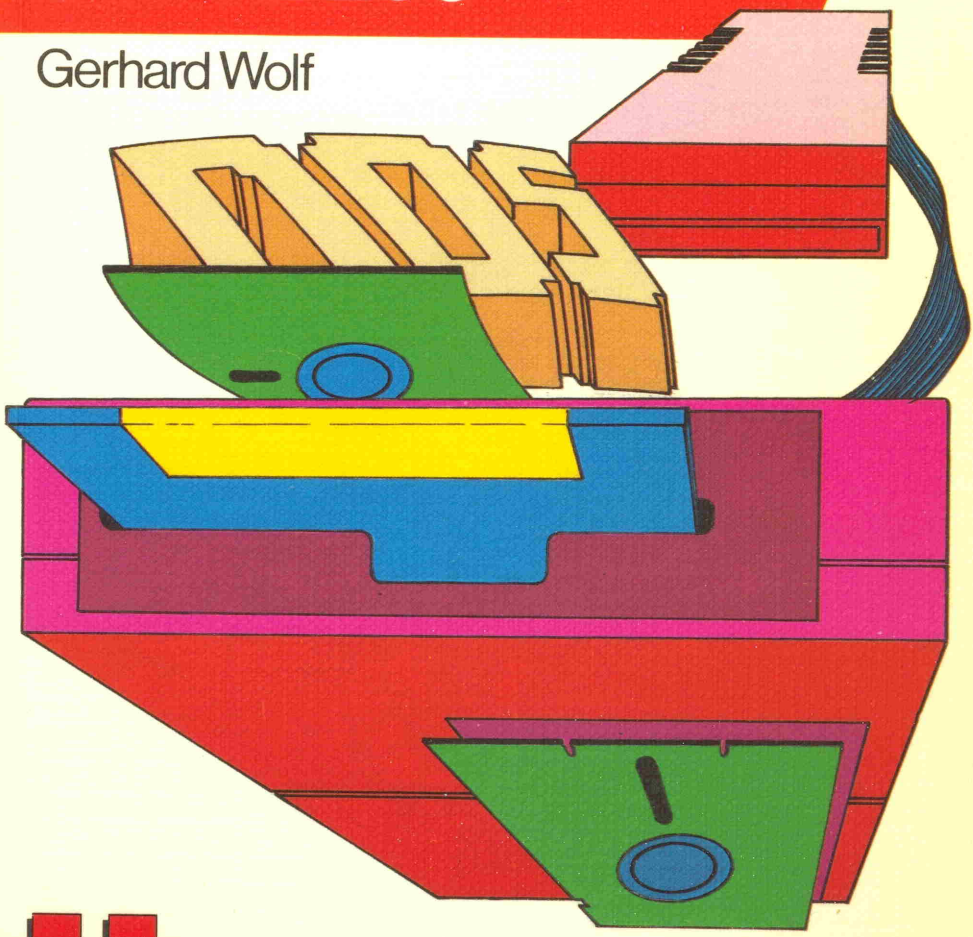
aktiv und kreativ computern

Das Laser-DOS für Laser 110, 210, 310 und VZ 200

Gerhard Wolf



Mein Home-Computer

Gerhard Wolf

Das Laser-DOS für Laser 110, 210, 310 und VZ 200

HC – Mein Home-Computer

Gerhard Wolf

Das Laser-DOS für Laser 110, 210, 310 und VZ 200

Aufbau und Anwendung des Disketten-Betriebssystems

VOGEL-BUCHVERLAG
WÜRZBURG

Vom selben Autor sind erschienen:

ROM-Listings für Laser 110, 210, 310 und VZ 200

(HC – Mein Home-Computer)

ISBN 3-8023-0852-2

Der BASIC-Interpreter im Laser 110, 210, 310 und VZ 200

(HC – Mein Home-Computer)

ISBN 3-8023-0874-3

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Wolf, Gerhard:
Das Laser-DOS für Laser 110, 210, 310 und VZ 200: Aufbau u. Anwendung d. Disketten-Betriebssystems / Gerhard Wolf. – 1. Aufl. – Würzburg: Vogel, 1985.
(HC – Mein Home-Computer)
ISBN 3-8023-0868-9

ISBN 3-8023-0868-9
1. Auflage. 1985

Printed in Germany

Umschlaggestaltung: Bernd Schröder, Böhl
Herstellung: Alois Erdl KG, Trostberg


LASER
COLOR COMPUTER 310
CTRL
SHIFT
RETURN
SPACE


LASER
COLOR COMPUTER 310
RETURN
CTRL
SHIFT

0. Einleitung

Um eine Sache zu verstehen und zu durchschauen, sind solide Grundkenntnisse unumgänglich. Erst dann ist man in der Lage, Zusammenhänge erkennen und richtig verwerten zu können.

Dieser Grundsatz gilt auch - oder erst recht - für den Einsatz eines Diskettensystems an einem Homecomputer, von dem man annehmen kann, daß er nicht in erster Linie für kommerzielle Zwecke beschafft wurde, sondern vordringlich, um einem Hobby zu frönen oder sich auf dem Gebiet der Datenverarbeitung fortzubilden.

Das vorliegende Buch befaßt sich speziell mit dem Anschluß eines Diskettensystems an einen LASER 110, 210, 310 oder einen VZ200.

Es beschreibt ausführlich die Grundlagen und den Aufbau des Systems und befaßt sich eingehend mit den Möglichkeiten des mitgelieferten Disketten-Betriebssystems (Disk Operating System = DOS).

Es wendet sich an alle Disketten-Nutzer, ob Computer-Neuling oder versierter "Freak".

In den ersten Kapiteln werden Grundlagen dargestellt und es wird anschaulich der Einsatz im Rahmen der verfügbaren BASIC-Sprache beschrieben. Es folgen dann die byte- (teilweise bit-) genaue Beschreibung der Datenorganisation auf der Diskette und eine ausführliche Darlegung der Disketten-Möglichkeiten für Assembler- und Maschinenprogramm - Könner.

Lassen Sie sich davon jedoch nicht abschrecken, falls Sie das erste Mal mit Disketten konfrontiert sind. Der Aufbau dieses Buches erlaubt den schrittweisen Einstieg in die Materie.

Beginnen Sie nach dem Studium der Grundlagen mit der Nutzung der allgemeinen DOS-Anweisungen und der Programm-/Datei-Verwaltung, indem Sie die Diskette zunächst nur als Speichermedium für Ihre Programme verwenden.

Sind Sie darin sicher, so probieren Sie, eigene Daten aus BASIC-Programmen auf der Diskette zu speichern und anschließend wieder zu verarbeiten.

Den beiden letzten Kapiteln sollten Sie sich nur zuwenden, wenn Ihnen die DOS-Anwendung im BASIC völlig vertraut ist und Sie einige Grundkenntnisse in der Z80--Assembler- und Maschinensprache-Programmierung erworben haben.

1. Das LASER 110, 210 und 310-Disketten-System

Komponenten

Das Disketten-Laufwerk (Floppy Disk Drive) **LASER DD20** wurde speziell für den Einsatz an einem der LASER-Computer 110, 210 und 310 entwickelt; es eignet sich jedoch auch für den VZ200. Diese Disketten-Laufwerke sind nicht kompatibel zu anderen Computer-Systemen (z.B. dem LASER 2001 oder LASER 3000 o.a.). Umgekehrt sind andere Disketten-Laufwerke auch nicht für diese Rechner geeignet.

Die Verbindung zu den Rechnern wird mit Hilfe einer Diskettensteuerung (Floppy Disk Controller) **LASER DI40** hergestellt, die an den System-Bus der Rechner angeschlossen wird und auf der "huckepack" eine Speichererweiterung eingesteckt werden kann (Bild 1.1).

Zwei Laufwerke können gleichzeitig an einer Diskettensteuerung angeschlossen und betrieben werden. Die Diskettensteuerung enthält auch die erforderlichen Systemprogramm-Erweiterungen, das Disketten-Betriebssystem (Disk Operating System = DOS).

Beim LASER 110-Computer und beim VZ200 mit internem 4K-RAM ist zusätzlich eine Speichererweiterung von mindestens 16K erforderlich.

Bild 1.1 LASER 310, Diskettensteuerung, 16K-Speichererweiterung und Laufwerk

Grundlagen der Disketten-Speicherung

Das Laufwerk

Es gibt eine große Anzahl unterschiedlicher Diskettensysteme der verschiedensten Hersteller. Eine der Hauptklassifizierungen dieser Laufwerke ist die Größe der zu verwendenden Disketten. Diese variieren von 3 1/2 Zoll über 5 1/4 Zoll bis zu 8 Zoll. Bei dem Diskettenlaufwerk LASER DD20 handelt es sich um ein 5 1/4 Zoll - Laufwerk, das von der Aufmachung und auch dem technischen Aufbau her sehr stark den TEAC-Laufwerken ähnelt.

Disketten-Laufwerke arbeiten mit einer runden, rotierenden Scheibe (Diskette genannt), die, wie ein Tonband, mit einer magnetisierbaren Schicht überzogen ist. In diese werden die Daten mittels eines Schreib-/Lesekopfes eingeschrieben bzw. wieder ausgelesen.

Um auf eine beliebige Stelle dieser Scheibe zugreifen zu können, muß der Schreib-/Lesekopf beweglich sein. Dazu ist er auf einer Schiene gelagert und läßt sich quer zur Scheibe verschieben.

Um Daten zu schreiben oder zu lesen wird der Kopf einfach die gewünschte Strecke nach innen oder außen verschoben und dann gewartet, bis die gewünschten Daten durch die Rotation darunter vorbeikommen (Bild 1.2).

Der Übersichtlichkeit und Wiederauffindbarkeit halber hat der Kopf auf dem Schlitten feste Rasterpositionen, was wiederum konzentrische Datenkreise auf der Diskette ergibt. Die verschiedenen Diskettensysteme haben zwischen 35 und 80 Rasterungen, beim LASER DD20 sind es 40.

Die sich daraus ergebenden Kreise mit Daten auf der Diskette nennt man **Spuren** (im englischen "Tracks"). Jede einzelne Spur kann durch den Einstellmechanismus des Kopfes exakt unmittelbar erreicht werden.

Innerhalb einer Spur werden die Daten Bit für Bit hintereinander aufgezeichnet, das heißt in der EDV-Sprache "sequentiell". Dadurch dauert es naturgemäß, nach Positionierung des Kopfes über der Spur, noch durchschnittlich eine halbe Umdrehung der Scheibe, bis die gewünschten Daten erreicht sind.

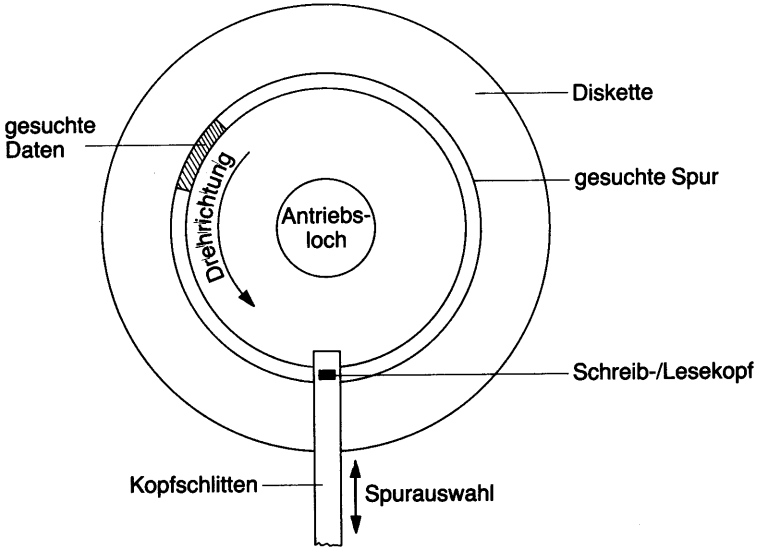


Bild 1.2 Datenzugriff bei einer Diskette

Die Zeit, die man zum Zugriff auf die gewünschten Daten benötigt, richtet sich also danach, wie schnell der Kopf auf die bestimmte Spur gestellt werden kann und wie schnell die Diskette rotiert.

Beim LASER DD20 wird die Diskette mit 80 Umdrehungen pro Minute angetrieben. Die Zeit, den Kopf von einer Spur zur nächsten zu bewegen, beträgt ca. 20 Millisekunden. Dies ergibt eine mittlere Zugriffszeit von 500 Millisekunden.

Während eines Schreib-/Lesevorgangs muß die Diskette wie bei einem Tonband fest an den Schreib-/Lesekopf angepreßt werden. Das wird mit einem Filzstückchen bewerkstelligt, welches über einen Hebelarm die Diskette von oben auf den Kopf drückt. Dieser Hebelarm ist mit dem Verschlußhebel an der Vorderseite des Laufwerks verbunden. Steht dieser in senkrechter Stellung (zu), so wird die Diskette an den Kopf gepreßt; in waagerechter Stellung ist die Diskette frei (Bild 1.3).
Eine elektronische Kopflade-Prozedur, wie sie bei vielen anderen Laufwerken üblich ist, gibt es hier nicht.

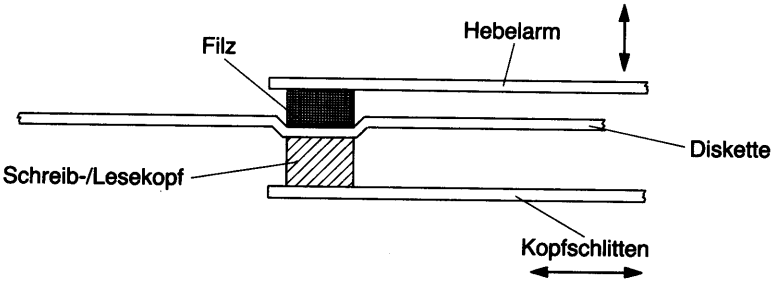


Bild 1.3 Darstellung einer Diskette im Laufwerk

Wenn Sie jetzt aufgepaßt haben, so müßte Ihnen aufgefallen sein, daß eine Diskette immer von unten beschrieben wird, also anders, als man beim Hineinschieben eigentlich denkt.

Mit dem Verschlußhebel ebenfalls verbunden ist der Antriebsmechanismus. Beim Schließen des Hebels (senkrecht) wird das Antriebsloch der Diskette auf einen Konus zentriert, der über einen Riemen vom Motor angetrieben wird. Wie genau eine Diskette auf dem Konus zentriert wird, ist eine der kritischen Komponenten des Laufwerks.

Die Lage einer Spur bezieht sich immer auf den Mittelpunkt der Diskette. Daher ist das sichere Schreiben und Wiederauffinden der Daten sehr stark davon abhängig, wie genau die Zentrierung der Diskette erfolgt. Leider tendiert das Laufwerk **LASER DD20** etwas dazu, die Disketten nicht paßgenau auf den Konus zu pressen. Beachten Sie bitte die im Abschnitt "Einlegen einer Diskette" erwähnten Hilfs- und Kontrollmöglichkeiten.

Der Laufwerkmotor wird zum Schutz der Disketten nur unmittelbar vor den Schreib- bzw. Leseoperationen eingeschaltet und anschließend sofort wieder ausgeschaltet. Sie erhalten über diesen Vorgang eine optische Anzeige in Form einer aufleuchtenden LED (light emitting diode) auf der Vorderseite des Laufwerks. Leuchtet dieses, sollten Sie keine Diskette entnehmen oder einlegen (siehe Anpreßvorgang).

Die Diskette

Aufbau

Bedingt durch den Anpreßvorgang der Diskette an den Schreib-/Lesekopf ist es ersichtlich, daß keine feste Scheibe als Grundmaterial verwendet werden kann. Es wird ein flexibles, formstabiles Kunststoffmaterial benutzt (meist eine Mylarfolie), das mit einer Eisenoxidbeschichtung versehen ist. Daraus leitet sich auch der englische Name "floppy disk" = flexible Scheibe ab.

Zur besseren Handhabung und zum Schutz der Beschichtung ist die Scheibe in eine feste Hülle mit einer filzartigen Auskleidung verpackt.
Die Auskleidung erfüllt drei wesentliche Aufgaben:

- Herabsetzen der Reibung zwischen Diskette und Hülle
- Ableiten statischer Aufladungen, die durch die Rotation entstehen
- Aufnahme eingedrungener Staub- und Schmutzteilchen zum Schutz der Beschichtung

Die Diskette verbleibt ständig in dieser Hülle, die zur Handhabung drei öffnungen besitzt:

- ein großes Loch in der Mitte für den Antriebsmechanismus
- ein ovaler Ausschnitt, durch den der Kopf auf die Beschichtung
- zugreifen kann
- eine kleine öffnung als Indexloch, das bei vielen Laufwerken zur physikalischen Spuranfangskennung genutzt wird (nicht beim LASER DD20).

Am Rand der Hülle befindet sich noch eine kleine rechteckige Aussparung. Dabei handelt es sich um eine Schreibschutzeinrichtung, mit der Sie den Inhalt einer Diskette schützen können. Wird diese Aussparung mit einem der den Diskettenpackungen beiliegenden Klebestreifen überklebt, so ist das Schreiben auf diese Diskette gesperrt (Bild 1.4).

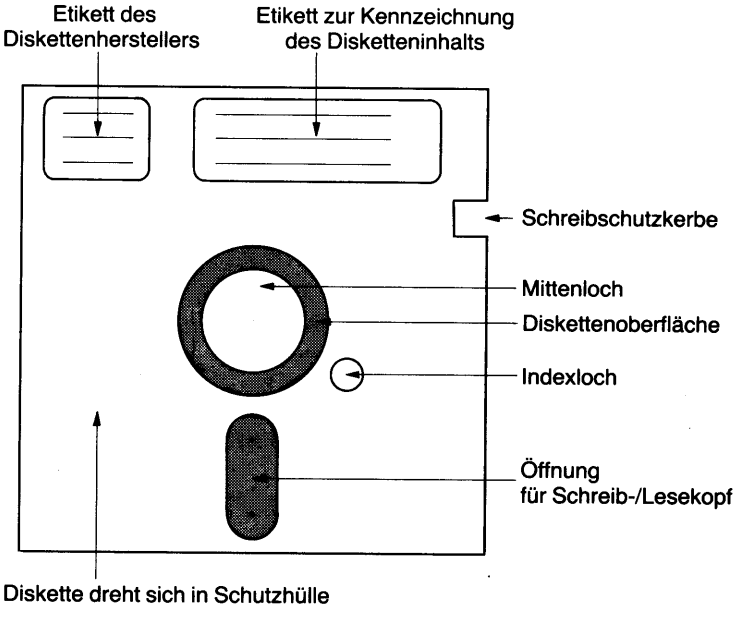


Bild 1.4 Diskette

Der eigentliche Datenträger besteht nur aus einer sehr dünnen Beschichtung aus Eisenoxiden und ist besonders behandelt, um eine vertretbar lange Zeit mit dem Schreib-/Lesekopf in Kontakt bleiben zu können. Dennoch wird diese Schicht bei ständigem Kopfkontakt mit der Zeit abgeschliffen. Als Anhaltswert kann eine Laufzeit von 50 - 60 Stunden dienen. Das ergibt eine doch recht lange Lebensdauer, da die Diskette ja nur bei Schreib-/Lese-Zugriffen läuft und der Kopf auch nicht immer an derselben Spur aufliegt.

Aufgrund dieses Abriebs bezeichnen böse Zungen die Diskettenverarbeitung auch als **'spanabhebende Datentechnik'.**

Einlegen der Diskette

Zum Einlegen einer Diskette wird der Verschlußhebel an der Laufwerköffnung waagerecht gestellt. Die Diskette wird nun mit dem ovalen Kopfausschnitt nach vorn und im Normalfall mit dem Etikett nach oben in das Laufwerk bis zum Anschlag eingeschoben.

Durch Drehen des Verschlußhebels in die senkrechte Stellung wird das Antriebsloch der Diskette auf den Antriebskonus des Laufwerks zentriert aufgepreßt und die Diskette an den Schreib-/Lesekopf gedrückt.

Normalerweise ist dies ausreichend, und es kann mit den Schreib-/Lesevorgängen begonnen werden.

Wie jedoch zuvor erwähnt, kann es zu Schwierigkeiten kommen, wenn die Diskette schief auf den Konus geklemmt wird, und dies passiert beim **LASER DD20** leider häufig.

Man sollte sich daher folgende Punkte zur Regel machen:

- Verwenden Sie nur Disketten mit einem Verstärkungsring um das Antriebsloch. Dies verhindert einen vorzeitigen Verschleiß am Antriebsloch durch schiefes Einklemmen am Konus.

- Zentrieren Sie die Diskette vor dem Einlegen mit der Hand so gut wie möglich in der Hülle.

- Testen Sie, bevor Sie etwas auf eine initialisierte Diskette schreiben, diese zuvor mir dem **DIR** - Kommando.

- Nach einer Neuinitialisierung entnehmen Sie die Diskette, legen sie neu ein und testen ebenfalls mit dem **DIR** - Kommando.

Sollte trotzdem bei einem Lesevorgang das Laufwerk immer wieder versuchen, den Kopf neu zu positionieren (das hören Sie an etwas lauteren Knackgeräuschen), so öffnen Sie bei laufendem Motor kurz den Verschlußhebel und schließen ihn sofort wieder. In aller Regel zieht sich die Diskette bei laufendem Motor sauber auf den Konus.

Eine Diskette hat zwei Seiten

Welche Disketten beschaffen Sie nun für den **LASER DD20**?

Wie bereits erwähnt, handelt es sich um ein 5 1/4 Zoll Laufwerk, also benötigen Sie auch 5 1/4-Zoll-Disketten. Doch da gibt es auch wieder viele verschiedene Sorten, hardsektoriert oder softsektoriert, einseitig oder doppelseitig, mit einfacher Dichte oder mit doppelter Dichte.

Um es auf einen Nenner zu bringen: Sie benötigen 5 1/4 Zoll - Disketten, softsektoriert, einseitig, mit einfacher Dichte. Diese tragen die Bezeichnung "**SSSD**"; das steht für "single sided, single density".

Disketten doppelter Dichte (SSDD) können Sie auch verwenden. Diese sind etwas eingehender geprüft, etwas teurer aber für das beim **LASER DD20** gewählte Aufzeichnungsverfahren nicht unbedingt erforderlich.

Wenn Sie sich nun eine solche "einseitige" Diskette genau betrachten, so wird Ihnen auffallen, daß trotzdem beide Seiten beschichtet sind und auch die ovale öffnung für den Kopf auf beiden Seiten der Hülle vorhanden ist.
Dies bedeutet, daß man prinzipiell beide Seiten der Diskette beschreiben kann.

Das Laufwerk **LASER DD20** ist jedoch nur mit einem Schreib-/Lesekopf ausgestattet. Man müßte also die Diskette drehen, um diesen an die andere Seite zu bringen. Tun Sie dies und versuchen Sie nun, etwas auf diese Seite zu schreiben, so wird Ihnen die Meldung "?DISK WRITE PROTECTED" ausgegeben. Dies liegt an der fehlenden Schreibschutzkerbe an der anderen Seite der Hülle. Erinnern Sie sich, daß eine Diskette schreibgeschützt ist, wenn Sie die Schreibschutzkerbe überkleben.
"Keine Schreibschutzkerbe" hat natürlich den gleichen Effekt.

Um die zweite Seite nutzen zu können ist also nur eine zweite Schreibschutzkerbe erforderlich, die Sie sehr leicht mit einem Locher an der Hülle anbringen können. Nehmen Sie eine andere Diskette als Schablone. Keine Angst, daß Sie die Diskette selbst beschädigen, diese reicht nicht so weit in die Ecken der Hülle (Bild 1.5).

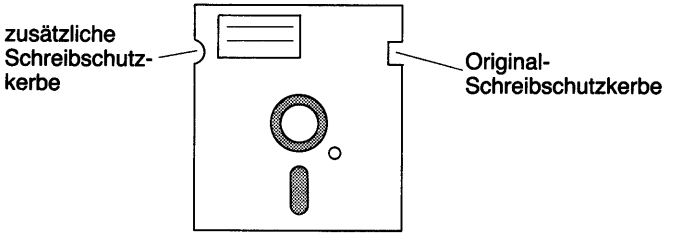


Bild 1.5 Doppelseitige Nutzung der Diskette

Damit haben Sie pro Diskette 80000 Byte zusätzlichen Speicherplatz geschaffen. Sie müssen allerdings die Diskette umdrehen, wenn Sie die Rückseite beschreiben bzw. lesen wollen.

