt es sich um einen Fehler im Inhaltsverzeichnis einer Diskette. Die Behandlung kann die gleiche sein wie bei den READ ERROR-Meldungen.

72 DISK FULL 0 0:

Diskette voll.

Die Floppy meldet sich, wenn auf einer Diskette kein Block mehr frei ist. Die letzte geschriebene Datei wird aller Wahrscheinlichkeit nach nicht mehr vollständig übertragen worden sein. Hier hilft nur eine neue Diskette zu verwenden.

Diskettenprüfprogramm

```
1 REM CHECK DISK --VER 1.2
2 DN=8:REM FLOPPY DEVICE NUMBER
5 DIMT(100):DIMS(100):REM BAD TRACK, SECTOR ARRAY
10 PRINT"  "TAB(9)"  DISKETTENPRÜFUNG"
20 INPUT"  ZU PRÜFENDES LAUFWERK";D$
30 OPEN15,DN,15
35 PRINT#15,"V"D$
40 PRINT"  PRÜFE LAUFWERK "D$
45 N%=RND(TI)*255
50 A$="":FORI=1TO255:A$=A$+CHR$(255AND(I+N%)):NEXT
60 GOSUB900
70 OPEN2,DN,2,"#"
80 PRINT:PRINT#2,A$;
85 T=1:S=0
90 PRINT#15,"B-A:"D$;T;S
100 INPUT#15,EN,EM$,ET,ES
110 IFEN=0THEN130
115 IFET=0THEN200:REM END
120 PRINT#15,"B-A:"D$;ET;ES:T=ET:S=ES
130 PRINT#15,"U2:2,"D$;T;S
134 NB=NB+1:PRINT"GEPRÜFTE BLOCKS: "NB
135 PRINT"PRÜFE SPUR "T", SEKTOR "S":I
140 INPUT#15,EN,EM$,ES,ET
150 IF EN=0THEN85
160 T(J)=T:S(J)=S:J=J+1
165 PRINT"  FEHLER BLOCK:"T",T,S
170 GOTO85
200 PRINT#15,"I"D$
210 PRINT"  INIT LAUFWERK "$:GOSUB900
212 CLOSE2
215 IFJ=0THENPRINT"  KEIN FEHLER!"
217 OPEN2,DN,2,"#"
218 PRINT"  FEHLER=","  SPUR  ","  SEKTOR  "
220 FORI=0TOJ-1
230 PRINT#15,"B-A:"D$,T(I);S(I)
240 PRINT,,T(I),S(I)
250 NEXT
260 PRINT"  J"FEHLERBLOCKS BELEGT"
270 CLOSE2:END
900 INPUT#15,EN,EM$,ET,ES
910 IF EN=0 THEN RETURN
920 PRINT"  INPUT ERROR #"EN,EM$,ET,ES
930 INPUT"CONTINUE "IN$:IFIN$="" THENRETURN
940 END
READY.
```

Stichwortverzeichnis

.L	4.5.	Laden eines Maschinenprog.	4.0.
.S	4.4.	Laden eines Programms	4.2.
Adressen der Puffer	7.12.	Löschen (Dateien)	3.5.
B-A	7.8.	Löschen (Diskette)	3.7.
B-E	7.10.0.	LOAD	4.2.
B-F	7.8.	M-E	7.10.1.
B-P	7.6.0.	M-R	7.9.1.
B-R	7.7.1.	M-W	7.9.0.
B-W	7.7.0.	M-W	7.9.0.
Begrenzungszeichen	1.1.3.3.	Maschinenprog. (Laden)	4.5.
Belegen (Blöcke)	7.8.	Maschinenprog. (Speichern)	4.4.
Blockverteilung	2.1.	Maschinenroutine (Ausführen)	7.10.1.
Blöcke	2.1.	Maschinenroutinen	7.4.
BAM	2.2.	N	3.7.
BIP	1.1.3.1.	OPEN	1.1.0.
C	3.3.	Programm (Laden)	4.2.
CHECK DISK	0.5.1.	Programm (Speichern)	4.1.
CLOSE	1.1.1.	Programm (Überprüfen)	4.3.
CLOSE	2.2.0.	Programmname	4.1.
COPY DISK FILES	0.5.1.	Programmname	4.2.
D	3.2.	Puffer	7.5.
Datei	2.0.	Puffer (Adressen)	7.12.
Dateiname	2.2.2.	Puffernummer	7.5.0.
Dateiname	3.0.	Pufferzeiger	7.6.
Dateityp '*PRG'	4.1.	PET DISK	0.5.1.
Daten (technische)	0.0.2.	PRG-Datei	2.0.0.
Datensicherung	0.4.	PRG-Datei	4.0.
Delimiter	1.1.3.3.	PRINT#15,"	1.1.2.
Diagnoseprogramm	0.5.0.	R	3.4.
Direktzugriff	2.0.2.	Resident Monitor	4.4.
Direktzugriff	7.	Resident Monitor	4.5.
Diskette	0.3.	S	3.5.
Diskette	0.4.	Schreibschutz	0.5.0.
Diskettenname	2.2.	Schreibschutz	3.2.
Diskettenname. Lesen des	7.13.	Sektor	2.1.
Duplizieren	3.2.	Speicher (Auslesen)	7.9.1.
DIAGNOSTIC BOOT	0.5.0.	Speicher (Schreiben in)	7.9.0.
DIM 3.4.	0.5.1.	Speichern (Maschinenprog.)	4.4.
Fehler	8.0.	Speichern eines Programms	4.1.
Fehlermeldung	8.0.	Sprungtabelle	7.11.0.
Floppykommandos	3.	Spur	2.1.
Formatieren	0.5.0.	Standardsprungtabelle	7.11.1.
Formatieren	3.7.	SAVE	4.1.
Freigeben (Block)	7.8.	SEQ-Datei	2.0.1.
GET#	1.1.3.2.	Test-/Demo-Diskette	0.5.
GET#	1.1.4.	Überprüfen (Diskette)	3.6.
I	3.1.	Überprüfen eines Programms	4.3.
Inhaltsverzeichnis	0.5.0.	Umbenennen	3.4.
Inhaltsverzeichnis	2.2.	User	7.11.
Inhaltsverzeichnis	2.2.1.	Utility-Programme	0.5.1.
Inhaltsverzeichnis	2.2.2.	V	3.6.
Inhaltsverzeichnis	4.1.	Verweisdatei	7.0.3.
Initialisieren	0.5.0.	VERIFY	4.3.
Initialisieren	3.1.		
ID (Identität)	0.5.0.		
ID (Identität)	2.2.		
ID. Lesen der	7.13.		
IEC-Bus	0.2.1.		
INPUT#	1.1.3.		
Kanal	6.2.		

Nachdruck, auch auszugsweise, nur mit schriftlicher Genehmigung von Commodore.



commodore

Commodore Büromaschinen GmbH
Postfach 426
6078 Neu-Isenburg
Telefon (06102) *27042 · Telex 04185663 como d