Behandlung der Disketten

Um Ihre Daten auf den Disketten so gut wie möglich vor Zerstorung zu bewahren, sollten Sie die folgenden Regeln unbedingt beachten:

- Verwahren Sie die Disketten stets in ihren Schutzhüllen, wenn diese außerhalb des Laufwerks sind.

- Achten Sie darauf, daß sich keine Diskette im Laufwerk befindet, wenn Sie die Stromversorgung ein- oder ausschalten.

- Bringen Sie Ihre Disketten nie in die Nähe starker magnetischer Felder (Transformatoren, Motoren, Magnete, Fernseher/Monitor, Radios usw.); die dort ausgestrahlten magnetischen Felder könnten den Dateninhalt zerstören.

- Fassen Sie die Diskette ausschließlich an der Hülle an. Vermeiden Sie jede Berührung mit der magnetisierbaren Beschichtung. Versuchen Sie auch nicht, die Beschichtung zu reinigen. Kratzer sind schnell in der Oberfläche, und Sie können dann die Diskette vergessen.

- Setzen Sie eine Diskette nie dem direkten Sonnenlicht oder größerer Hitze aus.

- Vermeiden Sie die Verschmutzung der Beschichtung mit Zigarettenasche, Staub oder anderen Sachen.

- Benutzen Sie ausschließlich einen Faserstift, wenn Sie das Etikett auf der Hülle beschriften wollen. Kugelschreiber oder Bleistifte könnten durch die Hülle die Beschichtung verletzen.

- Verwahren Sie Disketten nach Möglichkeit senkrecht auf (wie Schallplatten), so daß ein Druck auf die Seiten vermieden wird.

Tips zur Diskettenbeschriftung.
Jede Diskette trägt auf ihrer Hülle ein fest angebrachtes Etikett. Sie sollten dieses nur für wichtige Informationen verwenden, die sich während der Lebensdauer einer Diskette nicht ändern. So ist es z.B. sehr hilfreich, die Disketten mit einer fortlaufenden Nummer zur Archivierung zu versehen. Diese hätte dort ihren besten Platz. Weitere sinnvolle Daten sind u.a. auch Ihr Name, sowie das Datum der ersten Nutzung dieser Diskette.

Für Inhaltsangaben benutzen Sie am besten die jeder Diskettenpackung beiliegenden Klebeetiketten, die Sie auch einmal leicht auswechseln können. Wenn Sie keine wichtigen öffnungen damit verkleben, steht Ihnen dazu die ganze Hüllenoberfläche zur Verfügung.

Aufzeichnungsstruktur

Wodurch wird bei einer Diskette die Menge der Daten bestimmt, die man speichern kann? Jedes System hat seine eigene Speicherkapazität für eine Diskette; das geht bei den 5 1/4-Zoll-Disketten bis zu 1/2 Million Bytes (Zeichen) pro Diskettenseite. Beim **LASER DD20** sind es etwas mehr als 80000 Bytes.

Zwei entscheidende Faktoren beeinflussen die Speicherkapazität. Dies ist einmal die Anzahl der Rasterungen, mit denen sich der Kopf über der Diskette bewegt und die gleich der Anzahl der zu beschreibenden Datenspuren auf der Diskette ist. Es sind z.Zt. Variationen zwischen 35 und 80 bei verschiedenen Systemen bekannt.
Beim **LASER DD20** sind es 40 Spuren.
Der zweite Faktor ist die Art, wie die einzelnen Bits auf die Diskette geschrieben werden. Hier unterscheidet man zwischen "einfacher Dichte" (FM) und "doppelter Dichte" (MFM). Doppelte Schreibdichte ergibt auch etwa die doppelte Kapazität. Beim **LASER DD20** wird, wie bereits erwähnt, mit einfacher Dichte aufgezeichnet.

Doch könnte trotzdem die Speicherkapazität fast doppelt so groß sein, würde man die Daten genauso, wie sie im Speicher stehen, ohne weitere Maßnahmen hintereinander auf die Diskette schreiben. Damit bekommt man zwar viele Daten auf die Diskette, kann aber nicht mehr viel damit anfangen. Wie wollte man in einem Wust von Bits eine ganz bestimmte Information herausfinden, ohne immer alles von Anfang an durchgehen zu müssen.

Die Vorteile der Diskettenspeicherung kommen erst dann zum Tragen, wenn die Aufzeichnungen sinnvoll gegliedert werden, indem man sie in kleine überschaubare Einheiten unterteilt, von denen man weiß, wo sie auf der Diskette stehen. Nur so läßt sich die unmittelbare Zugriffsmöglichkeit auch ausnutzen. Dies heißt nichts anderes, als daß man die Aufzeichnungen formatieren muß.

Eine solche Formatierung wird erreicht, indem man die Aufzeichnung auf der Diskette innerhalb der 40 verschiedenen Spuren nochmals, wie bei einer Torte, in 16 gleiche Abschnitte (Sektoren) aufteilt. Jeder einzelne dieser Sektoren ist getrennt adressierbar und kann individuell behandelt werden.

Jede Spur besteht also aus 16 Sektoren, in denen jeweils 128 Datenbytes untergebracht werden können (Bild 1.6). Dies bedeutet für den **LASER DD20** eine genaue Speicherkapazität von

40 Spuren x 16 Sektoren x 128 Bytes = 81920 Bytes

pro Diskettenseite.

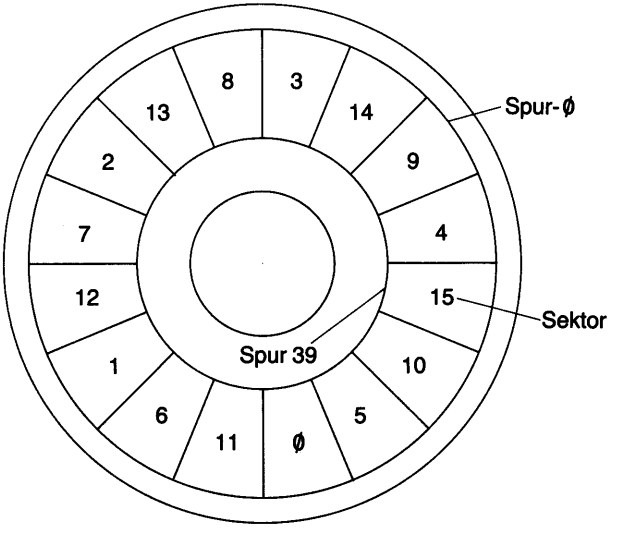


Bild 1.6 Anordnung der Spuren und Sektoren auf einer Diskette

Dies ist jedoch nicht alles, was auf einer formatierten Diskette gespeichert werden muß.

Um auf einen Sektor ohne große Umstände direkt zugreifen zu können, muß man wissen, wann die gesuchte Information gerade unter dem Kopf vorbeikommt.

Dazu erhält jeder Sektor einen Vorspann, ein sogenanntes Adreßfeld, in dem die Sektornummer und, um Kopfeinstellfehler zu erkennen, auch die Spurnummer vermerkt ist.

Zum Erkennen von Aufzeichnungsfehlern innerhalb eines Sektors wird am Sektorende noch ein Prüfsummenfeld angefügt.

Doch dies allein reicht auch noch nicht. Nur selten steht der Kopf gerade da, wo ein neues Byte auf der Spur beginnt. In aller Regel wird er mitten in einem Byte mit dem Lesen beginnen. Da die Daten aber lückenlos Bit für Bit hintereinander gespeichert sind, ist es unmöglich, den Anfang eines Bytes zu identifizieren.
Dies heißt, das zunächst einmal ein Aufzeichnungsbeginn gefunden wird. Man spricht hierbei von einer Synchronisation des Kopfes.

Dazu werden besonders festgelegte Bitfolgen und Aufzeichnungsmarken auf die Diskette geschrieben, die ein einfach zu erkennendes Muster tragen.

Es gibt zwei verschiedene Arten dieser Marken. Eine steht vor jedem Sektor- Adreßfeld, das ist die "Adreßmarke" ;eine zweite steht vor jedem Datenfeld eines Sektors, die "Datenmarke".

Jeder dieser Marken gehen Synchronisationsbytes voraus, und auf die Marken folgen unmittelbar die Daten. Damit kann man eindeutig unterscheiden, ob man sich vor einer Datenaufzeichnung oder vor einem Adressfeld befindet.

Weiterer Platz geht auf der Diskette durch "Aufzeichnungslücken" verloren, die sich hinter jedem Datenfeld eines Sektors befinden. Diese Lücken sind dringend erforderlich, um Schwankungen der Umdrehungsgeschwindigkeit in bestimmten Grenzen ausgleichen zu können (Bild 1.7).

Eine solche Grundstruktur der Diskette muß vor jedem Beschreiben mit Daten zunächst hergestellt werden. Diesen Vorgang nennt man "Initialisierung"; dafür steht ein eigenes Kommando zur Verfügung. Bei einer Initialisierung wird die Unterteilung in Sektoren vorgenommen, und es werden sämtliche Adreß- und Datenmarken geschrieben.

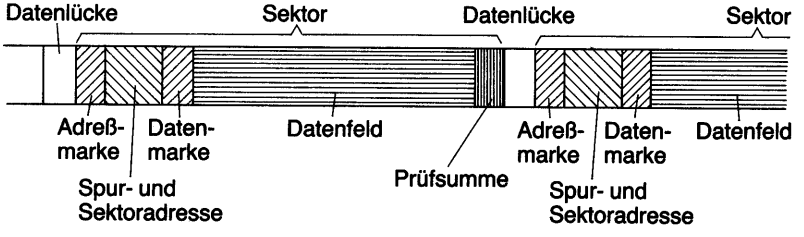


Bild 1.7 Datenstruktur auf einer Diskette

Aus der Bild 1.6 ist ersichtlich, daß die Sektoren nicht fortlaufend von 1 bis 16, sondern in 3er Sprüngen auf der Diskette numeriert sind. Durch diesen kleinen Trick ist es möglich, während einer Umdrehung der Diskette mehrere hintereinanderliegende Sektoren lesen zu können und damit den Zugriff erheblich zu beschleunigen.

Nach diesen Ausführungen sollte verständlich sein, wie vom Rechner jeder einzelne Sektor auf der Diskette gefunden werden kann.
Sie wollen aber in der Regel garnichts von einzelnen Sektoren wissen, Sie suchen ein ganz bestimmtes Programm auf der Diskette oder eine Datei, die Sie dort angelegt haben. Auf einer Diskette werden Sie im Regelfall auch mehr als ein Programm oder eine Datei gespeichert haben. Wie kommt man nun an eine solche komplette Aufzeichnung, ohne selbst Buch über Sektoren führen zu müssen?

Dazu wurde eine ganze Spur der Diskette geopfert. Auf der Spur Ø, der äußersten Spur, ist ein Inhaltsverzeichnis der Diskette angelegt, in dem festgehalten wird, welche Programme und Dateien auf der Diskette gespeichert und wo diese zu finden sind. Mit dem DOS-Kommando "**DIR**" können Sie sich dieses Inhaltsverzeichnis auf den Bildschirm ausgeben lassen.

Der letzte Sektor dieser Spur Ø hat noch eine spezielle Verwendung. In ihm wird für jeden Sektor der Diskette vermerkt, ob er frei oder mit gültigen Daten belegt ist.

Die Diskettensteuerung

Zum Anschluß eines Laufwerks an einen Rechner wird eine Diskettensteuerung (Floppy Disk Controller) vom Typ **LASER DI40** benotigt.

Diese wird an den System-Bus des Rechners angeschlossen und besitzt an der Rückseite zwei 20 polige Steckanschlüsse für den Anschluß von einem oder zwei Laufwerken.

Aufgabe der Diskettensteuerung ist die Umsetzung von logischen Aufträgen, wie z.B. "Lies ein Programm" in Einzelschritte zur spezifischen Ansteuerung der Laufwerke, z.B. Schrittpulse für die Spurverstellung, ein Bit lesen, ein Bit schreiben, Motor ein, Motor aus usw..

Die Diskettensteuerung enthält zusätzlich das Disketten-Betriebssystem (Disk Operating System), kurz **DOS** genannt. Dieses ist in 8K-ROMs gespeichert und erweitert den BASIC Sprachumfang des Rechners um 17 Befehle, die zum Betrieb der Diskettenstation erforderlich sind. Dazu gehören u.a. "**INIT**" zur Disketten-Initialisierung, "**SAVE**" und "**LOAD**" zum Speichern bzw. Laden eines BASIC-Programms.

Die Adressierung dieses Disketten-Betriebsystems erfolgt über die Adressen 4000 - 5FFF (hexadezimal), die für diesen Erweiterungszweck freigehalten wurden. Es wird außerdem am Ende des RAM-Bereichs ein 310 Byte - Arbeitsbereich reserviert.

Installation des Diskettensystems

Grundsätzlich gilt, daß alle Verbindungen von elektronischen Bauelementen nur bei ausgeschalteter Stromversorgung hergestellt oder gelöst weden sollen. Wie schnell ist sonst ein teurer hochintegrierter Baustein durch auftretende Spannungsspitzen zerstört.

Dies trifft auch für den Anschluß des Diskettensystems zu. Stellen Sie also sicher, daß die Stromversorgung des Rechnersystems unterbrochen ist.

Die Diskettensteuerung wird zunächst an den System-Bus auf der Rückseite des Rechners angeschlossen. Das ist die Steckleiste, an der vorher eine evtl. vorhandene Speichererweiterung angeschlossen war. Diese können Sie nun auf der Oberseite der Diskettensteuerung anschließen.

An der Rückseite der Diskettensteuerung befinden sich zwei 20-polige Steckleisten zum Anschluß der Laufwerke. Sie tragen die Bezeichnungen "D1" für Laufwerk 1 und "D2" für Laufwerk 2.

Besitzen Sie nur ein Laufwerk, so schließen Sie dieses an "D1" an.

Zu jedem Laufwerk benötigen Sie ein getrenntes Netzteil. Der 5-polige DIN-Stecker des Netzteils ist an der Rückseite des Laufwerks in die entsprechende Buchse einzustecken.

Verbinden Sie nun die Netzteile mit einer Netzsteckdose, so ist Ihr Aufbau komplett.

Sie sollten darauf achten, daß sich beim Herstellen oder Lösen der Netzverbindung keine Diskette im Laufwerk befindet, da dabei evtl. Dateninhalte zerstört werden könnten.

Die Stromzufuhr zu Ihren Disketten-Laufwerken sollten Sie immer dann unterbrechen, wenn Sie längere Zeit nicht mit dem System arbeiten.

Es empfiehlt sich, für alle Netzanschlüsse des Rechnersystems **eine** 220 Volt - Steckerleiste mit beleuchtetem Ein-/ Ausschalter zu beschaffen. So können Sie bei Abschluß der Arbeiten durch einfachen Tastendruck die gesamte Stromversorgung unterbrechen.

Denken Sie daran, daß bei den Rechnern LASER 110 und VZ200 mit internen 4K RAM mindestens eine 16K-Speichererweiterung vorhanden sein muß.

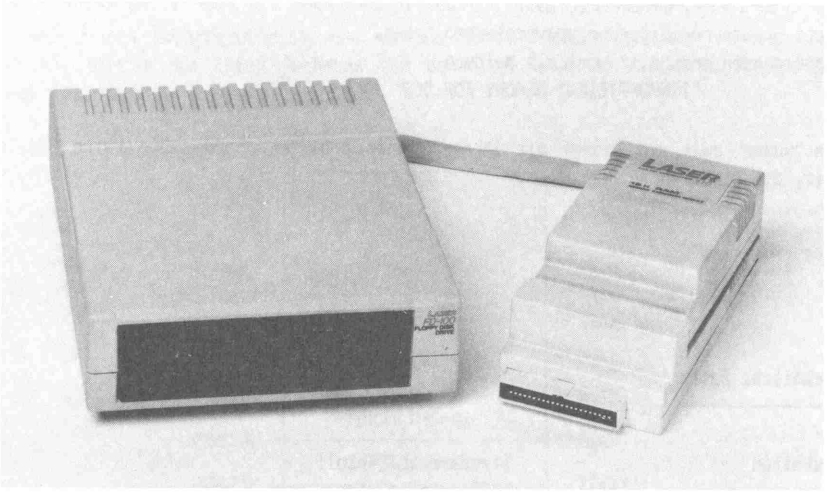


Bild 1.8 Diskettenlaufwerk, 16K-Speichererweiterung und Diskettensteuerung

System-Initialisierung

Nach erfolgreicher Installation schalten Sie Ihren Rechner wie gewohnt ein.

Die Initialisierungsroutine des Rechners erkennt automatisch, daß ein Disketten-System angeschlossen ist und startet kurzzeitig den Motor des an "D1" angeschlossenen Laufwerks. Dabei wird der Schreib-/Lesekopf auf die Spur 0 positioniert (das sind die Klackgeräusche).

Der Text **"BASIC V2.0"** oder **"BASIC V1.2"**

wird durch den Text **"DOS BASIC V1.0"** ersetzt.

Dies ist für Sie das sichere Zeichen, daß Ihr Disketten-Betriebssystem initialisiert ist.

Es stehen Ihnen nun die zuvor erwähnten zusätzlichen 17 Befehle zur Disketten-Bearbeitung zur Verfügung.

Das unterbleibt natürlich, wenn Sie bei einem LASER 110 oder VZ200 keine zusätzliche Speichererweiterung angeschlossen haben.
Statt dessen erscheint dort die Meldung
"?INSUFFICIENT MEMORY FOR DOS".

Sie können zwar wie bisher mit Ihrem Rechner arbeiten, haben aber keine Möglichkeit, die Diskette anzusprechen.

Technische Daten

Disketten	-	Standard 5 1/4 Zoll SSSD (single sided, single density)
Aufzeichnungsformat	-	einseitig einfache Dichte 40 Spuren 16 Sektoren / Spur 128 Byte / Sektor
Kapazität	-	80 KB
Laufwerk	-	80 Umdrehungen / Minute
Stromversorgung	-	+ 5V, +12V Gleichspannung über separates Netzteil

2. DOS - Operationen

Aufbau des LASER-DOS

Das LASER Disketten-Betriebssystem, kurz DOS genannt (Disk Operating System), ist ein zusätzliches ca. 8 KByte großes Programmpaket, das in ROM-Bausteinen im Gehäuse der Diskettensteuerung untergebracht worden ist.

Es belegt den Speicherbereich von Adresse 16384 (4000H) bis zur Adresse 24575 (5FFFH), der in den LASER Rechnern für derartige Erweiterungszwecke freigehalten wurde (Bild 2.1).

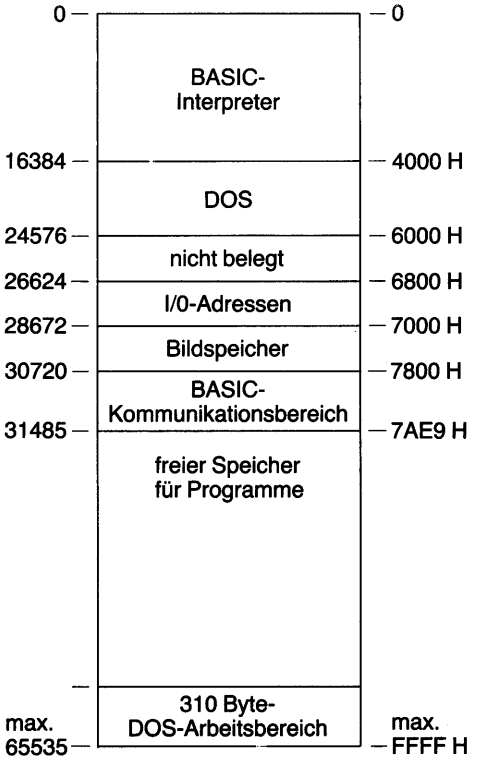


Bild 2.1 Die Speicherbelegung der LASER 110, 210, 310 und des VZ200

Beim Einschalten des Rechners wird das Vorhandensein des Disketten-Betriebssystems automatisch von der Initialisierungsroutine des Standard-ROMs erkannt und mit initialisiert, d.h. in die Ablaufroutinen des BASIC Interpreters eingebettet.

Das Disketten-Betriebssystem besitzt einen eigenen Kommando-Interpreter, der die 17 zusätzlichen Eingabekommandos selbständig erkennt und eigene Ausführungsroutinen anstößt.

Bei den zusätzlichen Befehlen handelt es sich ausschließlich um Diskettenoperationen, die Ihnen das Abspeichern und Wiederauffinden von Programmen und auch das Führen eigener Datenbestände auf der Diskette ermöglicht.

Nutzung der DOS - Kommandos

Kommando - Syntax

Die verfügbaren Kommandos werden ohne jegliche besondere Kennzeichnung wie übliche BASIC - Anweisungen eingegeben.

Die meisten dieser Kommandos können sowohl in BASIC-Programmen, als auch im Direkt-Modus, d.h. zur sofortigen Ausführung vom Bildschirm aus, benutzt werden.

Einige wenige sind jedoch auf den einen oder anderen Modus beschränkt. Wann genau welches Kommando benutzt werden darf, ist bei den Detailbeschreibungen im einzelnen vermerkt.

Von der Syntax her lassen sich die Kommandos in drei Kategorien einordnen:

- Kommandos, die keine Datei ansprechen

  Kommando [parameter]

- Kommandos, die eine Datei ansprechen

  Kommando "Dateiname" [,parameter]

- Kommandos, die zwei Dateien ansprechen

  Kommado "Dateiname1","Dateiname2"

Parameter sind Zusatzinformationen, die von einigen Kommandos benötigt werden. Sind mehrere Parameter erforderlich, so sind diese durch Komma zu trennen.

Eine kleine Einschränkung ergibt sich bei der Anwendung innerhalb von BASIC-Programmen.
Die zusätzlichen Disketten-Befehle werden nicht erkannt, wenn sie bei IF-Anweisungen direkt nach einem **THEN** oder **ELSE** angegeben werden.
Sie müssen stets als eigenständiges Kommando entweder an einem Zeilenanfang oder nach einem Kommandotrenner ":" eingegeben werden.

```
100  IF A = 1 THEN RUN "XYZ"              falsch

100  IF A = 1 THEN :RUN "XYZ"             richtig
```

oder

```
100  IF A <> 1 THEN 120
110  RUN "XYZ"
120  ......
```

Dateitypen und -spezifikationen

Beim LASER-DOS werden drei verschiedene Arten von Dateien unterschieden:

- BASIC-Programmdateien
  mit der Kennzeichnung "T" als Dateityp (=Textdatei).

  In diesem Dateityp werden BASIC-Programme auf der Diskette gespeichert.

- Maschinen-Programmdateien
  mit der Kennzeichnung "B" als Dateityp (=Binär-Datei).

  In diesem Dateityp werden Maschinenprogramme auf der Diskette gespeichert.

- Daten-Dateien
  mit der Kennzeichnung "D" als Dateityp (=Daten).

  In diesem Dateityp werden Ihre persönlichen Daten gespeichert, wenn Sie diese aus einem BASIC-Programm heraus auf der Diskette ablegen wollen.

BASIC- und Maschinenprogramme werden im gleichen Format auf der Diskette gespeichert. Die unterschiedliche Typenkennzeichnung erfolgt nur im Inhaltsverzeichnis und bewirkt eine unterschiedliche Behandlung beim Laden und Starten.

Daten-Dateien haben eine gänzlich andere Struktur, daher ergeben sich auch Einschränkungen bei der Anwendung einzelner Befehle.

Wollen Sie eine Datei auf der Diskette ansprechen oder neu anlegen, so ist in den Kommandos die Angabe eines Dateinamens erforderlich, der in das Inhaltsverzeichnis der Diskette eingetragen wird.

Ein Dateiname kann maximal acht Zeichen lang sein und aus einer beliebigen Buchstaben-, Zeichen- oder Ziffernfolge bestehen.

In den Kommandos ist der Dateiname immer in Anführungsstrichen anzugeben. Dabei darf auch der abschließende Anführungsstrich im Gegensatz zu anderen BASIC-Befehlen nicht vergessen werden, auch wenn keine weiteren Angaben erfolgen.

Leider erlaubt das LASER-DOS nicht die Verwendung einer String-Variablen anstelle des Dateinamens; dieser ist stets vollständig direkt im Kommando anzugeben. Dies erschwert die flexible Behandlung verschiedener Daten-Dateien. Wie Sie sich dennoch helfen können, ist im Kapitel 5 "Tips zur Programmierung" vermerkt.

Kommando-Übersicht

Die 17 zusätzlichen Diskettenkommandos lassen sich funktionell in drei Gruppen einordnen.

Allgemeine Anweisungen

INIT

Initialisieren einer Diskette.
Mit diesem Kommando wird die Grundstruktur auf die Diskette gebracht, d.h. sie wird in Spuren und Sektoren unterteilt.

DRIVE n

Laufwerks-Auswahl.
Hiermit können Sie eines der zwei anschließbaren Laufwerke für die weitere Verarbeitung auswählen.

DCOPY

Disketten kopieren.
Mit diesem Kommando kopieren Sie den Inhalt einer Diskette auf eine andere.

STATUS

Disketten-Status ausgeben.
Mit "STATUS" können Sie sich den noch verfügbaren Platz auf der Diskette ausgeben lassen.
(erst ab DISK BASIC V 1.2)

Datei-Verwaltungsfunktionen

DIR

Ausgabe des Inhaltsverzeichnisses.
Alle auf der Diskette gespeicherten Programme und Dateien werden auf dem Bildschirm aufgelistet.

SAVE "name"

Sichern eines BASIC-Programms.
Ein im Speicher befindliches BASIC-Programm wird mit dem Dateinamen "name" auf die Diskette geschrieben.

LOAD "name"

Laden eines BASIC-Programms.
Das mit "name" bezeichnete BASIC-Programm wird von der Diskette eingelesen.

RUN "name"

Laden und Starten eines BASIC-Programms.
Das mit "name" bezeichnete BASIC-Programm wird von der Diskette eingelesen und sofort gestartet.

BSAVE "name",aaaa,eeee

Sichern eines Maschinenprogramms
Ein im Speicher befindliches Maschinenprogramm wird mit dem Dateinamen "name" auf die Diskette geschrieben.

BLOAD "name"

Laden eines Maschinenprogramms.
Das mit "name" angegebene Maschinenprogramm wird von der Diskette eingelesen.

BRUN "name"

Laden und Starten eines Maschinenprogramms.
Das mit "name" angegebene Maschinenprogramm wird von der Diskette eingelesen und gestartet.

REN "name1","name2"

Umbenennen einer Datei.
Die mit "name1" bezeichnete Datei wird auf der Diskette in "name2" umbenannt.

ERA "name"

Löschen einer Datei.
Die mit "name" bezeichnete Datei wird auf der Diskette gelöscht.

DCOPY "name"

Kopieren eines Programms.
Das mit "name" bezeichnete BASIC- oder Maschinenprogramm wird auf eine andere Diskette kopiert.

Speichern und Verarbeiten von Daten

OPEN "name",n

öffnen einer Daten-Datei.
Die mit "name" bezeichnete Daten-Datei wird zum Schreiben oder Lesen geöffnet.

PR# "name",var1[var2...,varn]

Schreiben in eine Daten-Datei.
Die im Befehl angegebenen Variablen werden zu einem Datensatz zusammengefaßt und in die Daten-Datei "name" geschrieben.

IN# "name",var1[var2...,varn] Lesen aus einer Daten-Datei.
Aus der Datei "name" wird ein Datensatz gelesen und in die angegebenen Variablen übertragen.

CLOSE "name" Schließen einer Daten-Datei.
Die mit "name" bezeichnete Daten-Datei wird geschlossen.

Allgemeine Anweisungen

INIT

Vorbereiten einer Diskette

Syntax: **INIT**

als Direkt-Kommando und im Programm-Modus zugelassen.

Mit dem INIT-Kommando wird eine Diskette für die Speicherung von Programmen oder Daten vorbereitet; sie wird "initialisiert".

Dies bedeutet, daß die im Abschnitt "Aufzeichnungsstruktur dargestellte Grundstruktur auf der Diskette hergestellt wird.

Es werden 40 Spuren mit jeweils 16 Sektoren eingerichtet und alle Adreß- und Datenmarken geschrieben.

Nach dem Schreiben wird jeder Sektor einzeln adressiert und kontrollgelesen.

Der gesamte Initialisierungsvorgang nimmt ca. 2-3 Minuten in Anspruch.

Nach korrekter Durchführung meldet sich BASIC mit **READY**, und es kann das nächste Kommando eingegeben werden.

Der Initialisierungsvorgang kann jederzeit durch Betätigen der BREAK-Taste abgebrochen werden.

Achtung:

Mit dem INIT-Kommando wird eine nichtschreibgeschützte Diskette ohne jede weitere Prüfung überschrieben, d.h. evtl darauf befindliche Daten gehen verloren.

Mögliche Fehler:

?DISK WRITE PROTECTED	Die Schreibschutzkerbe der Diskette ist überklebt.
?DISK I/O ERROR	Beim Kontrollesen trat ein Fehler auf. (fehlerhafte Diskette oder schlechte Zentrierung - siehe Einlegen)

DRIVE

Laufwerks - Auswahl

Syntax: **DRIVE n**

n = Nummer des Laufwerks (1 o. 2)

als Direkt-Kommando und im Programm-Modus zugelassen.

Das DRIVE - Kommando dient zur Auswahl eines der zwei anschließbaren Laufwerke.

Nach Einschalten des Rechners und nach jedem Kopierkommando (DCOPY) ist automatisch immer Laufwerk 1 ausgewählt.

Wollen Sie auf Laufwerk 2 zugreifen, so muß zunächst mit **DRIVE 2** darauf umgeschaltet werden.

Alle DOS-Kommandos, außer DCOPY, werden auf dem ausgewählten Laufwerk ausgeführt. Achten Sie daher darauf, daß Sie immer das richtige Laufwerk ausgewählt haben. Ein 'INIT'-Kommando z.B. auf das falsche Laufwerk führt unweigerlich zur Zerstörung einer dort zufällig befindlichen Diskette mit wichtigen Daten.

Sind Sie nicht sicher, welches Laufwerk zur Zeit ausgewählt ist, so führen Sie vorsichtshalber ein entsprechendes DRIVE-Kommando (DRIVE 1 oder DRIVE 2) aus.

Das DRIVE - Kommando ändert nur die DOS-internen Zeiger, ein Diskettenzugriff findet nicht statt.

Mögliche Fehler

?FUNCTION CODE ERROR	falsche Laufwerks-Auswahl (nicht 1 oder 2)

DCOPY

Diskette kopieren

Syntax: **DCOPY**

nur als Direkt-Kommando zulässig.

Das DCOPY-Kommando ohne weitere Parameterangabe führt zur vollständigen Kopie einer Diskette auf eine zweite initialisierte Diskette.

Das Kopieren ist mit einem oder zwei Laufwerken möglich. Bei einem Laufwerk müssen Sie allerdings während des Kopiervorgangs die Disketten mehrfach wechseln.

Nach Eingabe des Kommandos werden Sie zunächst zur Auswahl des Ausgangs- und Ziellaufwerks aufgefordert.

SOURCE DISK (1/2)? (Source = Quelle)

DESTINATION DISK (1/2)? (Destination = Ziel)

Beantworten Sie jede dieser Fragen durch Betätigen der "1" oder "2" Taste.

Besitzen Sie nur ein Laufwerk, so antworten Sie auf jede Frage mit "1".

Mit CTRL/BREAK kann die Kommandoausführung abgebrochen werden.

Nach der Laufwerksauswahl beginnt der Kopiervorgang. Dazu wird der gesamte RAM-Speicher benutzt, um so wenig wie möglich zwischen Ausgangs- und Zielaufwerk umschalten zu müssen.

Kopieren Sie von einem Laufwerk zu einem zweiten, so läuft der gesamte Kopiervorgang automatisch ab. Bei nur einem Laufwerk (von 1 nach 1 bzw. von 2 nach 2) erhalten Sie vor jedem Lese- oder Schreibvorgang Gelegenheit, die jeweils richtige Diskette einzulegen.

INSERT SOURCE DISKETTE
(PRESS SPACE WHEN READY)

vor jedem Lesen von der Ausgangsdiskette, bzw.

INSERT DESTINATION DISKETTE
(PRESS SPACE WHEN READY)

vor jedem Schreiben auf die Zieldiskette.

Den Kopiervorgang können Sie jederzeit durch Betätigen der BREAK-Taste abbrechen.

Der Abschluß des Kopiervorgangs wird mit **READY** angezeigt.

Achtung

- Beachten Sie, daß die Zieldiskette zuvor initialisiert sein muß.
- Auf der Zieldiskette befindliche Daten werde überschrieben (auf richtige Laufwerks- und Diskettenwahl achten)
- Der gesamte verfügbare RAM-Bereich wird von DCOPY überschrieben, d.h. dort befindliche Daten oder Programme müssen Sie vorher sichern oder anschließend wieder laden.
- Beim Einsatz von "Extended BASIC" ist anschließend eine Neuinitialisierung des Rechners erforderlich (Aus-/Einschalten).
- Nach Abschluß ist, unabhängig von einem vorherigen DRIVE-Kommando, immer Laufwerk 1 ausgewählt.

Mögliche Fehler

?ILLEGAL DIRECT	Es wurde versucht, das DCOPY-Kommando aus einem Programm heraus aufzurufen.
?DISK WRITE PROTECTED	Die Schreibschutzkerbe der Zieldiskette ist überklebt.
?DISK I/O ERROR	Schreib- oder Lesefehler auf einer der beiden Disketten. (defekt oder schlechte Zentrierung)

Anmerkung

Dies ist eines der wichtigsten Kommandos des DOS.

Wie schon eingangs erwähnt, ist keine Diskette auf Dauer ein zuverlässiger Datenspeicher (Abrieb).

Fertigen Sie also von jeder Diskette, die für Sie wichtige Programme und Daten enthält, eine Kopie

- nach der Ersterstellung bzw. dem Erwerb
- nach jeder wesentlichen Änderung des Inhalts.

STATUS

Ausgabe des Disketten-Status

(erst ab DISK BASIC V 1.2)

Syntax: **STATUS**

Als Direktkommando und im Programm-Modus zugelassen.

Vom STATUS-Kommando wird der noch verfügbare Platz auf der Diskette ermittelt und ausgegeben.

Die Ausgabe erfolgt in zwei Formen. In der ersten Zeile wird die Anzahl der noch freien Sektoren in der Form

nn RECORDS FREE

ausgegeben.

In der zweiten Zeile erfolgt die Angabe der freien Bytes in der Form

nn.nnn K BYTES FREE

Beispiel:

```
STATUS
80 RECORDS FREE
10.0 K BYTES FREE
```

Mögliche Fehler:

?DISK I/O ERROR	Die Belegungs-übersicht der Diskette konnte nicht fehlerfrei gelesen werden.

Datei - Verwaltungsfunktionen

DIR

Ausgabe des Inhaltsverzeichnisses

Syntax: **DIR**

Als Direkt-Kommando und im Programm-Modus zugelassen.

Vom DIR-Kommando wird auf dem Bildschirm ein Verzeichnis aller auf der Diskette gespeicherten Programme und Dateien ausgegeben.

Die Auflistung beinhaltet Dateityp und Dateinamen.

Mögliche Dateitypen:

T	=	BASIC - Programm	(Textdatei)
B	=	Maschinen-Programm	(Binärdatei)
D	=	Daten - Datei	

Beispiel:

```
DIR
B:SCHACH
B:KALAH
T:AB.LAND
T:ANSCHR
D:KARTEI
READY
```

Die Diskette enthält

- zwei Maschinen-Programme, SCHACH und KALAH,
- zwei BASIC-Programme, AB.LAND und ANSCHR und
- eine Daten-Datei mit Namen KARTEI.

Die Auflistung kann durch Betätigen der Leertaste (SPACE) angehalten und mit derselben Taste wieder fortgesetzt werden.

Mögliche Fehler:

?DISK I/O ERROR	Das Inhaltsverzeichnis der Diskette konnte nicht einwandfrei gelesen werden.

SAVE

Sichern eines BASIC-Programms auf Diskette

Syntax: **SAVE "name"**

"name" = maximal 8-stelliger Programmname,
in Anführungszeichen eingeschlossen.

Als Direktkommando und im Programm-Modus zugelassen.

Ein im Speicher befindliches BASIC-Programm wird unter dem Dateinamen "name" auf der Diskette gespeichert.

Das Programm erhält die Typkennzeichnung "T" (Textdatei).

Im Direktmodus wird der Abschluß des Speichervorgangs mit **READY** angezeigt.

Im Programm-Modus wird das Programm mit dem auf "SAVE" folgenden Befehl fortgesetzt.

Beispiel:

SAVE "KARTEI"

überträgt ein im Speicher befindliches BASIC-Programm unter dem Namen "KARTEI" auf die Diskette.

Mögliche Fehler:

?SYNTAX ERROR	- kein Dateiname angegeben - Dateiname nicht in Anführungszeichen - Nach dem Dateinamen kein Zeilenende (RETURN) oder Kommandotrenner ":".
?DISK WRITE PROTECTED	Die Schreibschutzkerbe der Diskette ist überklebt.
?FILE ALREADY EXISTS	Eine Datei gleichen Namens ist bereits auf der Diskette vorhanden.
?DIRECTORY FULL	Im Inhaltsverzeichnis ist kein Platz mehr vorhanden (maximal 120 Einträge).

?DISK FULL	Auf der Diskette ist keine ausreichende Anzahl freier Sektoren für das Programm vorhanden.
?DISK I/O ERROR	Beim Schreiben oder Lesen auf der Diskette trat ein Fehler auf.

Der Schreibvorgang kann jederzeit durch Betätigen der BREAK-Taste abgebrochen werden. Abhängig vom Zeitpunkt der Tastenbetätigung wird jedoch nicht immer der Eintrag im Inhaltsverzeichnis gelöscht (Fehler im DOS).
Um eine einwandfreie Diskettenverwaltung sicherzustellen, sollten Sie daher in einem solchen Fall das Inhaltsverzeichnis mit DIR prüfen und ggf. die Datei mit **ERA** per Hand löschen.

LOAD

Laden eines BASIC-Programms von Diskette

Syntax: **LOAD "name"**

"name" = maximal 8-stelliger Programmname,
in Anführungszeichen eingeschlossen.

Als Direktkommando und im Programm-Modus zugelassen.

Ein unter dem Dateinamen "name" auf der Diskette gespeichertes BASIC-Programm wird in den Speicher geladen.

Der Abschluß des Ladevorgangs wird mit **READY** angezeigt.

Beispiel:

```
LOAD "KFZ"
```

überträgt das BASIC-Programm KFZ von der Diskette in den Speicher.

Sie können sich ein so geladenes BASIC-Programm anschließend mit LIST anschauen und ggf. modifizieren.

Achtung:

Vor dem Zurückschreiben eines modifizierten Programmes auf die Diskette müssen Sie entweder zuvor das dort befindliche Programm mit "ERA" löschen oder dem geänderten Programm einen anderen Namen geben.

Beispiel:

```
LOAD "XYZ"
>READY
LIST
 .
 .    modifizieren
 .
ERA "XYZ"
>READY
SAVE "XYZ"
```

Nach dem Einlesen des Programms wird auf jeden Fall in den Direkt-Modus (BASIC-Warmstart) verzweigt, unabhängig davon, ob der Aufruf direkt oder aus einem Programm heraus erfolgte.

Der Lesevorgang kann jederzeit durch Betätigen der BREAK-Taste abgebrochen werden.

Mögliche Fehler:

?SYNTAX ERROR	- kein Dateiname angegeben - Dateiname nicht in Anführungsstrichen eingeschlossen - Nach "name" kein Zeilenende (RETURN) oder Kommandotrenner ":".
?FILE NOT FOUND	- Auf der Diskette konnte kein Programm mit dem angegebenen Namen gefunden werden.

?FILE TYPE MISMATCH - Auf der Diskette wurde zwar eine Datei gleichen Namens gefundenn, dies ist jedoch kein BASIC-Programm (Dateityp = T).

?DISK I/O ERROR - beim Lesen von der Diskette trat ein Fehler auf.
(Diskette fehlerhaft oder Zentrierungs-problem)

RUN

Laden und Starten eines BASIC-Programms

Syntax: **RUN "name"**

"name" = maximal 8-stelliger Programmname, in Anführungsstriche eingeschlossen.

Als Direktkommando und im Programm-Modus zugelassen.

Ein unter "name" auf der Diskette gespeichertes BASIC-Programm wird in den Speicher geladen und ausgeführt.

Beispiel:

```
RUN "GRAFIK"
```

Das BASIC-Programm "GRAFIK" wird geladen und ausgeführt.

Mögliche Fehler:

?SYNTAX ERROR
- kein Dateiname angegeben
- Dateiname nicht in Anführungsstrichen eingeschlossen
- Nach "name" kein Zeilenende (RETURN) oder Kommandotrenner ":".

?FILE NOT FOUND - Auf der Diskette konnte kein Programm mit dem angegebenen Namen gefunden werden.

?FILE TYPE MISMATCH - Auf der Diskette wurde zwar eine Datei gleichen Namens gefunden, dies ist jedoch kein BASIC-Programm (Dateityp = T).

?DISK I/O ERROR - beim Lesen von der Diskette trat ein Fehler auf.
(Diskette fehlerhaft oder Zentrierungs-problem)

BSAVE

Sichern eines Maschinenprogramms auf Diskette

Syntax: **BSAVE "name",aaaa,eeee**

"name" = maximal 8-stelliger Programmname, in Anführungszeichen eingeschlossen.

aaaa = Programm-Anfangsadresse, 4-stellig, in hexadezimaler Schreibweise.

eeee = Programm-Endadresse, 4-stellig, in hexadezimaler Schreibweise.

Als Direktkommando und im Programm-Modus zugelassen.

Ein im Speicher befindliches Maschinenprogramm wird von der Adresse "aaaa" bis zur Adresse "eeee" unter dem Dateinamen "name" auf die Diskette geschrieben.

Es erhält im Inhaltsverzeichnis die Typkennzeichnung "B" (Binär-Datei).

Im Direktmodus wird der Abschluß des Speichervorgangs mit **READY** angezeigt. Im Programmodus wird das Programm mit dem auf BSAVE folgenden Kommando fortgesetzt.

Anstelle eines Maschinenprogramms kann mit diesem Kommando auch jeder beliebige Speicherbereich auf die Diskette übertragen und anschließend mit BLOAD wieder geladen werden.

Lediglich BRUN verlangt ein ablauffähiges Maschinenprogramm, da dieses nach dem Laden sofort gestartet wird.

Beispiel:

```
BSAVE "BOWLING",8000,94FF
```

Das Maschinenprogramm "BOWLING" wird von Adresse 8000H bis zur Adresse 94FFH auf die Diskette übertragen.

Mögliche Fehler:

?SYNTAX ERROR	- kein Dateiname angegeben - Dateiname nicht in Anführungszeichen - Start- und/oder Endadresse fehlen - Start- oder Endadresse nicht 4-stellig hexadezimal (0 - F) - Parameter nicht durch Komma getrennt.
?DISK WRITE PROTECTED	Die Schreibschutzkerbe der Diskette ist überklebt.
?FILE ALREADY EXISTS	Eine Datei gleichen Namens ist bereits auf der Diskette vorhanden.
?DIRECTORY FULL	Im Inhaltsverzeichnis ist kein Platz mehr vorhanden (maximal 120 Einträge).
?DISK FULL	Auf der Diskette ist keine ausreichende Anzahl freier Sektoren für das Programm vorhanden.
?DISK I/O ERROR	Beim Schreiben oder Lesen auf der Diskette trat ein Fehler auf.

Der Schreibvorgang kann jederzeit durch Betätigen der BREAK-Taste abgebrochen werden. Abhängig vom Zeitpunkt der Tastenbetätigung wird jedoch nicht immer der Eintrag im Inhaltsverzeichnis gelöscht (Fehler im DOS).
Um eine einwandfreie Diskettenverwaltung sicherzustellen, sollten Sie daher in einem solchen Fall das Inhaltsverzeichnis mit DIR prüfen und ggf. die Datei mit **ERA** per Hand löschen.

BLOAD

Laden eines Maschinenprogramms von Diskette

Syntax: **BLOAD "name"**

"name" = maximal 8-stelliger Programmname,
in Anführungsstriche eingeschlossen.

Als Direktkommando und im Programm-Modus zugelassen.

Ein unter dem Dateinamen "name" auf der Diskette gespeichertes Maschinenprogramm wird in den Speicher geladen.

Bei einem Direktkommando wird das Ende des Ladevorgangs mit **READY** angezeigt, im Programmodus wird das Programm mit dem auf BLOAD folgenden Befehl fortgesetzt.

Beispiel:

```
BLOAD "UPR01"
```

Maschinenprogramm UPR01 wird von der Diskette geladen.

Das Kommando eignet sich vor allem dazu, mit BSAVE gespeicherte Maschinenprogramm-Routinen aus einem BASIC-Programm heraus zu laden und über USR als Subroutine aufzurufen.

Beispiel:

```
.
.
220 BLOAD "UPRO1": 'UNTERPROGRAMM LADEN
230 POKE  30862,0: 'LSB STARTADRESSE = 00
240 POKE  30863,176:      'MSB STARTADRESSE = B0
250 A = USR(0):           'UNTERROUTINE AUFRUFEN
.
.
```

Das Unterprogramm UPRO1 soll von Diskette geladen und bei
Adresse B000H aufgerufen werden.

Mögliche Fehler:

?SYNTAX ERROR
- kein Dateiname angegeben
- Dateiname nicht in Anführungsstrichen eingeschlossen
- Nach "name" kein Zeilenende (RETURN) oder Kommandotrenner ":".

?FILE NOT FOUND
- Auf der Diskette konnte kein Programm mit dem angegebenen Namen gefunden werden.

?FILE TYPE MISMATCH
- Auf der Diskette wurde zwar eine Datei gleichen Namens gefundenn, dies ist jedoch kein Maschinenprogramm (Dateityp = B).

?DISK I/O ERROR
- beim Lesen von der Diskette trat ein Fehler auf.
  (Diskette fehlerhaft oder Zentrierungs-problem)

BRUN

Laden und Starten eines Maschinenprogramms

Syntax: **BRUN "name"**

"name" = maximal 8-stelliger Programmname,
in Anführungszeichen eingeschlossen.

Als Direktkommando und im Programm-Modus zugelassen.

Ein unter dem Dateinamen "name" auf der Diskette gespeichertes Maschinenprogramm wird in den Speicher geladen und ausgeführt.

Der Programmstart erfolgt ausschließlich an der Programm-Anfangsadresse (siehe BSAVE).

Beispiel:

BRUN "FIFFI"

Das Maschinenprogramm "FIFFI" wird geladen und gestartet.

Mögliche Fehler:

?SYNTAX ERROR	- kein Dateiname angegeben - Dateiname nicht in Anführungsstrichen eingeschlossen - Nach "name" kein Zeilenende (RETURN) oder Kommandotrenner ":".
?FILE NOT FOUND	- Auf der Diskette konnte kein Programm mit dem angegebenen Namen gefunden werden.
?FILE TYPE MISMATCH	- Auf der Diskette wurde zwar eine Datei gleichen Namens gefundenn, dies ist jedoch kein Maschinenprogramm (Dateityp = B).

?DISK I/O ERROR - beim Lesen von der Diskette trat ein Fehler auf.
(Diskette fehlerhaft oder Zentrierungs-problem)

RENAME

Umbenennen von Dateien und Programmen

Syntax: **REN "name1","name2"**

"name1" = Datei-/Programmname, alt, max. 8-stellig, in Anführungsstriche eingeschlossen.

"name2" = Datei-/Programmname, neu, max. 8-stellig, in Anführungsstriche eingeschlossen.

Als Direktkommando und im Programm-Modus zugelassen.

Ein auf der Diskette unter dem Namen "name1" befindliches Programm oder eine Datei wird in "name2" umbenannt.

Beispiel:

REN "OTTO","ANTON"

Die Datei "OTTO" wird in "ANTON" umbenannt.

Mögliche Fehler:

?SYNTAX ERROR
- "name1" und/oder "name2" fehlen.
- "name1" oder "name2" nicht in Anführungsstrichen
- Namen nicht durch Komma getrennt.

?DISK WRITE PROTECTED	Die Schreibschutzkerbe der Diskette ist überklebt.
?FILE NOT FOUND	Die mit "name1" bezeichnete Datei ist nicht auf der Diskette vorhanden.
?FILE ALREADY EXISTS	Die mit "name2" bezeichnete Datei ist bereits auf der Diskette vorhanden.
?DISK I/O ERROR	Beim Lesen oder Schreiben des Inhaltsverzeichnisses trat ein Fehler auf.

DCOPY

Kopieren eines Programms

Syntax: **DCOPY "name"**

"name" = maximal 8-stelliger Programmname,
in Anführungszeichen eingeschlossen.

Nur als Direkt-Kommando zulässig.

Das DCOPY-Kommando mit Angabe eines Programmnamens bewirkt ein Kopieren dieses Programms von einer Diskette auf eine andere.

Nach Eingabe des Kommandos werden Sie zunächst zur Angabe des Ausgangs- und Ziellaufwerks aufgefordert.

SOURCE DISK (1/2)?
DESTINATION DISK (1/2)?

Beantworten Sie jede dieser beiden Fragen durch Betätigen der "1"- oder "2"-Taste.

Besitzen Sie nur ein Laufwerk, so antworten Sie auf jede Frage mit "1".

Mit CTRL/BREAK können Sie die Kommandoausführung abbrechen.

Nach der Laufwerksauswahl beginnt der Kopiervorgang. Das Kopieren erfolgt durch Aufruf der LOAD- und SAVE-Routinen, wie sie auch bei LOAD und BLOAD, bzw. bei SAVE und BSAVE verwendet werden.

Aus diesem Grunde läßt sich mit dem DCOPY-Kommando auch keine einzelne Daten-Datei (Dateityp = D) kopieren, da diese anders strukturiert ist.

Kopieren Sie auf nur einem Laufwerk (SOURCE DISK = DESTINATION DISK), so wird vor dem Ladevorgang die Aufforderung

INSERT SOURCE DISKETTE
(PRESS SPACE WHEN READY)

und vor dem Schreibvorgang die Aufforderung

INSERT DESTINATION DISKETTE
(PRESS SPACE WHEN READY)

ausgegeben (SOURCE = Quelle, DESTINATION = Ziel).

Haben Sie die jeweils richtige Diskette eingelegt, so betätigen Sie die Leertaste zur Fortführung der Funktion.

Den Kopiervorgang können Sie jederzeit mit der BREAK-Taste abbrechen. Tun Sie dies während des Schreibvorgangs, so beachten Sie bitte die Hinweise bei SAVE und BSAVE.

Ist das Kopieren beendet, erscheint die Meldung **READY.**

Beispiel: (mit '>' sind Systemausgaben gekennzeichnet)

```
>READY
DCOPY "EMIL"
>SOURCE DISK (1/2)?
1
>DESTINATION DISK (1/2)?
1
>INSERT SOURCE DISKETTE
>(PRESS SPACE WHEN READY)
Leertaste
.
.  Ladevorgang
```

```
.
>INSERT DESTINATION DISKETTE
>(PRESS SPACE WHEN READY)
Leertaste
.
.  Schreibvorgang
.
>READY
```

Das zu kopierende Programm überschreibt seinen originären Speicherbereich im RAM.

Nach Abschluß des Kopierens ist, unabhängig von einem vorherigen DRIVE-Kommando, immer Laufwerk 1 ausgewählt.

Mögliche Fehler:

?ILLEGAL DIRECT	Es wurde versucht, das DCOPY-Kommando aus einem Programm heraus aufzurufen.
?SYNTAX ERROR	"name1" nicht in Anführungsstriche eingeschlossen.
?FILE NOT FOUND	Auf der Ausgangs-Diskette ist das Programm "name1" nicht vorhanden.
?FILE TYPE MISMATCH	Es wurde versucht, eine Daten-Datei zu kopieren.
?DISK WRITE PROTECTED	Die Schreibschutzkerbe der Zieldiskette ist überklebt.
?FILE ALREADY EXISTS	Auf der Zieldiskette ist bereits ein Programm mit dem Namen "name1" vorhanden.
?DIRECTORY FULL	Das Inhaltsverzeichnis der Zieldiskette ist voll. Das Programm kann nicht mehr eingetragen werden (max. 120 Dateien/Programme).
?DISK FULL	Auf der Zieldiskette ist kein Platz mehr vorhanden.

?DISK I/O ERROR	Schreib- oder Lesefehler auf einer der beiden Disketten.

ERASE

Datei oder Programm auf der Diskette löschen.

Syntax: **ERA "name"**

"name" = maximal 8-stelliger Programm- oder Dateiname, in Anführungszeichen.

Als Direktkommando und im Programm-Modus zugelassen.

Ein mit "name" bezeichnetes Programm oder eine Daten-Datei wird auf der Diskette geloscht.

Dazu wird der Eintrag im Inhaltsverzeichnis gestrichen und es werden alle von dieser Datei belegten Sektoren freigegeben.

Beispiel:

```
ERA "DAT1"
```

Die Datei mit dem Namen "DAT1" wird gelöscht.

Mögliche Fehler:

?SYNTAX ERROR	- kein Dateiname angegeben - Dateiname nicht in Anführungsstrichen
?DISK WRITE PROTECTED	Die Schreibschutzkerbe der Diskette ist überklebt.
?FILE NOT FOUND	Die angegebene Datei ist nicht auf der Diskette enthalten.
?DISK I/O ERROR	Schreib- oder Lesefehler auf der Diskette.

Speichern und Verarbeiten von Daten

Dateiorganisation und -zugriff

Das LASER-DOS erlaubt es Ihnen, aus einem BASIC-Programm heraus Daten auf der Diskette zu speichern und diese anschließend wieder zu verarbeiten.

Diese Daten werden in speziellen Daten-Dateien mit dem Typkennzeichen "D" abgelegt.

Die dabei angebotene Speicherform ist "sequentiell". Sequentiell bedeutet, daß die Daten in der Datei, wie bei einer Kassette, hintereinander gespeichert sind. Das Lesen oder Schreiben von Daten beginnt stets am Dateianfang, weitere Lese- und Schreibaufrufe greifen auf die nachfolgenden Positionen der Datei zu.

Im Gegensatz dazu steht die "direkte" Zugriffsart (random access), mit der auf beliebige Daten einer Datei direkt zugegriffen werden kann. Doch leider unterstützt das LASER-DOS diese Speicherform standardmäßig nicht. Sie läßt sich aber mit etwas Assembler-/Maschinensprache-Kenntnissen und den im letzten Kapitel beschriebenen Hilfsroutinen leicht nachbilden.

Sequentieller Zugriff ist datenfluß-orientiert, d.h. die Anzahl der Zeichen für einen Schreib- oder Lesevorgang kann variieren. Man spricht hier auch von Datensätzen variabler Länge, wobei ein Datensatz die Summe der Datenelemente ist, die mit einem Schreib- oder Leseaufruf auf die Diskette geschrieben oder von dieser gelesen werden.

Sequentielle Dateien stellen die einfachste Form der Datenspeicherung und Wiedergewinnung dar. Sie sind ideal, um unformatierte Daten zu speichern, ohne viel Platz zwischen den einzelnen Datenelementen zu verschenken. Daten werden in derselben Reihenfolge wieder gelesen, wie sie geschrieben wurden.

Um auf eine Daten-Datei zugreifen zu können, muß sie zuvor geöffnet werden. Dazu steht ein spezieller OPEN-Aufruf zur Verfügung. Mit dem öffnen spezifiziert man auch die Zugriffsart, ob Daten geschrieben oder gelesen werden sollen.

Nach Beendigung der Datenmanipulationen sollte jede Daten-Datei mit einem CLOSE-Aufruf geschlossen werden.

Bei der Bearbeitung von Daten-Dateien sollten Sie einige wichtige Punkte beachten:

- Wird eine nicht vorhandene Datei zum Schreiben geöffnet, so wird sie neu angelegt und es wird auf den Dateianfang positioniert.

- Wird eine bestehende Datei zum Schreiben geöffnet, so wird an das Dateiende positioniert, d.h. die Datei wird bei den folgenden Schreibaufrufen erweitert.

- Wollen Sie eine bestehende Datei von Anfang an neu beschreiben, so müssen Sie diese zuvor löschen.

- Das Lesen beginnt nach dem öffnen immer am Dateianfang. Suchen Sie Daten innerhalb einer Datei, so müssen Sie die davor liegenden überlesen.

- Um eine sequentielle Datei zu aktualisieren, lesen Sie die Ausgangsdatei ein und schreiben Sie die aktualisierten Daten in eine neue Datei.

- Beim Lesen der Datei muß die genaue Struktur des zu lesenden Datensatzes bekannt sein. Das bezieht sich nicht auf die Länge des Satzes und der einzelnen Elemente; Sie müssen aber die Anzahl der Elemente und das Format jedes einzelnen Elements (String, Integer usw.) kennen und für jedes Element ein entsprechendes Empfangsfeld bereitstellen.
- Innerhalb der Datei werden die Daten ausschließlich im ASCII-Format abgelegt. Die einzelnen Elemente werden durch Komma getrennt.
  Z.B. die Zahl 1.2345 beansprucht 8 Bytes Speicherplatz, einschließlich eines Leerzeichens für das Vorzeichen am Anfang und eines weiteren Leerzeichens am Ende.
  Der Text "ROBERT MAIER" beansprucht 12 Bytes auf der Diskette.

- Ein Datensatz sollte die Länge von 200 Bytes nicht überschreiten, da ansonsten beim Lesen Schwierigkeiten mit der internen Datenstruktur des BASIC auftreten.

  Zu den 200 Bytes zählen
  - die einzelnen Zeichen
  - die Kommas als Elemententrenner
  - bei Zahlen die Vorzeichenstelle und ein zusätzliches Leerzeichen
  - ein abschließendes RETURN (CR) am Satzende

- Es können maximal zwei Dateien gleichzeitig geöffnet werden, wobei die Zugriffsarten gleich oder gemischt sein können.

Achtung
Sollte sich Ihr System mit DISK-BASIC V1.0 melden, so arbeiten Sie vorsichtshalber nur mit einer Datei zu einer Zeit. Die Verwaltung zweier geöffneter Dateien ist dort noch fehlerhaft und kann zu erheblichen Datenverlusten führen.

- Das DOS sagt Ihnen nicht, wann beim Lesen das Dateiende erreicht ist. Sie müssen dies selbst festlegen, indem Sie z.B. eine bestimmte Endekennung als letztes in die Datei schreiben.

Beispiel für sequentielle Ausgabe:

Es soll eine Tabelle mit Daten zur Umrechnung englischer in metrische Maßeinheiten gespeichert werden.

Englische Einheit	Metrische Einheit
1 Inch	2.54001 cm
1 Mile	1.60935 km
1 Acre	4046.86 qm
1 Cubic Inch	0.01639 ltr
1 U.S. Gallon	3.785 ltr
1 Liquid Quart	0.9463 ltr
1 lbs	0.45359 kg

Die Daten sollen wie folgt auf der Diskette strukturiert und in eine Daten-Datei namens "ENG>MET" eingetragen werden:

"Engl.Einheit -> Metr.Einheit" , Umrechnungsfaktor

z.B. "IN->CM",2.54001

Das folgende Programm erzeugt eine solche Datei.

```
10  OPEN "ENG>MET",1
20  FOR I% = 1 TO 7
30  READ E$,F
40  PR# "ENG>MET",E$,F
50  NEXT
60  CLOSE "ENG>MET"
70  DATA "IN->CM",2.54001,"MI->KM",1.60935,"ACRE->QKM",4046.86E-6
```

```
80  DATA "CU.IN->LTR",1.638716E-2,"GAL->LTR",3.785
90  DATA "LIQ.QT->LTR",0.9463,"LB->KG",0.45359
100 END
```

Zeile 10 erzeugt die Datei "ENG>MET" und öffnet diese zum Schreiben.
In Zeile 40 wird jeweils ein Datensatz in die Datei geschrieben.
Zeile 60 schließt die Datei "ENG>MET" wieder.

Beispiel für sequentielle Eingabe:

Das folgende Programm liest die Datei "ENG>MET" in zwei parallele Matrizen ein und fragt anschließend nach Umrechnungsproblemen.

```
10  CLEAR 1000
20  DIM E$(6),F(6)
30  OPEN "ENG>MET",0
40  FOR I% = 0 TO 6
50  IN# "ENG>MET",E$(I%),F(I%)
60  NEXT
70  CLOSE "ENG>MET"
100 CLS: PRINT " UMRECHNUNG ENGLISCH=>METRISCH"
110 PRINT: FOR I% = 0 TO 6
120 PRINT TAB(4); USING "(## )   %         %   ";I%,E$(I%)
130 NEXT
140 PRINT @320, "WELCHE UMRECHNUNG (0-6)";
150 INPUT W%: IF W% > 6 THEN 190
160 INPUT "ENGLISCHER WERT";V
170 PRINT "DER METRISCHE WERT IST" V*F(W%)
180 INPUT "WEITER MIT <RETURN>";X
190 GOTO 100
```

In Zeile 30 wird die Datei für die Eingabe geöffnet. Das Lesen beginnt am Dateianfang.
In Zeile 50 wird jeweils ein Datensatz mit den Elementen E$ (Einheit) und F (Faktor) gelesen und auf die Matrizen verteilt.
Beachten Sie, daß die Variablenliste beim Einlesen gleich der des Schreibbefehls im vorherigen Programm ist.
In Zeile 70 wird die Datei wieder geschlossen.

Aktualisieren einer Datei

Wollen Sie eine bestehende Datei um ein oder mehrere Sätze erweitern, so öffnen Sie diese Datei zum Schreiben und geben einfach mit PR# zusätzliche Datensätze ein, die an den bestehenden Datenbestand angehängt werden.

Wollen Sie Daten innerhalb einer Datei ändern, so empfiehlt sich folgendes Verfahren (nicht mit DISK BASIC V1.0).

1. Zu bearbeitende Datei zum Lesen öffnen.
2. Eine zweite neue Datei zum Schreiben öffnen
3. Lesen eines Satzes und Bearbeiten der Daten
4. Schreiben des Satzes in die neue Datei
5. Wiederholen der Punkte 3 u. 4 bis zum Dateiende
6. Beide Dateien schließen
7. Die Ausgangsdatei löschen
8. Die neue Datei in die Ausgangsdatei umbenennen

Beim DISK BASIC V1.0 bleibt nur die Lösung, die zu bearbeitende Datei komplett in den Speicher zu lesen, zu bearbeiten und komplett in die neue Datei zu schreiben. Das limitiert allerdings die Größe der Datei auf den verfügbaren Speicher.

OPEN

öffnen einer Datei

Syntax: **OPEN "name",n**

"name" = maximal 8-stelliger Dateiname, in Anführungszeichen eingeschlossen.

n = Art des Zugriffs
0 - Lesen
1 - Schreiben

Nur im Programm-Modus zugelassen.

Mit dem OPEN - Kommando wird eine Daten-Datei (Typ = D) zum Schreiben oder zum Lesen geöffnet.

Das OPEN-Kommando legt für jede geöffnete Datei intern einen Datei-Kontrollblock an, der Funktionscodes und Zeiger enthält.

Ferner wird entsprechend dem Zugriffscode auf die Datei positioniert:

- Beim Lesen immer auf den Dateianfang
- Beim Schreiben auf eine neue Datei auf den Dateianfang
- Beim Schreiben auf eine vorhandene Datei auf das Dateiende.

Da nur zwei Datei-Kontrollblöcke im System vorgesehen sind, können gleichzeitig auch nur zwei Dateien geöffnet sein. Die Zugriffsart ist dabei ohne Belang, es können beide zum Schreiben, beide zum Lesen oder eine zum Lesen und die zweite zum Schreiben geöffnet werden (siehe Einschränkung DISK BASIC V1.0 auf den vorhergehenden Seiten).

Beispiel:

```
OPEN "TEST",0
```

Die Daten-Datei "TEST" wird zum Lesen geöffnet.

Eine Daten-Datei kann nur einmal zu einer Zeit geöffnet werden. Der Versuch, dieselbe Datei nochmals zu öffnen, führt zu einer Fehlermeldung.

Da die Datei-Kontrollblöcke (FCB) sich außerhalb der BASIC-Programme befinden, bleibt eine Datei geöffnet, wenn das aufrufende Programm vor dem CLOSE-Aufruf wegen Fehlers oder Betätigen der BREAK-Taste abgebrochen wurde und vielleicht schon garnicht mehr im Speicher ist. Eine solche Datei läßt sich nicht mehr ohne weiteres öffnen.

Sollte der Fall eintreten, daß ein BASIC-Programm abgebrochen wird, ohne seine Dateien sauber zu schließen, so sollten Sie dies per Direktkommando **CLOSE "Dateiname"** nachholen.

Mögliche Fehler:

?ILLEGAL DIRECT	Es wurde versucht, das OPEN-Kommando im Direkt-Modus auszuführen.

?SYNTAX ERROR	- ein oder beide Parameter fehlen - kein Komma als Trenner - "name" nicht in Anführungszeichen - Zugriffsart nicht 0 oder 1
?FILE ALREADY OPEN	Datei ist bereits geöffnet, ggf. mit dem Direktkommando "CLOSE" schließen.
?FILE TYPE MISMATCH	Die im OPEN-Kommando adressierte Datei ist keine Daten-Datei
?FILE NOT FOUND	Eine zum Lesen zu öffnende Datei ist nicht auf der Diskette vorhanden.
?DISK BUFFER FULL	Es sind bereits zwei Dateien geöffnet und es steht kein Datei-Kontrollblock mehr zur Verfügung.
DISK I/O ERROR	Beim Lesen von der Diskette trat ein Fehler auf.

PR#

Schreiben von Datensätzen in eine Datei

Syntax: **PR# "name",Elementenliste**

"name" = maximal 8-stelliger Dateiname, in Anführungszeichen eingeschlossen.

Elementenliste = Liste von Variablen und Werten, die in die Datei geschrieben werden sollen. Die einzelnen Elemente sind durch Komma zu trennen.

Nur im Programm-Modus zugelassen.

Stellt aus den Werten der Elementliste einen Datensatz zusammen und veranlaßt das Schreiben in die Daten-Datei.

Diese muß zuvor mit einem OPEN-Kommando zum Schreiben geöffnet worden sein.

Beispiel:

```
200 A1 = -40.456: B$ = "STRING-WERT"
210 OPEN "TEST",1
220 PR# "TEST",A1,B$,"DAS WAR'S"
230 CLOSE "TEST"
240 END
```

Nach dem öffnen der Datei "TEST" in Zeile 210 wird in Zeile 220 ein Datensatz zusammengestellt und in diese Datei geschrieben.

Der Datensatz enthält die aktuellen Werte von A1 und B$ und zusätzlich die Zeichenfolge "DAS WAR'S". Die Werte können später mit einem IN#-Kommando wieder eingelesen werden.

Dabei ist darauf zu achten, daß die Elementenliste des IN#-Kommandos in Bezug auf Anzahl und Art der Elemente gleich dem PR#-Kommando ist.

Die Werte, die durch die Elementenliste repräsentiert werden, sollten insgesamt nicht länger als 200 Zeichen sein. Dazu zählen neben den Werten selbst auch sämtliche Trennzeichen (Kommas) zwischen den Werten, bei numerischen Werten zusätzlich die Vorzeichenstelle und ein abschließendes Leerzeichen und zuletzt noch die Datensatzendekennung (CR).

Der Datensatz im vorausgegangenen Beispiel hätte eine Länge von 31 Zeichen

```
-40.456 ,STRING-WERT,DAS WAR'S
```

Leider weiß man oft bei der Erstellung der Elementenliste nicht genau, wie groß die einzelnen Variablen zum Zeitpunkt der Speicherung sein werden. Dann hilft nur vorsichtiges Schätzen. Bleiben Sie stets auf der sicheren Seite und teilen Sie im Zweifelsfall Ihre Elementenliste auf mehrere PR#-Kommandos auf.

Das PR#-Kommando merkt es leider nicht, wenn ein Datensatz zu lang wird. Dieser wird einfach in vollständiger Länge auf die Diskette geschrieben. Probleme macht anschließend das Einlesen mit dem IN#-Kommando, wobei im einfachsten Falle "nur" Daten verlorengehen.

Das Schreiben eines Datensatzes bewirkt nicht unbedingt auch ein physikalisches Schreiben in die Datei. Auf die Diskette geschrieben werden immer nur volle Sektoren. Die Daten des PR#-Kommandos werden in einem internen Puffer gesammelt, der die Größe eines Sektors hat. Immer dann, wenn der Puffer voll ist, wird er in einen freien Sektor auf der Diskette übertragen und anschließend ein neuer freier Sektor ermittelt. Dieses Schreiben eines Sektors kann mitten in einem PR#-Kommando erfolgen; es können auch mehrere PR#-Kommandos erforderlich sein, einen Sektor zu füllen.

Ein mit PR# zu schreibender Datensatz wird ohne Rücksicht auf Sektorgrenzen geschrieben. Man bezeichnet den Sektor auch als physikalische Einheit, während ein Datensatz eine logische Einheit darstellt.

Mögliche Fehler:

?ILLEGAL DIRECT	Es wurde versucht, ein PR#-Kommando direkt auszuführen.
?SYNTAX ERROR	- kein Dateiname angegeben - Dateiname nicht in Anführungszeichen - kein Element in der Liste - kein Komma als Trenner
?FILE NOT OPEN	Datei wurde zuvor ncht geöffnet.
?ILLEGAL WRITE	Die Datei wurde zum Lesen geöffnet.
?DISK WRITE PROTECTED	Die Schreibschutzkerbe der Diskette ist überklebt.
?DISK FULL	Auf der Diskette konnte kein freier Sektor mehr ermittelt werden.
?DISK I/O ERROR	Beim Lesen oder Schreiben trat auf der Diskette ein Fehler auf.

Achtung

Tritt einer dieser Fehler auf, so wird das Programm mit der entsprechenden Fehlermeldung beendet. Beachten Sie bitte, daß diese Datei anschließend nicht geschlossen wurde, Sie sollten dies manuell nachholen.

IN#

Einlesen von Datensätzen aus einer Datei

Syntax: **IN# "name",Elementenliste**

"name" = maximal 8-stelliger Dateiname, in Anführungszeichen eingeschlossen.

Elementenliste = Liste von Variablen, die aus der Datei gelesen werden sollen. Die einzelnen Variablen sind durch Komma zu trennen.

Nur im Programm-Modus zugelassen.

IN# liest einen Datensatz aus der angegebenen Datei und weist die Elemente dieses Satzes den angegebenen Variablen zu.

Die Datei muß zuvor durch ein OPEN-Kommando zum Lesen geöffnet worden sein.

Beispiel:

```
200  OPEN "TEST",0
210  IN#  "TEST",X,A$,B$
220  CLOSE "TEST"
 .
 .
```

Dieses Beispiel bezieht sich auf den im Beispiel des PR#-Kommandos kreierten Datensatz in der Datei "TEST. Die dort gespeicherten Daten werden der Reihe nach den Variablen des IN#-Kommandos zugewiesen.

Nach Ausführen der Zeile 210 enthalten die Variablen folgende Werte:

```
 X = -40.456
A$ = "STRING-WERT"
B$ = "DAS WAR'S"
```

Die Elementenliste des IN#-Kommandos muß der des PR#-Kommandos hinsichtlich Anzahl und Typ der Variablen entsprechen. Ebenso ist die Reihenfolge bei unterschiedlichen Typen einzuhalten. Die Namensgebung ist ohne Belang.

Werden mit IN# fortlaufend Sätze aus einer Datei gelesen, so ergibt sich die Schwierigkeit, das Dateiende zur rechten Zeit zu erkennen. Eine spezielle "END OF FILE" - Kennung gibt es beim LASER-DOS nicht.

Es bieten sich verschiedene Lösungsmöglichkeiten an:

- die Anzahl der Sätze ist bekannt, sie werden im auslesenden Programm mit einem Zähler mitgezählt.

- eine zweite kleine Datei enthält den Satzzähler für die Hauptdatei.

- vor jeden richtigen Datensatz wird ein kurzer Kennsatz geschrieben, der nur aus einem einzigen alphanumerischen Zeichen besteht (z.B. PR# "name","A").

  Beim auslesenden Programm wird vor jedem Lesen eines Datensatzes zunächst dieser Kennsatz gelesen (z.B. IN# "name",A$) Ist die empfangende Stringvariable anschließend leer, so ist das Ende der Datei erreicht.

Mögliche Fehler:

?ILLEGAL DIRECT	Der IN#-Befehl wurde als Direkt-Kommando eingegeben.
?SYNTAX ERROR	- kein Dateiname angegeben - Dateiname nicht in Anführungszeichen - kein Element in der Liste - kein Komma als Trenner
?FILE NOT OPEN	Die angegebene Datei ist nicht geöffnet.
?ILLEGAL READ	Die Datei wurde zum Schreiben geöffnet.
?DISK I/O ERROR	Beim Lesen von der Diskette trat ein Fehler auf.

?REDO	Eine der angegebenen Variablen paßt vom Typ her nicht zu den von der Diskette eingelesenen Daten. Das Programm läuft weiter, die Variable bleibt leer.
?EXTRA IGNORED	In der Variablenliste des IN#-Kommandos sind weniger Variable angegeben als Werte im Datensatz vorhanden sind. Die überzähligen Werte werden ignoriert, das Programm läuft weiter.
??	Die Variablenliste enthält mehr Variable als Werte im Datensatz vorhanden sind. Das Programm erwartet jetzt die Eingabe der fehlenden Werte über die Tastatur.

Achtung

Tritt einer dieser Fehler auf (außer REDO, EXTRA IGNORED und ??), so wird das Programm nach Ausgabe der entsprechenden Meldung beendet. Beachten Sie bitte, daß diese Datei nicht geschlossen wurde, Sie sollten dies manuell nachholen.

CLOSE

Schließen einer Daten-Datei

Syntax: **CLOSE "name"**

"name" = maximal 8-stelliger Dateiname, in Anführungszeichen eingeschlossen.

Als Direkt-Kommando und im Programm-Modus zugelassen

Mit dem CLOSE-Kommando wird eine zuvor bearbeitete Daten-Datei geschlossen.

Bei einer zum Lesen geöffneten Datei bzw. inaktiven Datei (d.h. der letzte Dateizugriff erfolgte nicht auf diese Datei) oder im Direkt-Modus, wird lediglich der Datei-Kontrollblock (FCB = File Control Block) wieder freigegeben. Ein Diskettenzugriff findet nicht statt.

Wird das CLOSE-Kommando jedoch im Programm-Modus gegeben und ist die zu schließende Datei zum Schreiben geöffnet und gerade aktiv, so wird zusätzlich noch der letzte im Puffer befindliche Sektor zurück auf die Diskette geschrieben, so daß keine Daten verlorengehen.

Es ist guter Programmiererbrauch, jede geöffnete Datei nach Benutzung auch wieder zu schließen. Unerläßlich ist es jedoch bei Ausgabedateien, es sei denn, Sie nehmen evtl. Datenverluste in Kauf.

Beispiel:

```
CLOSE "MAILBOX"
```

Die Daten-Datei "MAILBOX" wird geschlossen.

Das Schließen und Wiederöffnen einer Datei ist immer auch dann erforderlich, wenn Sie die Zugriffsart wechseln wollen (z.B. von Schreiben auf Lesen).

Ist die zu schließende Datei überhaupt nicht geöffnet, d.h. es ist kein offener Datei-Kontrollblock für diese Datei vorhanden, so wird das CLOSE-Kommando ohne jede Fehlermeldung übergangen. Dies eignet sich vor allem dazu, prophylaktisch alle in einem Programm benutzten Dateien am Ende zu schließen, ohne zu prüfen, welche gerade geöffnet ist.

Mögliche Fehler:

?SYNTAX ERROR	- kein Dateiname angegeben - Dateiname nicht in Anführungszeichen.
?DISK WRITE PROTECTED	Die Schreibschutzkerbe der Diskette ist überklebt.
?DISK I/O ERROR	Beim Schreiben auf die Diskette trat ein Fehler auf.

4. Fehlermeldungen

Zusammengefaßte Auflistung der möglichen DOS-Fehlermeldungen und ihrer vermutlichen Ursachen.

?DIRECTORY FULL	Es wurde versucht, ein Programm oder eine Datei auf der Diskette zu speichern, deren Inhaltsverzeichnis bereits 120 Einträge enthält.
?DISK BUFFER FULL	Es wurde versucht, mit OPEN eine Datei zu öffnen, obwohl bereits zwei Dateien geöffnet sind.
?DISK FULL	Auf der Diskette ist kein freier Sektor mehr vorhanden.
?DISK I/O ERROR	Beim Schreiben oder Lesen trat ein Fehler auf. z.B. Adreßmarke nicht gefunden, Prüfsumme falsch u.a.
?DISK WRITE PROTECTED	Es wurde versucht, auf eine Diskette zu schreiben, deren Schreibschutzkerbe überklebt ist.
?FILE ALREADY EXISTS	Ein auf die Diskette zu speicherndes Programm ist dort bereits enthalten.
?FILE ALREADY OPEN	Es wurde ein OPEN-Aufruf auf eine Datei abgesetzt, die bereits geöffnet ist.
?FILE NOT FOUND	Eine zum Lesen adressierte Datei oder ein zu ladendes Programm ist nicht auf der Diskette vorhanden.
?FILE NOT OPEN	Es wurde versucht, mit IN# oder PR# eine Datei zu bearbeiten, die zuvor nicht geöffnet wurde.

?FILE TYPE MISMATCH	Es wurde versucht, eine Datei falschen Typs zu bearbeiten. LOAD/RUN - Dateityp ungleich "T" BLOAD/BRUN - Dateityp ungleich "B" OPEN - Dateityp ungleich "D" DCOPY - Dateityp = "D"
?ILLEGAL DIRECT	Es wurde versucht, ein DCOPY-Kommando im Programm-Modus anzuwenden, bzw. eines der Kommandos OPEN, IN# oder PR# im Direkt-Modus.
?ILLEGAL READ	Der Befehl IN# wurde für eine Datei abgesetzt, die zum Schreiben geöffnet ist.
?ILLEGAL WRITE	Der Befehl PR# wurde für eine Datei abgesetzt, die zum Lesen geöffnet ist.
?INSUFFICIENT MEMORY FOR DOS	Es wurde versucht, das DOS-System auf einem LASER 110 oder VZ200 ohne Speichererweiterung zu initialisieren.

5. Tips zur Programmierung

1. Wie in der Beschreibung der letzten Abschnitte bereits erwähnt, besteht das Problem, daß bei Abbruch eines Programms durch Fehler oder BREAK-Taste nicht notwendigerweise auch alle Dateien korrekt abgeschlossen werden.

   Ein erneuter Start eines solchen Programms nach Fehlerkorrektur o.ä. führt dann in aller Regel zur Meldung "FILE ALREADY OPEN".

   Man kann jetzt im Direkt-Modus manuell diese Dateien abschließen, wenn ihre Namen bekannt sind.

   Bei neu zu erstellenden Dateien in einem Programm reicht dies aber nicht aus. Solche Dateien müssen anschließend auch noch gelöscht werden, da ansonsten bei einem erneuten OPEN die Datei fortgeschrieben wird und Sie Ihre Daten mehrfach in der Datei haben.

   Für alle diese Fälle empfiehlt sich folgendes Verfahren:

   Bei allen in der Entwicklung befindlichen Programmen definieren Sie am Ende einen Block mit CLOSE-Aufrufen und ggf. auch Lösch-Aufrufen für alle im Programm angesprochenen Dateien. Bei einer o.a. Programm-Unterbrechung rufen Sie dann einfach diese Routine mit **RUN Zeilennummer** auf.

   Beispiel:

   Sie bearbeiten in einem Programm die drei Dateien DAT1, DAT2 zum Lesen, und DAT 3 wird neu erstellt.

```
.
.       eigenes Programm
.
.
4800  END

20000 CLOSE "DAT1"
20010 CLOSE "DAT2"
20020 CLOSE "DAT3"
20030 ERA "DAT3"
20040 END
```

Tritt eine Programm-Unterbrechung auf, so bereinigen Sie mit **RUN 20000** sauber Ihre Dateien und können nach Korrektur Ihr Programm problemlos neu starten.

2. Oft ergibt sich die Notwendigkeit, daß eine Datei bei Start des Programms auf der Diskette vorhanden sein muß (siehe Programmbeispiel "Anschriftenverz."), obwohl sie noch keine Daten enthält.

   Wenden Sie ein ähnliches Verfahren wie oben an, indem Sie am Ende des eigentlichen Programms folgende Zeilen definieren:

```
.
.       eigenes Programm
.
6000 END

10000 OPEN "MAILBOX",1
10010 CLOSE "MAILBOX"
10020 END
```

   Mit **RUN 10000** kreieren Sie eine leere Datei "MAILBOX" auf der Diskette.

3. Ein Engpaß des LASER-DOS ist, daß die Dateinamen in den Kommandos direkt angegeben werden müssen und nicht durch Variable ersetzt werden konnen.

   Wie kann man trotzdem unterschiedliche Dateien in einem Programm bearbeiten?

   Zum Verständnis des nachstehenden Lösungsansatzes sind Kenntnisse der BASIC-Programmstruktur erforderlich.

   Hier die wichtigsten Punkte:

   - Die Anfangsadresse eines BASIC-Programms findet man in den Speicherstellen 78A4H und 78A5H (30884 u. 30885 dezimal).

   - Eine BASIC-Zeile hat folgenden Aufbau:

     2 Bytes - Zeiger auf die nächste Zeile
     2 Bytes - Zeilennummer
     n Bytes - Zeilentext
     1 Byte - Zeilenendekennung (X'00')

Im Zeilentext enthaltene BASIC-Schlüsselworte, außer den DOS-Kommandos, sind im Text als einstellige "TOKENS" dargestellt. Das in einer Programmauflistung zwischen Zeilennummer und Zeilentext eingeschobene Leerzeichen ist nicht Bestandteil der Zeile.

Unter Berücksichtigung dieser Voraussetzungen läßt sich das u.a. Beispiel leicht nachvollziehen und nachbilden.

Kernpunkt dieses Beispiels ist, daß alle Datei-Aufrufe sich am Anfang des Programms befinden, dadurch besser zählbar sind und bei späteren Programm-Änderungen nicht verschoben werden.

Beispiel

Ein Programm wertet nach Auswahl des Nutzers eine von drei möglichen Dateien DAT1, DAT2 oder DAT3 aus.

```
10 GOTO 100
20 OPEN "DAT1",0:RETURN
30 IN# "DAT1",A$,B$,C:RETURN
40 CLOSE "DAT1":RETURN

100 CLEAR 1000
110 A=PEEK(30885)*256+PEEK(30884)
120 CLS
130 INPUT "DATEIVERSION (1-3)";X$
140 IF X$<"1" OR X$>"3" THEN 120
150 POKE A+23,ASC(X$)
160 POKE A+42,ASC(X$)
170 POKE A+69,ASC(X$)
180 GOSUB 20
190 GOSUB 30
.
.       bearbeiten der Daten, ggf mehrere Sätze
.       mit GOSUB 30 lesen.
.
400 GOSUB 40
410 END
```

Zeile 10 führt einen Sprung zum eigentlichen Programmanfang durch.

In den Zeilen 20, 30 und 40 sind die Dateiaufrufe als einzelne Subroutinen definiert.

In Zeile 110 wird die Programm-Startadresse ermittelt.

In den Zeilen 130 und 140 wird die gewünschte Dateiversion erfragt.

Diese wird, wenn korrekt, in den Zeilen 150, 160 und 170 in die Dateinamen der Zeilen 20, 30 und 40 übertragen.

In den Zeilen 180, 190 und 400 ist beispielhaft die Dateibearbeitung durch Subroutinen-Aufruf angedeutet.

6. Anwendungsbeispiel "Adreßverwaltung"

Das auf den nachfolgenden Seiten dargestellte Programm "Adreßverwaltung" zeigt ein typisches Anwendungsbeispiel für eine Diskettenbearbeitung mit dem LASER-DOS.

Es erlaubt die Eingabe, Speicherung und Bearbeitung von bis zu 100 Adressen.

Die Adressen werden auf der Diskette in einer Datei "MAILBOX" gespeichert.

Das Bearbeitungsverfahren wurde mit Rücksicht auf die Schwächen des DOS BASIC 1.0 so gewählt, daß bei Programmstart der Dateiinhalt komplett in den Speicher gelesen wird und am Ende der Bearbeitung bei Änderungen wieder komplett auf die Diskette zurückgeschrieben wird.

Bedienung des Programms

Das Programm wird mit **RUN "ANSCHR"** geladen und gestartet. Sofort nach dem Start wird der Inhalt der Datei "MAILBOX" in programminterne Matrizen übertragen. Diese Datei muß sich auf der Diskette befinden, ansonsten wird das Programm nach Ausgabe einer Fehlermeldung beendet.

Nach dem Ladevorgang wird das Menü ausgegeben

(1) NEUER EINTRAG
(2) EINTRAG AKTUALISIEREN
(3) EINTRAG LOESCHEN
(4) EINTRAG LESEN
(5) SORTIERT AUFLISTEN
(6) PROGRAMM BEENDEN

Sie wählen eine dieser Funktionen durch Eingabe der entsprechenden Ziffer.

Nach Abschluß der Funktionen 1 bis 5, diese sind m.E. selbsterklärend, kehrt das Programm zur Menü-Ausgabe zurück.

Nach Abschluß der Funktion 6 wird das Programm beendet.

Wurde der Dateninhalt während des Programmlaufs verändert, so werden die Adressen vor Ausführung der Funktionen 5 und 6, wenn erforderlich, alphabetisch nach Nachnamen und Vornamen sortiert.

In der Funktion 6 werden die Daten bei Änderungen auf die Diskette zurückgeschrieben.

Das Programm erwartet beim Start das Vorhandensein der Datei MAILBOX. Wenden Sie das Programm erstmalig an, so ist keine derartige Datei auf der Diskette vorhanden. Mit folgender Prozedur können Sie sich eine leere Datei "MAILBOX" auf der Diskette anlegen:

```
LOAD "ANSCHR"
RUN 3000
RUN
```

Dies kreiert eine Datei "MAILBOX" und startet anschließend das eigentliche Hauptprogramm.

Zum Programmaufbau

Das Programm ist modular aufgebaut, d.h. jede Funktion ist in einer in sich abgeschlossenen Routine realisiert.

Nach dem Start wird zunächst die Datei "MAILBOX" eingelesen (Zeilen 220 - 280).

Beachten Sie, daß bei dieser Datei zur Endekennung eine Lösung gewählt wurde, bei der vor jedem "echten" Datensatz ein Kennsatz mit einem kurzen alphanum. Text gespeichert wird. Das Einlesen und Auswerten des Kennsatzes erfolgt in den Zeilen 240 und 250, das Lesen des "echten" Satzes in Zeile 280.

Auf die einzelnen Programmroutinen zur Adreßbearbeitung (Funktionen 1-5) soll nicht im Einzelnen eingegangen werden. Diese haben mit dem DOS direkt nichts zu tun.

Sie können sich die Routinen bei Bedarf selbst analysieren.

Nur ein Hinweis. Zum Sortieren der Anschriften (Zeilen 2200 - 2390) wurde ein "SHELL" SORT - Verfahren verwendet, das von der Struktur zwar etwas komplizierter, jedoch von der Laufzeit her erheblich besser als der einfache und übliche "BUBBLE" SORT ist.

Das Zurückschreiben der Daten auf die Diskette erfolgt in den Zeilen 1800 bis 1920.

Die Daten werden aus Sicherheitsgründen zunächst in eine temporäre Datei "TEMP" geschrieben. Die alte Datei "MAILBOX" wird anschließend gelöscht und dann die temporäre Datei in "MAILBOX" umbenannt. Dis hat den Vorteil, daß bei auftretenden Fehlern während des Schreibens (DISK FULL o.ä.) zumindest noch die alte Datei vorhanden ist.

Wenn Sie das Programm interessiert, so tippen Sie es einfach ab und speichern es auf der Diskette mit **SAVE "ANSCHR".**

```
100 '****************************************
110 '*
120 '*        ADRESS - VERWALTUNG
130 '*
140 '****************************************
150 '
160 CLEAR 2000
170 HD$="     ANSCHRIFTENVERZEICHNIS"
180 DIM NN$(99),VN$(99),TI$(99),ST$(99),NR$(99),PL$(99),OT$(99)
200 'ANSCHRIFTENLISTE EINLESEN
210 GOSUB 2450
220 OPEN "MAILBOX",0
230 FOR N = 0 TO 100
240 IN# "MAILBOX",A$
250 IF A$ = "" THEN 280
260 IN# "MAILBOX",NN$(N),VN$(N),TI$(N),ST$(N),NR$(N),PL$(N),OT$(N)
270 NEXT N
280 CLOSE "MAILBOX"
300 'MENUE AUSGEBEN
310 GOSUB 2400
320 PRINT
330 PRINT TAB(4);"(1) NEUER EINTRAG"
340 PRINT TAB(4);"(2) EINTRAG AKTUALISIEREN"
350 PRINT TAB(4);"(3) EINTRAG LOESCHEN"
360 PRINT TAB(4);"(4) EINTRAG LESEN"
370 PRINT TAB(4);"(5) SORTIERT AUFLISTEN"
380 PRINT TAB(4);"(6) PROGRAMM BEENDEN"
390 GOSUB 2500
400 A$=INKEY$: IF A$ < "1" OR A$ > "6" THEN 400
410 IF A$ = "1" THEN 500
420 IF A$ = "2" THEN 700
430 IF A$ = "3" THEN 1200
440 IF A$ = "4" THEN 1300
450 IF A$ = "5" THEN 1500
460 IF SO = 1 GOSUB 2200
470 IF MO = 1 GOTO 1800
480 CLS: END
500 'NEUER EINTRAG
510 GOSUB 2400
520 IF N = 99 PRINT "DATEI MAILBOX BEREITS VOLL": GOTO 2450
530 INPUT "NACHNAME    ";NN$: IF NN$ = "" THEN 510
540 INPUT "VORNAME     ";VN$
550 INPUT "TITEL       ";TI$
560 INPUT "STRASSE     ";ST$
```

```
570 INPUT "HAUSNUMMER  ";NR
580 INPUT "POSTLEITZAHL";PL$
590 INPUT "ORT         ";OT$
600 GOSUB 2000
610 IF GF = 1 THEN 2550
620 NN$(N)=NN$: VN$(N)=VN$: TI$(N)=TI$: ST$(N)=ST$: NR(N)=NR
630 PL$(N)=PL$: OT$(N)=OT$
640 N = N + 1
650 M0 = 1: S0 = 1
660 PRINT: PRINT "EINTRAG DURCHGEFUEHRT"
670 GOTO 2600
700 'EINTRAG AKTUALISIEREN
710 GOSUB 2100: GOSUB 2000
720 IF GF = 0 THEN 2700
730 GOSUB 2400
740 PRINT "1. NACHNAME: ";NN$(I)
750 PRINT "2. VORNAME:  ";VN$(I)
760 PRINT "3. TITEL:    ";TI$(I)
770 PRINT "4. STRASSE:  ";ST$(I)
780 PRINT "5. HAUSNR.: ";NR(I)
790 PRINT "6. PLZ:      ";PL$(I)
800 PRINT "7. ORT:      ";OT$(I)
810 GOSUB 2500
820 A$=INKEY$: IF A$ < "0" OR A$ > "7" THEN 820
830 IF A$ = "0" THEN 300
840 IF A$ > "1" THEN 890
850 INPUT "NACHNAME ";NN$
860 IF NN$ = "" THEN 730
870 IF NN$<>NN$(I) THEN NN$(I)=NN$: S0 = 1: M0 = 1
880 GOTO 730
890 IF A$ > "2" THEN 930
900 INPUT "VORNAME ";VN$
910 IF VN$<>VN$(I) THEN VN$(I)=VN$: S0 = 1: M0 = 1
920 GOTO 730
930 IF A$ > "3" THEN 970
940 INPUT "TITEL" ";TI$
950 IF TI$<>TI$(I) THEN TI$(I)=TI$: M0 = 1
960 GOTO 730
970 IF A$ > "4" THEN 1010
980 INPUT "STRASSE ";ST$
990 IF ST$<>ST$(I) THEN ST$(I)=ST$: M0 = 1
1000 GOTO 730
1010 IF A$ > "5" THEN 1050
1020 INPUT "HAUSNUMMER ";NR
```

```
1030 IF NR<>NR(I) THEN NR(I)=NR: MO = 1
1040 GOTO 730
1050 IF A$ > "6" THEN 1090
1060 INPUT "POSTLEITZAHL ";PL$
1070 IF PL$<>PL$(I) THEN PL$(I)=PL$: MO = 1
1080 GOTO 730
1090 INPUT "ORT ";OT$
1100 IF OT$<>OT$(I) THEN OT$(I)=OT$: MO = 1
1110 GOTO 730
1200 'EINTRAG LOESCHEN
1210 GOSUB 2100: GOSUB 2000
1220 IF GF = 0 THEN 2700
1230 PL$(I) = "XXXX"
1240 PRINT: PRINT "EINTRAG GELOESCHT"
1250 MO = 1: GOTO 2600
1300 'EINTRAG LESEN
1305 IF SO = 1 GOSUB 2200
1310 GOSUB 2100: GOSUB 2000
1320 IF GF = 0 THEN 2700
1330 GOSUB 2400
1340 PRINT TI$(I)
1350 PRINT VN$(I)" "NN$(I)
1360 IF ST$(I) = "" THEN 1390
1370 PRINT ST$(I);
1380 IF NR(I) = 0 PRINT " " ELSE PRINT NR(I)
1390 IF PL$(I) = "" THEN 1395 ELSE PRINT PL$(I)"  ";
1395 PRINT OT$(I)
1400 I = I + 1: GOSUB 1610
1410 IF I < N THEN 1330 ELSE 300
1500 'LISTE AUSGEBEN
1510 IF SO = 1 GOSUB 2200
1520 I = 0
1530 GOSUB 2400
1540 IF N = 0 PRINT "KEINE EINTRAEGE VORHANDEN": GOTO 2600
1550 FOR J = 1 TO 12
1560 IF I = N THEN 1600
1570 IF PL$(I) = "XXXX" THEN 1590
1580 PRINT NN$(I)", "VN$(I)
1590 I = I + 1: NEXT J
1600 GOSUB 1610: IF I < N THEN 1530 ELSE 300
1610 PRINT @480,"<RETURN> = WEITER, <E> = ENDE";
1620 A$ = INKEY$
1630 A$ = INKEY$: IF A$ = "" THEN 1630
1640 IF A$ = "E" THEN I = N: RETURN
```

```
1650 IF A$ <> CHR$(13) THEN 1630
1660 RETURN
1800 'DATEN AUF DISKETTE SCHREIBEN
1810 GOSUB 2450
1820 OPEN "TEMP",1
1830 IF N = 0 THEN 1890
1840 FO I = 0 TO N-1
1850 IF PL$(I) = "XXXX" THEN 1880
1860 PR# "TEMP","A"
1870 PR# "TEMP",NN$(I),VN$(I),TI$(I),ST$(I),NR(I),PL$(I),OT$(I)
1880 NEXT I
1890 CLOSE "TEMP"
1900 ERA "MAILBOX"
1910 REN "TEMP","MAILBOX"
1920 CLS: END
2000 'EINTRAG IN LISTE SUCHEN
2010 FOR I = 0 TO N-1
2020 IF PL$(I) = "XXXX" THEN 2060
2030 IF NN$ <> NN$(I) THEN 2060
2040 IF VN$ = "" THEN 2070
2050 IF VN$ = VN$(I) THEN 2070
2060 NEXT I: GF = 0: RETURN
2070 GF = 1: RETURN
2100 'SUCHKRITERIEN EINLESEN
2110 GOSUB 2400
2120 INPUT "NACHNAME";NN$
2130 IF NN$ = "" THEN 2110
2140 INPUT "VORNAME ";VN$
2150 RETURN
2200 'SORTIEREN DER EINTRAEGE
2210 GOSUB 2450: SO = 0
2220 IF N < 2 RETURN
2230 M = N
2240 M = INT (M/2): IF M = 0 RETURN
2250 J = 1: K = N - M
2260 I = J
2270 L = I + M
2280 X = I - 1: Y = L - 1
2290 IF NN$(X) < NN$(Y) THEN 2390
2300 IF NN$(X) > NN$(Y) THEN 2320
2310 IF VN$(X) <= VN$(Y) THEN 2390
2320 NN$=NN$(X): VN$=VN$(X): TI$=TI$(X): ST$=ST$(X): NR=NR(X)
2330 PL$=PL$(X): OT$=OT$(X): NN$(X)=NN$(Y): VN$(X)=VN$(Y)
2340 TI$(X)=TI$(Y): ST$(X)=ST$(Y): NR(X)=NR(Y)
```

```
2350 PL$(X)=PL$(Y): OT$(X)=OT$(Y): NN$(Y)=NN$
2360 VN$(Y)=VN$: TI$(Y)=TI$: ST$(Y)=ST$: NR(Y)=NR
2370 PL$(Y)=PL$: OT$(Y)=OT$
2380 I = I - M: IF I > 0 THEN 2270
2390 J = J + 1: IF J > K THEN 2240 ELSE 2260
2400 'KOPFZEILE AUSGEBEN
2410 CLS: PRINT HD$: PRINT: RETURN
2420 'BITTE WARTEN
2460 CLS: PRINT @228,"***** BITTE WARTEN *****"
2470 RETURN
2500 'ZIFFER EINLESEN
2510 PRINT: PRINT "BITTE ZIFFER EINGEBEN (0=ENDE)": PRINT
2520 A$ = INKEY$
2530 RETURN
2550 'MELDUNG "ANSCHRIFT VORHANDEN"
2560 PRINT: PRINT "ANSCHRIFT BEREITS VORHANDEN"
2600 'WARTEN AUF RETURN
2610 PRINT: INPUT "WEITER MIT <RETURN>";X
2620 GOTO 300
2700 'MELDUNG "ANSCHRIFT NICHT DA"
2710 PRINT: PRINT "ANSCHRIFT NICHT VORHANDEN"
2720 GOTO 2600
3000 OPEN "MAILBOX",1
3010 CLOSE "MAILBOX"
3020 END
```

Aufbau und Organisation der Diskette

Aufbau der Diskette nach Initialisierung

Bevor mit einer Diskette gearbeitet werden kann, muß auf ihr die Grundstruktur der Spuren und Sektoren vorhanden sein.

Diese Grundstruktur wird mit Hilfe der Initialisierung (INIT - Kommando) auf die Diskette geschrieben.

Sie besteht aus 40 Spuren a 16 Sektoren mit jeweils 128 Bytes Datenkapazität. Wie bereits im Abschnitt "Aufzeichnungsstruktur" vermerkt, kommt zu jedem Sektor noch ein bestimmter "Overhead", der aus notwendigen Synchronisations- und Adressierungsfeldern besteht. Daraus ergibt sich eine Gesamtlänge von 154 Bytes pro Sektor.

Ein solcher Sektor hat folgende Grundstruktur:

Byte 0 - 6	Adreß-Synchronisation	7 x X'80'
Byte 7 - 10	Adreßmarke	X'FE E7 18 C3'
Byte 11 - 13	Adreßfeld Byte 11 = Spurnummer (0 - 39) Byte 12 = Sektornummer (0 - 15) Byte 13 = Prüfsumme "Adreßfeld" (Spur# + Sektor#)	
Byte 14 - 19	Daten-Synchronisation	6 x X'80'
Byte 20 - 23	Datenmarke	X'C3 18 E7 FE'
Byte 24 - 151	Datenfeld	= 128 Byte
Byte 152 - 153	Prüfsumme "Datenfeld"	

Bei der Initialisierung wird jeder Sektor vollständig geschrieben, wobei Datenfeld und Prüfsumme (Byte 24 - 153) = X'00' gesetzt werden.

Die Sektoren werden nicht fortlaufend rund um die Diskette numeriert, sondern in Dreiersprüngen angeordnet (siehe Bild 1.6). Dadurch wird erreicht, daß fortlaufende Sektoren einer Spur während einer Umdrehung der Diskette erreicht werden können, wenn eine bestimmte Verarbeitungszeit dazwischen nicht überschritten wird.

Diese beträgt 94 ms vom Ende eines Sektors bis der Anfang des nächsten Sektors in der numerischen Reihenfolge unter dem Schreib-/Lesekopf erscheint. 94 ms sind eine riesige Zeitspanne, in der man umfangreiche Datenmanipulationen durchführen kann.

Von den Spuren einer Diskette stehen 39 zur Speicherung von Programmen und Daten zur Verfügung.

Die erste Spur einer Diskette (Spur 0) dient der Diskettenverwaltung. Auf ihr befinden sich das Inhaltsverzeichnis der Diskette und eine Sektoren-Belegungsübersicht.

Das Inhaltsverzeichnis

In den ersten 15 Sektoren der Spur 0 (Sektor 0 - 14) befindet sich das Inhaltsverzeichnis der Diskette.

Jeder Eintrag belegt einen Platz von 16 Bytes.

Dies ergibt eine Kapazität von 8 Einträgen pro Sektor und 8 x 15 = 120 Einträgen im gesamten Inhaltsverzeichnis (siehe Fehlermeldung "?DIRECTORY FULL").

Ein Eintrag im Inhaltsverzeichnis hat folgende Struktur:

Byte 0	Belegungsstatus / Dateityp 0 - Ende der benutzten Einträge im Inhaltsverzeichnis 1 - freigegebener Eintrag (z.B. nach einem "ERA") D - Eintrag verweist auf eine Daten-Datei T - Eintrag verweist auf eine Text-Datei (BASIC-Progr.) B - Eintrag verweist auf eine Binär-Datei (Maschinen-Programm)
Byte 1	Trennzeichen (immer ':')

Byte 2 - 9	Dateiname
Byte 10 - 11	Adresse des ersten Sektors dieser Datei Byte 10 - Spurnummer Byte 11 - Sektornummer
Byte 12 - 13	nur bei Dateityp = T oder B Programm-Anfangsadresse im Speicher
Byte 14 - 15	nur bei Dateityp = T oder B Programm-Endadresse+1 im Speicher

Mit dem Kommando "DIR" werden einfach ,ohne jede Aufbereitung, die ersten 10 Bytes jedes belegten Eintrags auf dem Bildschirm ausgegeben.

Wird eine Datei gelöscht, so wird lediglich das Statusbyte (Byte 0) auf '1' gesetzt. Alle anderen Einträge bleiben erhalten.

Die Sektorenverwaltung

Im letzten Sektor der Spur 0 befindet sich die Belegungsübersicht für die Sektoren der Diskette.

Für jeden Sektor ab Spur 1 ist dort ein Bit reserviert, das anzeigt, ob der entsprechende Sektor frei (Bit = 0) oder belegt (Bit = 1) ist.

Bei 39 Spuren und 16 Sektoren pro Spur ergibt das 624 erforderliche Bits oder 78 Bytes, die in diesem Sektor relevante Informationen enthalten.

Beim Schreiben einer Datei dient diese Belegungs-übersicht zur Ermittlung der erforderlichen Sektoren zur Speicherung. Es werden stets von vorn nach hinten die Sektoren belegt und dabei durch Löschen evtl. entstandene Lücken aufgefüllt.

Zuordnungsbeispiel:

Spur 0/ Sektor 15

Byte 0	==>	Spur 1, Sektoren 0 - 7
Byte 1	==>	Spur 1, Sektoren 8 - 15
Byte 2	==>	Spur 2, Sektoren 0 - 7
.		
.		
.		
Byte 77	==>	Spur 39, Sektoren 8 - 15

Speicherung von Programmen und Dateien

Alle auf der Diskette gespeicherten Programme erhalten einen entsprechenden Eintrag im Inhaltsverzeichnis, wobei im ersten Byte der Typ der Datei oder des Programms vermerkt ist.

Zwischen Textdateien (BASIC-Programmen) und Binärdateien (Maschinen-Programmen) ist diese Typenkennzeichnung der einzige Unterschied. Die Aufzeichnungsstrukturen sind identisch.

Die unterschiedliche Typkennzeichnung bewirkt eine andere Behandlung nach dem Laden bzw. Starten eines solchen Programms (siehe Kommandobeschreibungen LOAD/RUN bzw. BLOAD/BRUN).

Im Inhaltsverzeichnis befindet sich in den Bytes 10 u. 11 ein Zeiger auf den ersten von diesem Programm belegten Sektor.

In den Bytes 12 - 15 des Inhaltsverzeichnisses befinden sich die Angaben, in welchen Speicherbereich dieses Programm beim Laden zu übertragen ist. Dabei enthalten die Bytes 12 u. 13 die Anfangsadresse und die Bytes 14 u. 15 die Endadresse+1 des übertragungsbereichs.

Die Datensektoren enthalten in den Bytes 0 - 125 des Datenfeldes eine 1:1-Kopie des Speicherbereichs, d.h. in binärer Datendarstellung.

Die von einem Programm belegten Sektoren müssen physikalisch nicht hintereinander liegen, sondern können auf der Diskette verstreut sein. Um ein Programm dennoch zusammenhängend lesen zu können, sind die einzelnen Sektoren untereinander verzeigert.

In den Bytes 126 und 127 des Datenfeldes befindet sich ein Zeiger auf den nächsten Sektor dieses Programms (Spur- und Sektornummer) oder '0' im letzten belegten Sektor.

Bei Daten-Dateien mit der Typkennzeichnung 'D' fehlen im Inhaltsverzeichnis die Angaben über einen zu belegenden Speicherbereich, die Bytes 12 - 15 sind ohne Belang.

In den Bytes 10 und 11 ist ebenfalls der Zeiger auf den ersten belegten Sektor enthalten.

Die einzelnen Sektoren der Datei sind wie bei den Programmen miteinander verzeigert.

Jeder Sektor einer Daten-Datei enthält in den ersten 126 Bytes des Datenfeldes die eigentlichen Daten und in den letzten beiden Bytes des Datenfeldes einen Zeiger auf den nächsten belegten Sektor bzw. '0' am Ende der Datei.

Unterschiedlich zu den beiden anderen Dateitypen ist die Struktur der Daten in den ersten 126 Bytes.

Die Datendarstellung ist ausschließlich im ASCII-Format. Die Speicherung erfolgt auf der Basis von Datensätzen, wobei mit jedem PR#-Kommando ein vollständiger Datensatz in die Datei geschrieben wird.

Datensätze enthalten eine definierte Endekennung. Dies ist das ASCIIZeichen für "Carriage Return" X'0D'.

Ein Datensatz orientiert sich nicht an Sektorengrenzen. Es können mehrere Datensätze in einem Sektor enthalten sein; ein Datensatz kann sich auch über mehrere Sektoren erstrecken. Außer beim ersten Datensatz einer Datei müssen die Datensätze auch nicht an einer Sektorengrenze beginnen.

Innerhalb der Datensätze sind die verschiedenen Datenfelder durch Kommas voneinander getrennt, sie werden beim Einlesen den im IN#-Kommando definierten Variablen zugewiesen.

Speicherresidente Arbeitsbereiche

Zur Bearbeitung der Disketten werden vom DOS in den letzten 310 Bytes des verfügbaren RAM-Speichers verschiedene Datenstrukturen angelegt, in denen sich Bearbeitungs-Vektoren, Datei-Verwaltungsblöcke und Ein-/Ausgabepuffer befinden (Bild 1.7).

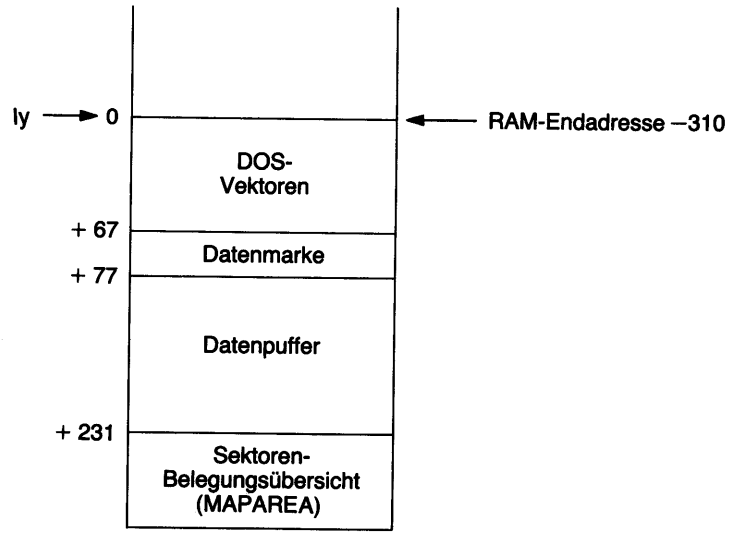


Bild 7.1 Die Speicherbereiche des DOS

DOS - Vektoren

Die ersten 67 Byte dieses DOS-Arbeitsbereichs beinhalten die DOS-Vektoren.

Der Anfang der DOS-Vektoren wird vom DOS bei der System-Initialisierung mit dem Z80-Register 'IY' adressiert. Das DOS erwartet, daß dieses Register von Anwenderprogrammen nicht verändert wird, ansonsten führt es unweigerlich zum System-Crash und evtl. auch zur Zerstörung von Dateninhalten auf der Diskette (bittere Erfahrung des Autors).

Der DOS-Vektorbereich ist wie folgt strukturiert:

DOSVTR = IY

Name	Byte	Offset	Bedeutung
FILNO	1	IY+0	Dateinummer. Bei Bearbeitung einer Daten-Datei steht hier die Nummer des benutzten Datei-Verwaltungsblocks (FCB). 0 = FCB1 , 1 = FCB2
FNAM	8	IY+1	Dateiname. Name der zu bearbeitenden Datei. Ist vom Anwenderprogramm vor jedem Datei-/ Programmzugriff einzutragen.
TYPE	2	IY+9	Dateityp. Byte 1 = Soll-Type. Byte 2 = Ist-Type. Vom Anwenderprogramm ist im ersten Byte der Typ der zu bearb. Datei anzugeben.
DK	1	IY+11	ausgewählter Laufwerk. X'10' = Laufwerk 1 X'80' = Laufwerk 2 bei der Initialisierung wird X'10' gesetzt.
RQST	1	IY+12	Zugriffsart. 0 = Lesen 1 = Schreiben Ist vom Anwenderprogramm zu setzen. Beim BASIC erfolgt dies durch das OPEN-Kommando.
SOURCE	1	IY+13	Ausgangslaufwerk (Quelle) beim DCOPY-Kommando (1 oder 2)
UBFR	2	IY+14	Adresse eines Anwender-Pufferbereichs, in den oder aus dem Daten übertragen werden sollen.

			Beim Laden und Speichern von Programmen ist dies der Programmbereich. Beim Lesen von Daten-Dateien ist es der BASIC Ein-/Ausgabepuffer
DESTIN	1	IY+16	Ziellaufwerk beim DCOPY-Kommando (1 oder 2)
SCTR	1	IY+17	Nummer des zu adressierenden Sektors.
TRCK	1	IY+18	Nummer der zu adressierenden Spur.
RETRY	1	IY+19	Wiederholungszähler bei Lesefehlern (Prüfsumme). Wird bei der Initialisierung auf 10 gesetzt.
DTRCK	1	IY+20	aktuelle Spurnummer, über der sich der Schreib-/Lesekopf befindet.
NSCT	1	IY+21	Merkfeld für den nächsten zu adressierenden Sektor.
NTRK	1	IY+22	Merkfeld für die nächste zu adressierende Spur.
FCB1	13	IY+23	Datei-Verwaltungsblock 1. (Struktur siehe eigene Beschreibung)
FCB2	13	IY+36	Datei-Verwaltungsblock 2. (Struktur siehe eigene Beschreibung)
DBFR	2	IY+49	Zeiger auf den Datenpuffer des DOS für das Schreiben und Lesen eines Sektors Dieser befindet sich gleich im Anschluß an die DOS-Vektoren im Arbeitsbereich.
LTHCPY	1	IY+51	Kopie des Kommandobytes für die Disketten-Steuerung.
MAPADR	2	IY+52	Zeiger auf den Puffer des DOS, in den die Sektoren-Belegungsübersicht zwischengespeichert wird. Dieser befindet sich hinter dem Daten-

			puffer im Arbeitsbereich.
TRKCNT	1	IY+54	Spurzähler für das DCOPY-Kommando.
TRKPTR	1	IY+55	Spurzeiger für das DCOPY-Kommando.
PHASE	1	IY+56	Schrittpuls-Raster für die Spurverstellung.
DCPYF	1	IY+57	Flag für DCOPY
RESVE	10	IY+58	reserviert für Erweiterungen.

Datei-Verwaltungsblöcke (File Control Blocks = FCB)

Innerhalb der DOS-Vektoren befinden sich zwei 13 Bytes große Datei-Verwaltungsblöcke, FCB1 und FCB2.

Diese sind bei der Bearbeitung von Daten-Dateien erforderlich, um Status- und Kontrollinformationen über die im Zugriff befindliche Datei zu führen.

Vom OPEN - Kommando wird ein freier Datei-Verwaltungsblock ermittelt und mit den erforderlichen Parametern für die zu öffnende Datei versehen.

Die Kommandos IN# und PR# orientieren sich an dem entsprechenden Datei-Verwaltungsblock, z.B. welcher Sektor der Datei zu lesen ist und an welchem Byte dieses Sektors die Verarbeitung fortzusetzen ist.

Vom CLOSE - Kommando werden die Datei-Verwaltungsblöcke wieder freigegeben.

Da nur zwei dieser Blöcke vorhanden sind, können auch gleichzeitig nur zwei Dateien geöffnet sein.

Ein Datei-Verwaltungsblock hat folgende Struktur:

FCB1 bzw. FCB2

Name	Bytes	Bedeutung
FLAG	1	Zeigt den Status des FCB an. 0 - FCB nicht belegt. 1 - FCB belegt, Datei z.Zt. nicht aktiv 2 - FCB belegt, Datei aktiv. Aktiv bedeutet, daß sich z.Zt. ein aktueller Sektor dieser Datei zur Bearbeitung im Datenpuffer befindet
ACCESS	1	Zugriffsart für diese Datei. 0 - Lesen 1 - Schreiben
FNAM	8	Dateiname
TRK#	1	Spurnummer
SCTR#	1	Sektornummer des z.Zt. in Arbeit befindlichen Sektors der Datei.
PTR	1	Zeiger auf das nächste zu bearbeitende Byte im o.a. Sektor.

Ein- / Ausgabepuffer

Im DOS-Arbeitsbereich befinden sich zwei Pufferbereiche, einer zur Zwischenspeicherung der zu lesenden bzw. zu schreibenden Sektoren und ein zweiter für die Sektoren-Belegungsübersicht.

Datenpuffer (DBFR)

Dieser Puffer hat eine Größe von 154 Bytes und dient als Zwischenspeicher für den direkten Datenaustausch mit der Diskette.

Vom Datenpuffer aus werden beim Schreiben die Sektoren auf die Diskette übertragen, beim Lesen werden die Sektoren von der Diskette in den Datenpuffer übertragen.

Vor dem Datenpuffer werden bei der Initialisierung die 10 Bytes der Datenmarke gesetzt, so daß beim Schreiben eines Sektors ein vollständiger Informationsblock (Datenmarke + Datenfeld) zur Verfügung steht.

Während der normalen Schreib-/Leseoperationen werden nur die ersten 128 Bytes des Datenpuffers zur Aufnahme des Datenfeldes eines Sektors benutzt.

Die volle Länge von 154 Bytes wird lediglich bei der Disketten-Initialisierung benötigt, um einen vollständigen Sektor, einschließlich sämtlicher Synchronisationsfelder, Adreßfelder und Kennmarken aufnehmen zu können.

Sektoren-Belegungsübersicht (MAP)

Am Ende des DOS-Arbeitsbereichs befindet sich ein 80 Bytes großer Pufferbereich, in dem die Sektoren-Belegungsübersicht aus dem Sektor 15 der Spur 0 von der Diskette zwischengespeichert wird.

Bei der Speicherung eines Programms oder dem Schreiben einer Daten-Datei erfolgt die Sektorenauswahl und -belegung ausschließlich in diesem Pufferbereich, nachdem zu Beginn der aktuelle Sektor eingelesen wurde. Erst wenn der Speichervorgang für das Programm abgeschlossen ist, wird auch die Belegungsübersicht wieder auf die Diskette geschrieben.

8. Kommunikation zwischen dem DOS und der Diskettensteuerung

Die Verbindung zwischen dem DOS und der Disketten-Steuerung wird über 4 Ein-/Ausgabe-Ports hergestellt. Dies sind in hexadezimaler Schreibweise die Ports 10H, 11H, 12H und 13H.

PORT 10H = Kommando-Register (O/P LATCH)

über diesen Port wird die Steuerinformation in das Kommandoregister der Disketten-Steuerung übertragen.

Bit 7 = Laufwerk 2 - Auswahl (1 = ja)

Bit 6 = Zugriffsart (0 = Write, 1 = Read)

Bit 5 = Ausgabepuls beim Schreiben auf die Diskette

Bit 4 = Laufwerk 1 - Auswahl (1 = ja)

Bits 3-0 = Schrittphasen für die Spurverstellung

Eine Kopie des Port-Inhalts wird in den DOS-Vektoren im Feld LTHCPY gehalten.

Initialisiert wird dieses Feld mit '0110 0001'.

Die aktuellen Schrittphasen werden im Feld "PHASE" der DOS-Vektoren gehalten und bearbeitet.

Das ausgewählte Laufwerk steht im Feld DK.

Beim Einschalten eines Laufwerks werden Laufwerksauswahl (DK) und die Schrittphase (PHASE) mit dem Inhalt von LTHCPY verknüpft.

PORT 11H = READ and STROBE SHIFT-Register

über diesen Port werden die Daten von der Diskette gelesen. Sie werden bitweise von der Diskettensteuerung von links seriell eingeschoben.

PORT 12H = POLL DATA

Über diesen Port erfolgt die Synchronisation beim Lesen.

Es wird ein negativer Puls erzeugt, wenn an Port 11 das nächste Bit zur Verfügung steht.

PORT 13H = Schreibschutz-Status (READ/WRITE PROTECT STATUS)

Über diesen Port kann der Schreibschutz-Status einer Diskette ermittelt werden (Schreibschutzkerbe überklebt oder nicht).

Das Ergebnis wird in Bit 7 übergeben.
(1 = schreibgeschützt)

9. Die wichtigsten DOS-Routinen und ihre Anwendung in Maschinen-Programmen

Aufruf und Übersicht

Wie bereits mehrfach erwähnt, belegt das DOS den Adreßraum 4000H bis 5FFFH. Es befindet sich in ROM-Bausteinen, die in die Diskettensteuerung eingebaut sind. Die Diskettensteuerung ist am System-Bus des Rechners angeschlossen.

Das Vorhandensein einer Diskettensteuerung wird beim Einschalten des Rechners durch Abprüfen einer bestimmten Bytefolge (AA 55 E7 18) ermittelt. Wird diese Bytefolge gefunden, so wird automatisch die darauf folgende Initialisierungsroutine aufgerufen, die u.a. einen BASIC-Vektor umhängt, so daß die speziellen DOS-Befehle erkannt und ausgeführt werden können.

Dies ist der RESTART 10 - Vektor an der Adresse 7803H im BASIC-Kommunikationsbereich. Wird vom BASIC dieser Vektor innerhalb der Befehlsanalyse aufgerufen (bei Adresse 1D5AH), so prüft zunächst das DOS, ob ein spezieller DOS-Befehl anliegt. Wenn nein, wird das Programm mit der normalen BASIC Befehlsausführung fortgesetzt. Wird ein DOS-Befehl erkannt, so werden die erforderlichen Ausführungsroutinen im DOS aufgerufen.

Von der Initialisierungsroutine des DOS wird auch der DOS-Vektorenbereich am Speicherende aufgebaut und mit Anfangswerten gefüllt.

Ob eine Diskettensteuerung am System-Bus angeschlossen ist, können Sie innerhalb eines Maschinenprogramms auch an der entsprechenden Bytefolge bei Adresse 4000H überprüfen.

Das DOS selbst besteht aus einer Anzahl in sich abgeschlossener Unterroutinen. Ein Großteil davon läßt sich auch aus Maschinenprogrammen aufrufen, so daß dort individuelle Disketten- und Datenbehandlungen programmiert und ausgeführt werden können.

Sie könnten das bestehende Dateisystem des DOS damit bearbeiten, aber auch ganz neue Strukturen und Verarbeitungsformen anlegen, so z.B. die zuvor erwähnten Dateien mit Direktzugriff, die vom DOS nicht unterstützt werden.

Die einzelnen Routinen direkt an ihren Startadressen anzuspringen, wäre eine der Nutzungsmöglichkeiten. Sie würden die Programme damit jedoch auf eine DOS-Version fixieren, da sich bei jeder Änderung des DOS auch ein Teil dieser Adressen verschiebt.

Eine elegantere Lösung ist die Nutzung einer Sprungleiste am Anfang des DOS, die vom Hersteller für einen derartigen Nutzungszweck angelegt wurde und den Aufruf der wichtigsten Unterroutinen ermöglicht.

Der Aufruf erfolgt mit Hilfe des Z80-Befehls

CALL xxxxH.

Folgende Unterroutinen sind über diese Sprungleiste erreichbar:

Name	Aufruf	Funktion
PWRON	CALL 4003H	Laufwerk einschalten
PWROFF	CALL 4006H	Laufwerk ausschalten
ERROR	CALL 4009H	DOS-Fehlerbehandlung
RDMAP	CALL 400CH	Sektor-Belegungsraster laden
CLEAR	CALL 400FH	Sektor löschen
SVMAP	CALL 4012H	Sektor-Belegungsraster schreiben
INIT	CALL 4015H	Diskette initialisieren
CSI	CALL 4018H	Befehlsparameter interpretieren
HEX	CALL 401BH	Umwandlung ASCII in HEX
IDAM	CALL 401EH	Adreßmarke auf Diskette suchen
CREATE	CALL 4021H	Eintrag ins Inhaltsverzeichnis schreiben
MAP	CALL 4024H	Einen freien Sektor ermitteln
SEARCH	CALL 4027H	Datei im Inhaltsverzeichnis suchen

FIND	CALL 402FH	freien Platz im Inhaltsverzeichnis suchen
WRITE	CALL 4032H	Sektor auf die Diskette schreiben
READ	CALL 4035H	Sektor von der Diskette lesen
DLY	CALL 4038H	n Millisekunden Verzögerung
STPIN	CALL 403BH	Kopf n Spuren vorsetzen
STPOUT	CALL 403EH	Kopf n Spuren zurücksetzen
LOAD	CALL 4041H	ein Programm laden
SAVE	CALL 4044H	ein Programm speichern

Bevor Sie jedoch eine dieser Unterroutinen aufrufen, müssen oft ganz bestimmte Eingangsparameter gesetzt sein, z.B. Eintrag des Dateinamens in den DOS-Vektor, Laufwerksauswahl im Feld DK usw...

Es ist ferner auch wichtig zu wissen, welche Ergebnisse eine solche Unterroutine wohin liefert und welche möglichen Fehlerkennungen evtl. gemeldet werden.

Sie sollten auch wissen, welche Register in den Unterroutinen verändert werden, damit Sie diese zuvor sichern können.

Die folgenden Seiten beschreiben jede einzelne dieser Unterroutinen mit ihren Eingangs- und Ausgangsgrößen, benutzten Registern und Fehlercodes. Dazu jeweils eine Funktionsbeschreibung und ein Anwendungs-/Aufrufbeispiel.

Zwei zusätzliche Funktionen, die Sie nicht über die Sprungleiste erreichen, aber leicht selbst programmieren können, sind ebenfalls aufgeführt. Dies sind die Laufwerksauswahl und das Abprüfen des Schreibschutzes der Diskette.

Beachten Sie, daß Sie in Ihrem Programm das Register IY nicht verändern. Dort wird bei der Initialisierung die Anfangsadresse der DOS-Vektoren eingetragen, die nicht nur das DOS, sondern auch Sie bei der Verwendung der o.a. Routinen ständig benötigen.

Von einigen Routinen werden im Register A Fehlercodes zurückgegeben. Sie sollten dieses auf alle Fälle prüfen, ob Ihr Aufruf erfolgreich war oder nicht.

Alle Daten, die zwischen dem Rechner und der Diskette bewegt werden, benutzen den Datenpuffer im DOS-Arbeitsbereich als Zwischenspeicher. Denken Sie daran, daß bei jedem Lesen oder Schreiben eines Sektors sein Inhalt verändert wird.

Vom Betriebssystem wird alle 20 ms eine Unterbrechung (Interrupt) erzeugt, die normalerweise dazu benutzt wird, den Bildschirminhalt zu aktualisieren und die Tastatur auszuwerten.

Diese Unterbrechungen sind jedoch bei Diskettenzugriffen, wo es auf ganz genaue Zeitintervalle ankommt, nicht wünschenswert; sie würden einen fehlerfreien Zugriff unmöglich machen.

Aus diesem Grund müssen Sie vor jedem Diskettenzugriff mit DI (disable interrupts) die Unterbrechungen abschalten und anschließend mit EI (enable interrupts) wieder einschalten.

Sie müssen in vielen Fällen auch selbst prüfen, ob die zu bearbeitende Diskette schreibgeschützt ist; ansonsten werden Schreiboperationen trotzdem durchgeführt.

PWRON

Einschalten eines Laufwerks

Aufruf: **CALL 4008H**

Eingang: DK (IY+11) in den DOS-Vektoren = Laufwerkskennung
X'10' = Laufwerk 1
X'80' = Laufwerk 2

Ausgang: -

benutzte Register: A

Fehlerbehandlung: keine

Das in DK ausgewählte Laufwerk wird eingeschaltet. Der Antriebsmotor läuft und die rote LED an der Vorderseite des Laufwerks leuchtet auf.

Sie sollten vor einem Zugriff auf dieses Laufwerk einen Zeitraum von einigen Millisekunden zur Stabilisierung der Umdrehungsgeschwindigkeit abwarten.

Beispiel:

```
.
.
DI                      ;Interrupts ausschalten
LD      A,80H           ;Laufwerk 2 auswählen
LD      (IY+11),A
CALL    4008H           ;Laufwerk einschalten
LD      BC,50           ;50 ms warten
CALL    4038H
.
.
```

Laufwerk 2 wird eingeschaltet, anschließend wartet das Programm 50 ms zur Stabilisierung.

PWROFF

Ausschalten eines Laufwerks

Aufruf: **CALL 400BH**

Eingang: -

Ausgang: -

Benutzte Register: A

Fehlerbehandlung: keine

Ein eingeschaltetes Laufwerk wird ausgeschaltet. Der Antriebsmotor bleibt stehen und die LED an der Vorderseite des Laufwerks erlischt.

Beispiel:

```
.
CALL    400BH           ;Laufwerk ausschalten
.
```

ERROR

DOS - Fehlerbehandlung

Aufruf: **CALL 400EH** oder

JP 400EH

Eingang: A = Fehlercode (0 - 17)

Ausgang: Sprung in den BASIC-Interpreter

Benutzte Register: AF, BC, DE, HL

Fehlerbehandlung: keine

Entsprechend dem im A-Register übergebenen Fehlercode wird eine Fehlermeldung ausgegeben. Ein evtl. eingeschaltetes Laufwerk wird ausgeschaltet (bei Fehlercode > 1).

Diese Routine unterscheidet sich insofern von den anderen Routinen, als nach Abschluß nicht zum aufrufenden Programm, sondern in den BASIC-Interpreter gesprungen wird.
Der Stack-Pointer wird dabei auf den BASIC-Stack gesetzt.

A=	erzeugte Meldung	weiterer Ablauf
0	keine	zum BASIC-Interpreter (1B9AH)
1	?SYNTAX ERROR	Ausgabe und weiterer Ablauf über 1997H
2	?FILE ALREADY EXISTS	zum BASIC-Interpreter (1B9AH)
3	?DIRECTORY FULL	- " -
4	?DISK WRITE PROTECTED	- " -
5	?FILE NOT OPEN	- " -
6	?DISK I/O ERROR	- " -
7	?DISK FULL	- " -

8	?FILE ALREADY OPEN	- " -
9	?DISK I/O ERROR	- " -
10	?DISK I/O ERROR	- " -
11	?UNSUPPORTED DEVICE	- " -
12	?FILE TYPE MISMATCH	- " -
13	?FILE NOT FOUND	- " -
14	?DISK BUFFER FULL	- " -
15	?ILLEGAL READ	- " -
16	?ILLEGAL WRITE	- " -
17	BREAK	zur BREAK-Routine (1DA0H) im BASIC

Beispiel:

```
.
.
LD      A,7             ;Fehlercode 7 laden
CALL    400EH           ;Meldung "DISK FULL" ausgeben.
```

Die Meldung "?DISK FULL" wird ausgegeben und anschließend meldet sich BASIC mit READY.

Anmerkung:

Anhand des Zeilennummerfeldes im BASIC-Kommunikationsbereich (78A2H) wird von der ERROR-Routine unterschieden, ob es sich um einen Direktbefehl oder um einen Programm-Befehl handelt.

Enthält das Feld 78A2H/78A3H einen Wert ungleich X'FFFF', so wird dies als Zeilennummer interpretiert und diese hinter der Fehlermeldung mit ausgegeben (Fehlercodes 1 - 16).

Diese Funktion läßt sich beim Austesten von Maschinenprogrammen vielleicht auch sinnvoll nutzen, indem in 78A2H/78A3H bestimmte Werte gesetzt werden, die Ihnen bei Auftreten eines Fehlers einen Hinweis auf die entsprechende Stelle des Programms geben.

Beispiel:

```
        .
        .
        OR      A               ;prüfen, ob Fehler aufgetreten
        JR      Z,XY            ;nein, weiter!
        LD      HL,10           ;Zeilenkennung setzen
        LD      (78A2H),HL
        JP      400EH           ;Fehlerroutine aufrufen
XY      .
        .
```

Enthält A einen Wert ungleich 0, so wird die entsprechende Fehlermeldung mit dem Hinweis auf den Ort des Auftretens ausgeben.
z.B. A = 3, dann "?DIRECTORY FULL IN 10".

RDMAP

Sektoren - Belegungsraster laden

Aufruf: **CALL 4011H**

Eingang: Das entsprechende Laufwerk muß eingeschaltet sein.

Ausgang: Das Sektoren-Belegungsraster befindet sich in dem 80-Byte Puffer am Ende des DOS-Arbeitsbereichs (MAPAREA).

Das Laufwerk bleibt eingeschaltet.

Benutzte Register: AF, BC, DE, HL

Fehlerbehandlung: Im Fehlerfall wird von dieser Routine selbständig in die ERROR-Routine verzweigt. Eine eigene Auswertung ist nicht möglich.

Die Sektoren - Belegungsübersicht (Belegungsraster) wird aus dem Sektor 15 der Spur 0 von der Diskette in den Arbeitsspeicher geladen und in die MAPAREA am Ende des DOS-Arbeitsbereichs übertragen.

Beachten Sie, daß Sie für das Ein- und Ausschalten des Laufwerks selbst Sorge tragen müssen.

Beispiel:

```
.
.
DI                      ;Interrupts ausschalten
LD      A,10H           ;Laufwerk 1 auswählen
LD      (IY+11),A
CALL    4008H           ;und einschalten
LD      BC,50
CALL    4038H           ;50 ms warten
CALL    4011H           ;Belegungsraster laden
CALL    400BH           ;Laufwerk ausschalten
LD      L,(IY+52)       ;Anfangsadresse Belegungs-
LD      H,(IY+53)       ;raster in HL
EI                      ;Interrupts wieder einschalten
.
.
```

Die Sektoren-Belegungsübersicht wird von der Diskette im Laufwerk 1 geladen. Die Anfangsadresse der MAPAREA wird anschließend im Registerpaar HL zur Verfügung gestellt.

Intern aufgerufene Routinen: READ

CLEAR

Löschen eines Sektors auf der Diskette

Aufruf: **CALL 4014H**

Eingang: Das entsprechende Laufwerk muß eingeschaltet sein.

SCTR (IY+17) = Sektornummer
TRCK (IY+18) = Spurnummer

Ausgang: Der adressierte Sektor wird auf der Diskette physikalisch gelöscht.

Benutzte Register: AF, BC, DE, HL

Fehlerbehandlung: Im Fehlerfall wird von der Routine selbständig in die ERROR-Routine verzweigt. Eine eigene Fehlerbehandlung ist nicht möglich.

Der in den DOS-Vektoren SCTR (IY+17) und TRCK (IY+18) adressierte Sektor wird auf der Diskette physikalisch gelöscht, d.h. mit binären Nullen (X'00') überschrieben.

Beachten Sie, daß Sie für das Ein- und Ausschalten des Laufwerks selber Sorge tragen müssen.

Prüfen Sie vor dem Löschen des Sektors, ob die Diskette nicht schreibgeschützt ist.

Beispiel:

```
.
.
DI                      ;Interrupts ausschalten
LD      A,80H           ;Laufwerk 2 auswählen
LD      (IY+11),A
CALL    4008H           ;und einschalten
LD      BC,50           ;50 ms warten
CALL    4038H
IN      A,(13H)         ;Schreibschutz prüfen
OR      A
LD      A,4
JP      M,400EH         ;schreibgeschützt!
LD      (IY+17),12      ;Sektornummer setzen
LD      (IY+18),28      ;Spurnummer setzen
CALL    4014H           ;Sektor löschen
CALL    4003H           ;Laufwerk ausschalten
EI                      ;Interrupts wieder einschalten
.
```

Sektor 12 in Spur 28 der Diskette in Laufwerk 2 wird gelöscht.

Intern aufgerufene Routinen: WRITE

SVMAP

Sektoren - Belegungsraster auf die Diskette schreiben

Aufruf: **CALL 4017H**

Eingang: Das entsprechende Laufwerk muß eingeschaltet sein.

Das aktuelle Belegungsraster steht im DOS-Arbeitsbereich in der MAPAREA.

Ausgang: Das Sektoren-Belegungsraster wurde in Sektor 15 der Spur 0 auf die Diskette im ausgewählten Laufwerk geschrieben.

Das Laufwerk bleibt eingeschaltet.

Benutzte Register: AF, BC, DE, HL

Fehlerbehandlung: Im Fehlerfall wird von der Routine direkt in die ERROR-Routine verzweigt. Eine eigene Fehlerbehandlung ist nicht möglich.

Die Sektoren - Belegungsübersicht (Belegungsraster) wird aus dem entsprechenden Puffer des DOS-Arbeitsbereichs (MAPAREA) in den Datenpuffer übertragen und von dort in den Sektor 15 der Spur 0 auf die Diskette geschrieben.

Beachten Sie, daß Sie das Ein- und Ausschalten des Laufwerks selbst veranlassen müssen.

Prüfen Sie vor dem Zurückschreiben, ob die Diskette nicht schreibgeschützt ist.

Beispiel:

```
.
.
DI                      ;Interrupts ausschalten
LD      A,10H           ;Laufwerk 1 auswählen
LD      (IY+11),A
CALL    4008H           ;und einschalten
LD      BC,50           ;50 ms warten
CALL    4038H
IN      A,(13H)         ;Schreibschutz prüfen
OR      A
LD      A,4
JP      M,4409H         ;schreibgeschützt!
CALL    4017H           ;Belegungsraster zurückschreiben
CALL    400BH           ;Laufwerk ausschalten
EI                      ;Interrupts wieder einschalten
.
.
```

Die Sektoren - Belegungsübersicht wird aus dem DOS-Arbeitsbereich (MAPAREA) auf die Diskette im Laufwerk 1 zurückgeschrieben (Spur 0, Sektor 15).

Intern aufgerufene Routinen: WRITE

INIT

Diskette initialisieren

Aufruf: **CALL 401AH**

Eingang: DK (IY+11) = Laufwerkskennung
X'10' = Laufwerk 1
X'80' = Laufwerk 2

Ausgang: Die Diskette im ausgewählten Laufwerk wurde initialisiert.

Benutzte Register: AF, BC, DE, HL, BC', DE', HL'

Fehlerbehandlung: Im Fehlerfall wird von der Routine direkt in die ERROR-Routine verzweigt. Eine eigene Fehlerbehandlung ist nicht möglich.

Eine im ausgewählten Laufwerk befindliche Diskette wird neu initialisiert, d.h. sie wird in 40 Spuren a 16 Sektoren unterteilt und mit den entsprechenden Synchronisations- und Kennmarken versehen.

Alle vorher auf dieser Diskette befindlichen Daten werden dabei gelöscht.

Diese Routine führt selbst das Ein- und Ausschalten des Laufwerks durch.

Der Schreibschutz wird von INIT überprüft, die Unterbrechungen werden zu Beginn ausgeschaltet.

Beispiel:

```
        .
        .
        LD      A,10H           ;Laufwerk auswählen
        LD      (IY+11),A
        CALL    401AH           ;Diskette initialisieren
        EI                      ;Interrupts wieder einschalten
        .
        .
```

Die Diskette im Laufwerk 1 wird initialisiert.

Intern aufgerufene Routinen: IDAM
STPIN
STPOUT
DLY

CSI

Befehlsparameter interpretieren

Aufruf: **CALL 401DH**

Eingang: HL = Anfangsadresse eines in Anführungszeichen eingeschlossenen Dateinamens.
Dieser muß mit X'00' oder ':' abgeschlossen sein.

Ausgang: Der Dateiname wurde geprüft und in das Feld **FNAM** der DOS-Vektoren übertragen.

HL = Adresse des Abschlußzeichens

Benutzte Register: AF, B, HL

Fehlerbehandlung: Ist der Dateiname nicht in Anführungszeichen eingeschlossen oder nicht korrekt abgeschlossen, so wird zum BASIC verzweigt und die Meldung "?SYNTAX ERROR" ausgegeben.

Die ersten acht Zeichen eines in Anführungszeichen eingeschlossenen Dateinamens werden in das Feld **FNAM** der DOS-Vektoren übertragen.

Hinter dem abschließenden Anführungszeichen muß sich eine BASIC - Zeilenendekennung X'00' oder ein Kommandotrenner ':' befinden.

Diese Routine wird vom DOS-BASIC zur Syntaxprüfung benutzt.

Ein Diskettenzugriff findet nicht statt.

Beispiel:

```
          .
          .
          LD     HL,DNAM1          ;Dateiname adressieren
          CALL   401DH             ;in FNAM übertragen
          .
          .
DNAM1     DEFM   '"MAILBOX":'
```

Der Dateiname "MAILBOX" wird für eine anschließende Adressierung in den DOS-Vektor FNAM übertragen.

HEX

Umwandlung ASCII in HEX

Aufruf: **CALL 4020H**

Eingang: HL = Anfangsadresse einer 4 Byte hexadezimalen Zahl im ASCII-Format.

Ausgang: DE = äquivalenter hexadezimaler Wert (binär)
HL = Eingangsadresse + 4

Benutzte Register: AF, DE, HL

Fehlerbehandlung: Carry = 0 - kein Fehler

Carry = 1 - Fehler bei der Umwandlung

Diese Routine kann zur Umwandlung einer hexadezimalen Adreßeingabe von der Tastatur ins binäre Äquivalent benutzt werden.

Vom DOS-BASIC wird diese Routine z.B. vom BSAVE-Kommando zur Interpretation und übernahme der Programm-Anfangsadresse und Endadresse benutzt.

Beispiel:

```
          .
          .
          LD     HL,ASCII        ;ASCII-Wert adressieren
          CALL   4020H           ;umwandeln
          JR     NC, A1          ;Carry=0? ok, zu A1
          LD     A,1             ;Carry=1, "SYNTAX ERROR"
          JP     400EH           ;über ERROR-Routine ausgeben
   A1     LD     (BIN),DE        ;Binärwert speichern
          .
          .
 ASCII    DEFM   'A31C'
 BIN      DEFS   2
```

Die im Feld "ASCII" befindliche Zeichenkette wird in den hexadezimalen Wert umgewandelt und im Feld "BIN" gespeichert.

IDAM

Adreßmarke auf der Diskette suchen

Aufruf: **CALL 4023H**

Eingang: Das entsprechende Laufwerk muß eingeschaltet sein.

SCTR (IY+17) = Sektornummer
TRCK (IY+18) = Spurnummer

Ausgang: bei fehlerfreiem Rücksprung befindet sich der Schreib-/Lesekopf direkt vor der Datenmarke des gesuchten Sektors.

Benutzte Register: AF, BC, DE, HL

Fehlerbehandlung:
A = 0 - Adreßmarke gefunden
A = 9 - Adreßmarke nicht gefunden
A = 17 - BREAK-Taste betätigt

wenn A = 0, ist Z-Flag gesetzt
wenn A <> 0, ist Z-Flag gelöscht

Diese Routine wird benutzt, um vor dem Schreiben oder Lesen eines Sektors den Schreib-/Lesekopf vor der Datenmarke dieses Sektors zu positionieren.

IDAM positioniert den Kopf zunächst über der gewünschten Spur und liest dann Adreßmarke nach Adreßmarke, bis der gewünschte Sektor gefunden wurde. Danach muß sofort der Schreib- oder Lesevorgang für das Datenfeld beginnen, da die Diskette sich ja weiterdreht.

In den Routinen READ und WRITE zum Lesen bzw. Schreiben eines Sektors ist IDAM bereits integriert.

Beispiel:

```
.
.
DI                      ;Interrupts ausschalten
LD      A,80H           ;Laufwerk 2 auswählen
LD      (IY+11),A
CALL    4008H           ;Laufwerk einschalten
LD      BC,50           ;50 ms warten
CALL    4038H
LD      (IY+17),6       ;Sektor adressieren
LD      (IY+18),14      ;Spur adressieren
CALL    4023H           ;Sektor suchen
JP      NZ,400EH        ;Fehler oder BREAK !
.
.       Sektor lesen  oder schreiben
.
```

Der Schreib-/Lesekopf soll zum anschließenden Lesen oder Schreiben vor der Datenmarke des Sektors 6 der Spur 14 positioniert werden.

Intern benutzte Routinen: STPIN
STPOUT

CREATE

. Eintrag ins Inhaltsverzeichnis schreiben

Aufruf: **CALL 4026H**

Eingang: Dateiname in FNAM (IY+1)
Dateityp in TYPE (IY+9)

Das Belegungsraster muß geladen sein (MAPAREA).

Das Laufwerk muß eingeschaltet sein.

Ausgang: Die Datei wurde im Inhaltsverzeichnis eingetragen, der erste Datensektor für diese Datei wurde reserviert.

Benutzte Register: AF, BC, DE, HL

Fehlerbehandlung:

A = 0	-	Eintrag durchgeführt
A = 2	-	Datei bereits vorhanden
A = 3	-	kein Platz im Inhaltsverzeichnis
A = 7	-	kein Sektor frei (Diskette voll)
A = 9	-	Eine Adreßmarke wurde nicht gefunden
A = 10	-	beim Lesen trat ein Prüfsummenfehler auf
A = 17	-	die BREAK-Taste wurde betätigt.

Für die in FNAM angegebene Datei wird ein Eintrag ins Inhaltsverzeichnis vorgenommen und der erste freie Sektor für diese Datei reserviert.

Zunächst erfolgt mit SEARCH die Prüfung, ob eine Datei gleichen Namens bereits vorhanden ist. Wenn nein, wird mit FIND ein freier Eintrag im Inhaltsverzeichnis ermittelt und mit MAP ein erster freier Sektor gesucht. Verlief beides erfolgreich, wird der Eintrag ins Inhaltsverzeichnis vorgenommen.

Dazu werden Dateityp, Trenner ':', Dateiname und die Adresse des ersten Sektors in den gefundenen 16-Byte Eintrag des Inhaltsverzeichnisses eingetragen, und das Inhaltsverzeichnis wird zurückgeschrieben.

Das Belegungsraster sollte anschließend auch auf die Diskette zurückgeschrieben werden, da ansonsten der erste Sektor nicht endgültig belegt ist. Vergessen Sie dies, so wird es später zu einer Doppelbelegung diese Sektors führen und damit ggf. zu einem nicht entwirrbaren Durcheinander der Daten.

Schlitzohren können das natürlich ausnutzen und zwei verschiedene Einträge für dieselbe Datei mit unterschiedlichen Namen ins Inhaltsverzeichnis vornehmen. Sinnvoll ?????

Vor dem Aufruf von CREATE sollten Sie auf jeden Fall den Schreibschutz der Diskette prüfen.

Durch Auswertung des A-Registers kann der Erfolg der Aktion überprüft werden.

Beispiel:

```
          .
          .
          DI                        ;Interrupts ausschalten
          LD      (IY+11),10H       ;Laufwerk 1 auswählen
          CALL    4008H             ;und einschalten
          LD      BC,50             ;50 ms Pause
          CALL    4038H
          IN      A,(13H)           ;Schreibschutz prüfen
          OR      A
          LD      A,4
          JP      M,400EH           ;schreibgeschützt !
          CALL    4011H             ;Belegungsraster laden
          LD      HL,DNAM           ;Dateinamen ins Feld
          CALL    401DH             ;FNAM übertragen
          LD      (IY+9),'B'        ;Typ = 'B' setzen
          CALL    4026H             ;Datei ins Inhaltsverzeichnis eintragen
          OR      A                 ;Fehler dabei aufgetreten?
          JP      NZ,400EH          ;ja, zur ERROR-Routine
          CALL    4017H             ;Belegungsraster zurückschreiben
          CALL    4008H             ;Laufwerk ausschalten
          EI                        ;Interrupts wieder zulassen
          .
          .
DNAM      DEFM    '"KARTEI":'
```

Für die Binärdatei "KARTEI" (Typ = B) wird ein Eintrag ins Inhaltsverzeichnis der Diskette im Laufwerk 1 vorgenommen.

Intern benutzte Routinen:

SEARCH
FIND
MAP
READ
WRITE

MAP

Einen freien Sektor auf der Diskette ermitteln

Aufruf: **CALL 4029H**

Eingang: Das Belegungsraster muß im DOS-Arbeitsbereich stehen (MAPAREA).

Ausgang: NSCT (IY+21) = Sektornummer
NTRK (IY+22) = Spurnummer

Der mit NSCT und NTRK adressierte Sektor wurde im internen Belegungsraster (MAPAREA) reserviert.

Benutzte Register: AF, BC, HL

Fehlerbehandlung: A = 0 - Sektor gefunden
A = 7 - kein Sektor mehr frei

Diese Routine ermittelt einen freien Sektor in der internen Belegungsübersicht des DOS (MAPAREA), die zuvor mit dem aktuellen Belegungsraster von Spur 0, Sektor 15 der Diskette gefüllt sein sollte.

Wird ein freier Sektor ermittelt, so wird das entsprechende Bit in der Belegungsübersicht auf 1 gesetzt.

Beachten Sie jedoch, daß eine endgültige Belegung erst dann erfolgt ist, wenn das Belegungsraster auf die Diskette zurückgeschrieben wurde.

Das Ergebnis (die Sektoradresse) wird in den Feldern NSCT und NTRK der DOS-Vektoren übergeben. Wollen Sie auf den Sektor zugreifen, z.B. mit WRITE, so müssen Sie diese Adresse zuvor in die Felder SCTR und TRCK übertragen.

Von der Routine MAP wird kein Diskettenzugriff durchgeführt.

Beispiel:

```
.
.
DI                      ;Interrupts ausschalten
LD      (IY+11),80H     ;Laufwerk 2 auswählen
CALL    4008H           ;und einschalten
LD      BC,50
CALL    4038H           ;50 ms Pause
CALL    4011H           ;Belegungsraster laden
CALL    4029H           ;freien Sektor ermitteln
OR      A               ;Fehler?
JP      NZ,400EH        ;Ja, zur ERROR-Routine
CALL    4017H           ;Belegungsraster zurückschreiben
CALL    4003H           ;Laufwerk ausschalten
EI                      ;Interrupts wieder einschalten
.
.
```

Auf der Diskette im Laufwerk 2 wird ein freier Sektor ermittelt und belegt. Die Adresse des Sektors wird in den Feldern NSCT und NTRK der DOS-Vektoren übergeben.

SEARCH

Datei im Inhaltsverzeichnis suchen

Aufruf: **CALL 402CH**

Eingang: Das Laufwerk muß eingeschaltet sein.

FNAM (IY+1) = Dateiname

Ausgang: wenn Datei vorhanden, Typ der Datei in TYPE+1 (IY+10).
Der Sektor des Inhaltsverzeichnisses mit dem gefundenen Eintrag steht im Datenpuffer.
DE zeigt auf das Byte hinter dem Namen.
SCTR und TRCK enthalten die Adresse des Sektors.

Benutzte Register: AF, BC, DE, HL

Fehlerbehandlung:

A = 0	-	Datei nicht vorhanden
A = 2	-	Datei vorhanden
A = 9	-	Adreßmarke nicht gefunden
A = 10	-	Prüfsummen-Fehler beim Lesen
A = 13	-	Datei nicht vorhanden
A = 17	-	die BREAK-Taste wurde betätigt

Die Routine SEARCH überprüft, ob im Inhaltsverzeichnis der adressierten Diskette bereits eine Datei mit dem in FNAM gespeicherten Namen existiert.

Das Ergebnis der Suche wird im A-Register übergeben.

A = 2 bedeutet, daß ein entsprechender Eintrag vorhanden ist.
Der Sektor des Inhaltsverzeichnisses steht im Datenpuffer und DE zeigt auf das Byte hinter dem Namen des gefundenen Eintrags (= Adresse des 1. Sektors dieser Datei).
Die Felder SCTR und TRCK der DOS-Vektoren enthalten die Sektoradresse innerhalb des Inhaltsverzeichnisses.
Das Feld TYPE+1 (IY+10) enthält den Typ der gefundenen Datei. Diesen müssen Sie ggf. selbst auswerten.

A = 0 oder A = 13
haben die gleiche Bedeutung. Die angegebene Datei ist nicht im Inhaltsverzeichnis der Diskette enthalten.
A=0 - das Ende der gültigen Einträge wurde erreicht.
A=13 - das Ende des Inhaltsverzeichnisses wurde erreicht.

Alle anderen Werte von A weisen auf einen Fehler oder auf die Betätigung der BREAK - Taste hin.

Beispiel:

```
.
.
DI                      ;Interrupts ausschalten
LD      (IY+11),10H     ;Laufwerk 1 auswählen
CALL    4008H           ;und einschalten
LD      BC,50
CALL    4038H           ;50 ms Pause
CALL    4011H           ;Belegungsraster laden
```

```
        LD      HL,DNAM         ;Dateiname in FNAM übertragen
        CALL    401DH
        CALL    4026H           ;Datei im Inhaltsverzeichnis suchen
        OR      A
        JR      Z,A1            ;nicht vorhanden!
        CP      0DH
        JR      Z,A1            ;nicht vorhanden!
        CP      2               ;Fehler?
        JP      NZ,400EH        ;ja, zur ERROR-Routine
        IN      A,(13H)         ;Schreibschutz prüfen
        OR      A
        LD      A,4
        JP      M,400EH         ;schreibgeschützt, zur ERROR-Routine
        EX      DE,HL           ;Adresse des Eintrags in HL
        LD      DE,-10          ;HL auf den Anfang des Eintrags
        ADD     HL,DE
        LD      (HL),1          ;Eintrag freigeben
        CALL    4032H           ;Sektor des Inhaltsverz. zurückschr.
        .
        .       belegte Sektoren im Belegungsraster freigeben
        .
A1      CALL    4017H           ;Belegungsraster zurückschreiben
        CALL    4008H           ;Laufwerk ausschalten
        EI                      ;Interrupts wieder einschalten
        .
        .
DNAM    DEFM    '*TAGEBUCH*:'
```

Die Datei *TAGEBUCH* wird, wenn vorhanden, im Inhaltsverzeichnis der Diskette im Laufwerk 1 gelöscht. Wenn dort nicht vorhanden, wird die Löschroutine übersprungen.

Beachten Sie, daß dieses Beispiel nicht voll auskodiert wurde. Neben der Löschung des Eintrags im Inhaltsverzeichnis müssen Sie auch noch alle belegten Sektoren dieser Datei im Belegungsraster freigeben.

Intern benutzte Routinen: READ

FIND

einen freien Eintrag im Inhaltsverzeichnis suchen

Aufruf: **CALL 402FH**

Eingang: Das Laufwerk muß eingeschaltet sein

Ausgang: SCTR = Sektornummer
TRCK = Spurnummer

Der adressierte Sektor des Inhaltsverzeichnisses steht im Datenpuffer.
HL zeigt auf den Anfang des freien Eintrags.

Benutzte Register: AF, BC, DE, HL

Fehlerbehandlung:

A = 0	-	ok, freier Eintrag ermittelt
A = 3	-	kein freier Eintrag vorhanden
A = 9	-	eine Adreßmarke wurde nicht gefunden
A = 10	-	beim Lesen Prüfsummenfehler aufgetreten
A = 17	-	BREAK - Taste betätigt

Im Inhaltsverzeichnis der eingeschalteten Diskette wird ein freier Eintrag ermittelt.

Im Register A wird das Ergebnis übermittelt.

Wenn erfolgreich (A = 0), befindet sich der entsprechende Sektor des Inhaltsverzeichnisses im Datenpuffer und Registerpaar HL zeigt auf den Anfang des freien Eintrags.

Die Felder SCTR und TRCK enthalten die Adresse dieses Sektors.
Es kann jetzt dort ein Eintrag vorgenommen werden.

Beispiel:

```
.
.
DI                      ;Interrupts ausschalten
LD      (IY+11),10H     ;Laufwerk 1 auswählen
CALL    4008H           ;und einschalten
LD      BC,50
CALL    4038H           ;50 ms Pause
CALL    402FH           ;freien Eintrag ermitteln
OR      A               ;erfolgreich?
JP      NZ,400EH        ;nein, zur ERROR-Routine
CALL    400BH           ;Laufwerk ausschalten
EI                      ;Interrupts wieder einschalten
.
.
```

Auf der Diskette im Laufwerk 1 wird ein freier Eintrag im Inhaltsverzeichnis ermittelt.

Intern benutzte Routinen: READ

WRITE

Sektor auf die Diskette schreiben

Aufruf: **CALL 4032H**

Eingang: Das Laufwerk muß eingeschaltet sein.

Der zu schreibende Sektor muß im Datenpuffer stehen.

Die Adresse des zu schreibenden Sektors muß in den Feldern SCTR und TRCK der DOS-Vektoren stehen.

Ausgang: Der im Datenpuffer stehende Sektor wurde auf die Diskette übertragen.

Benutzte Register: AF, BC, DE, HL, BC', DE', HL'

Fehlerbehandlung:

A = 0	-	ok, Sektor geschrieben
A = 9	-	Adreßmarke nicht gefunden
A = 17	-	die BREAK-Taste wurde betätigt

Die im Datenpuffer befindlichen Daten werden in den adressierten Sektor der Diskette übertragen. Dabei wird die Prüfsumme ermittelt und ans Sektorende gestellt.

Der Sektor wird einschließlich der vor dem Datenpuffer befindlichen Datenmarke (10 Bytes) und der Prüfsumme übertragen, also insgesamt 140 Bytes werden auf die Diskette geschrieben.

Beispiel:

```
.
.
DI                      ;Interrupts ausschalten
LD      (IY+11),10H     ;Laufwerk 1 auswählen
CALL    4008H           ;und einschalten
LD      BC,50
CALL    4038H           ;50 ms Pause
IN      A,(13H)         ;Schreibschutz prüfen
OR      A
LD      A,4
JP      M400EH          ;schreibgeschützt !
LD      (IY+17),10      ;Sektornummer in SCTR
LD      (IY+18),5       ;Spurnummer in TRCK
CALL    4032H           ;Sektor auf Diskette schreiben
OR      A               ;Fehler aufgetreten ?
JP      NZ,400EH        ;ja, zur ERROR-Routine
CALL    400BH           ;Laufwerk ausschalten
EI                      ;Interrupts wieder einschalten
.
.
```

Die Daten aus dem Datenpuffer werden auf Spur 5, Sektor 10 der Diskette im Laufwerk 1 geschrieben.

Intern benutzte Routinen: IDAM

READ

Sektor von der Diskette lesen

Aufruf: **CALL 4035H**

Eingang: Das Laufwerk muß eingeschaltet sein.

Adresse des zu lesenden Sektors in SCTR (IY+17) und TRCK (IY+18) der DOS - Vektoren.
RETRY (IY+19) = Anzahl der Leseversuche

Ausgang: Der gelesene Sektor steht im Datenpuffer.

Benutzte Register: AF, BC, DE, HL

Fehlerbehandlung:

A = 0	-	ok, Sektor wurde gelesen
A = 9	-	Adreßmarke nicht gefunden
A = 10	-	Prüfsumme falsch
A = 17	-	die BREAK-Taste wurde betätigt

Der mit SCTR und TRCK adressierte Sektor wird von der Diskette in den Datenpuffer übertragen.

Dabei wird die Prüfsumme ermittelt und am Sektorende mit dem dort gespeicherten Wert verglichen.

Im Feld RETRY (IY+19) der DOS-Vektoren können Sie angeben, wie viele Leseversuche bei falscher Prüfsumme durchgeführt werden sollen (Standardwert = 10).

Beispiel:

```
.
.
DI                      ;Interrupts ausschalten
LD      (IY+11),80H     ;Laufwerk 2 auswählen
CALL    4008H           ;und einschalten
LD      BC,50           ;50 ms Pause
CALL    4038H
LD      (IY+17),14      ;Sektornummer in SCTR
LD      (IY+18),28      ;Spurnummer in TRCK
```

```
CALL    4035H           ;Sektor lesen
OR      A               ;Fehler aufgetreten?
JP      NZ,400EH        ;ja, zur ERROR-Routine
CALL    400BH           ;Laufwerk ausschalten
EI                      ;Interrupts wieder einschalten
.
.
```

Die Daten des Sektors 14 auf der Spur 28 werden von der Diskette im Laufwerk 2 in den Datenpuffer übertragen.

Intern benutzte Routinen: IDAM

DLY

n Millisekunden Verzögerung

Aufruf: **CALL 4038H**

Eingang: BC = Anzahl Millisekunden

Ausgang: -

Benutzte Register: AF, BC

Fehlerbehandlung: -

Diese Routine bewirkt eine Verzögerung, deren Dauer in Millisekunden durch den Eintrag im Registerpaar BC bestimmt wird.

Beispiel:

```
.
.
DI                      ;Interrupts ausschalten
LD      BC,1000
CALL    4038H           ;1 Sekunde warten
EI                      ;Interrupts wieder zulassen
.
.
```

bewirkt eine Verzögerung von einer Sekunde.

STPIN

Schreib-/ Lesekopf n Spuren vorsetzen

Aufruf: **CALL 403BH**

Eingang: Das Laufwerk muß eingeschaltet sein.

BC = Anzahl Spuren

Ausgang: Der Schreib-/Lesekopf wurde die entsprechende Anzahl Spuren vorgesetzt.
DTRCK (IY+20) enthält die neue aktuelle Spurnummer.

Benutzte Register: AF, BC

Fehlerbehandlung: -

Der Schreib-/Lesekopf wird um die in B enthaltene Anzahl Spuren, jedoch maximal bis zur Spur 39 vorgesetzt.

Beispiel:

```
.
.
DI                      ;Interrupts ausschalten
LD      (IY+11),10H     ;Laufwerk 1 auswählen
CALL    4008H           ;und einschalten
LD      BC,50
CALL    4038H           ;50 ms Pause
LD      B,10            ;Kopf 10 Spuren vorsetzen
CALL    403BH
CALL    4008H           ;Laufwerk ausschalten
EI                      ;Interrupts wieder zulassen
.
.
```

STPOUT

Schreib-/ Lesekopf n Spuren zurücksetzen

Aufruf: **CALL 403EH**

Eingang: Das Laufwerk muß eingeschaltet sein.

BC = Anzahl Spuren

Ausgang: Der Schreib-/Lesekopf wurde die entsprechende Anzahl Spuren zurückgesetzt.
DTRCK (IY+20) enthält die neue aktuelle Spurnummer.

Benutzte Register: AF, BC

Fehlerbehandlung: -

Der Schreib-/Lesekopf wird um die in B enthaltene Anzahl Spuren, jedoch maximal bis zur Spur 0 zurückgesetzt.

Beispiel:

```
.
.
DI                      ;Interrupts ausschalten
LD      (IY+11),10H     ;Laufwerk 1 auswählen
CALL    4008H           ;und einschalten
LD      BC,50
CALL    4038H           ;50 ms Pause
LD      B,5             ;Kopf 5 Spuren zurücksetzen
CALL    403EH
CALL    400BH           ;Laufwerk ausschalten
EI                      ;Interrupts wieder zulassen
.
.
```

LOAD

Laden eines Programms oder Speicherbereichs

Ausruf: **CALL 4041H**

Eingang: Dateiname in FNAM (IY+1)
Dateityp in TYPE (IY+9)

Ausgang: 78A4/A5 = Anfangsadresse
78F9/FA = Endadresse + 1

Benutzte Register: AF, BC, DE, HL

Fehlerbehandlung:

A = 0	-	ok
A = 9	-	Adreßmarke nicht gefunden
A = 10	-	Prüfsummen-Fehler
A = 12	-	Datei gefunden, aber falscher Typ
A = 13	-	Datei nicht vorhanden
A = 17	-	die BREAK-Taste wurde betätigt

Das im Feld FNAM bezeichnete Programm wird von der Diskette in den Speicher übertragen.

Nach erfolgreichem Verlauf werden im BASIC-Kommunikationsbereich bei Adresse 78A4/A5 die Startadresse und bei Adresse 78F9/FA die Endadresse+1 des Speicherbereichs übergeben.

Diese Routine funktioniert nur bei solchen Dateien, die im Inhaltsverzeichnis in den Bytes 12 und 13 die Anfangsadresse und in den Bytes 14 und 15 die Endadresse+1 der Datei enthalten, also nicht bei Standard-Datendateien.

Wird ein Programm mit den BASIC-Befehlen SAVE oder BSAVE oder von Maschinenprogrammen mit der Routine SAVE gespeichert, so erfolgt dieser Eintrag automatisch.

Die Routine LOAD schaltet selbständig das Laufwerk ein und aus und schaltet auch die Unterbrechungen aus.

Beispiel:

```
        .
        .
        LD      (IY+11),80H     ;Laufwerk 2 auswählen
        LD      HL,DNAM         ;Dateiname in FNAM
        CALL    401DH           ;eintragen
        LD      (IY+9),'B'      ;Typ = 'B' (Binär-Datei)
        CALL    4041H           ;Programm laden
        OR      A               ;Fehler aufgetreten ?
        JP      NZ,400EH        ;ja, zur ERROR-Routine
        EI                      ;Interrupts wieder einschalten
        .
        .
DNAM    '"GRAFDR":'
```

Die Binärdatei oder das Maschinenprogramm "GRAFDR" wird von der Diskette in Laufwerk 1 in den Speicher übertragen.

Intern benutzte Routinen: SEARCH
READ

SAVE

Speichern eines Programms oder Speicherbereichs auf Diskette

Aufruf: **CALL 4044H**

Eingang: 78A4/A5 = Anfangsadresse des Speicherbereichs
78F9/FA = Endadresse+1 des Speicherbereichs

Das Laufwerk muß eingeschaltet sein.

FNAM (IY+1) = Datei-/Programmname
TYPE (IY+9) = Typ der Datei

Ausgang: Der adressierte Speicherbereich wurde auf die Diskette übertragen und unter dem angegebenen Namen ins Inhaltsverzeichnis eingetragen.

Benutzte Register: AF, BC, DE, HL

Fehlerbehandlung: Bei auftretenden Fehlern wird von der SAVE-Routine direkt in die ERROR-Routine verzweigt. Eine eigene Fehlerbehandlung ist nicht möglich.

Ein Speicherbereich oder Programm wird aus dem Speicher auf die Diskette übertragen. Anfangs- und Endadresse sind in den entsprechenden BASIC-Zeigern 78A4 und 78F9 zu übergeben.

Dieser Bereich wird unter dem in FNAM angegebenen Namen und dem im ersten Byte von TYPE eingetragenen Typ in das Inhaltsverzeichnis eingetragen.

Die Diskette müssen Sie zuvor selbst einschalten; ausgeschaltet wird sie bei Abschluß des Speichervorgangs von der SAVE-Routine.

Den Schreibschutz sollten Sie vorher auch prüfen.

Beispiel:

```
        .
        LD      HL,8000H        ;Anfangsadresse
        LD      (78A4H),HL
        LD      HL,9000H        ;Endadresse + 1
        LD      (78F9H),HL
        LD      HL,DNAM         ;Dateiname in FNAM
        CALL    401DH           ;übertragen
        LD      (IY+9),'B'      ;Typ = Binärdatei
        LD      (IY+11),10H     ;Laufwerk 1 auswählen
        CALL    4008H           ;und einschalten
        IN      A,(13H)         ;Schreibschutz prüfen
        OR      A
        LD      A,4
        JP      M,400EH         ;schreibgeschützt!
        CALL    4044H           ;Speicherbereich auf Diskette schreiben
        EI                      ;Interrupts wieder einschalten
        .
DNAM    DEFM    '"TESTDAT":'
```

Der Speicherbereich 8000 - 8FFF wird unter dem Namen "TESTDAT" mit dem Typ "B" auf die Diskette übertragen.

Intern aufgerufene Routinen: READ CREATE
MAP SEARCH
WRITE

Zur Ergänzung dieser Routinen sind hier noch zwei Funktionen aufgeführt, die Sie in vielen der vorausgegangenen Beispielen wiederfinden.

DRIVE

Auswahl eines Laufwerks

Diese Funktion ist nicht über die Sprungleiste erreichbar.

Sie ist jedoch leicht zu vollziehen, da lediglich der korrekte Code des ausgewählten Laufwerks in das Feld DK (IY+11) der DOS-Vektoren einzutragen ist.

```
Laufwerk 1   =   LD  (IY+11),10H

Laufwerk 2   =   LD  (IY+11),80H
```

Mit Hilfe dieses Codes wird von der PWRON-Routine das richtige Laufwerk ausgewählt und eingeschaltet.

WPROCT

Schreibschutz prüfen

Sie sind in vielen Fällen selbst dafür zuständig, den Schreibschutz-Status einer Diskette zu prüfen, bevor Sie eine Schreiboperation durchführen.

Die Information darüber erhalten Sie über Port 13H. Ist die Schreibschutzkerbe der Diskette überklebt, so wird in Bit 7 dieses Ports eine 1 übergeben.

Das Laufwerk muß dazu ausgewählt und eingeschaltet sein.

Beispiel:

```
        .
        .
IN      A,(13H)         ;Port 13 einlesen
OR      A               ;Byte prüfen
LD      A,4             ;Fehlercode setzen
JP      M,400EH         ;wenn negativ, dann ist die
                        ;Diskette schreibgeschützt.
        .
        .
```

Ist die Diskette schreibgeschützt, so wird mit Fehlercode 4 in die ERROR-Routine verzweigt und dort die Meldung "DISK WRITE PROTECTED" ausgegeben.

aktiv und kreativ computern

Früher oder später wird bei jedem Computerbesitzer der Wunsch nach einem Diskettenspeicher übermächtig. Zu groß sind die Vorteile des direkten Zugriffs gegenüber der sequentiellen Arbeitsweise. Nur so kann man alle Möglichkeiten seines Geräts voll ausschöpfen.

In diesem Band wird das Disketten-Betriebssystem des Laser-Computers in seinem Aufbau und seiner Anwendung detailliert beschrieben. Neben den BASIC-DOS-Befehlen werden auch die Schnittstelle und Anwendbarkeit in Maschinenprogrammen erläutert. Anwendungsbeispiele erleichtern den Einstieg in die Diskettenwelt.

VOGEL-BUCHVERLAG
WÜRZBURG

ISBN 3-8023-0868-9