Karsten Schramm

Die Floppy 1570/1571

Das Handbuch für die Programmierung der Floppy 1570 und 1571. Mit ausführlich dokumentiertem Listing der DOS-Betriebssystem-Routinen.

Markt & Technik Verlag AG

Vorwort für die elektronische Veröffentlichung:

Das Buch wurde durch Spiro Trikaliotis eingescannt und bearbeitet. Trotz Sorgfalt können Fehler nicht ausgeschlossen werden. Sollten Sie solche finden, kontaktieren Sie mich bitte unter meiner E-Mail-Adresse cbm@trikaliotis.net. Sollten sie Fragen oder Hinweise haben, wenden Sie sich bitte nicht an den Autor des Buches, sondern an mich.

Bitte respektieren Sie, dass die Rechte an dem Buch weiterhin beim Autor, Herrn Karsten Schramm, liegen.

Die Veröffentlichung in elektronischer Form erfolgt mit Genehmung durch den Autor, bei dem ich mich hiermit noch einmal sehr herzlich bedanke. Ebenfalls bedanke ich mich bei Pearson Education Deutschland GmbH, Rechtsnachfolgerin des damaligen Markt&Technik Verlages, die bereitwillig sämtliche Rechte an dem Buch an den Autor zurückübertragen hat. Den Dank möchte ich dabei ausdrücklich Frau Stephanie Eckert und Pia Kleine von Pearson, sowie Herrn David Göhler vom WEKA Zeitschriftenverlag aussprechen; Herr Göhler hat den entscheidenden Tipp gegeben, durch den der richtige Ansprechpartner gefunden werden konnte.

Da ich das durch den Autor eingeräumte Recht zur Veröffentlichung nicht übertragen darf, weise ich darauf hin, dass das Buch nicht anderweitig zur Verfügung gestellt werden darf. Verweisen Sie bitte stattdessen auf meine Web-Seite http://www.trikaliotis.net/Book für den Download.

Der Text wurde so gut wie möglich in Originalform belassen. So wurde insbesondere darauf geachtet, dass die Seitenzahlen so gut wie möglich übereinstimmen. Da aber Seitenwechsel innerhalb eines Absatzes vermieden wurden ist es häufig so, dass der erste oder der letzte Absatz einer Seite auf der vorherigen oder nachfolgenden Seite erscheinen.

Das Dateilisting von „Burstmon 3.1“ (Listing 13.1) wurde mit PETCAT (aus dem VICE Paket, http://www.viceteam.org/) aus einer Original-Datei erzeugt.

Das ROM-Listing in Anhang C konnte per automatischer Bearbeitung nicht in ausreichender Qualität gescannt werden. Daher wurde scriptgesteuert ein Disassembler-Listing des Original-ROMs mittels VICE erzeugt und die Kommentare des Buchs in dieses Listing übertragen. Das Original-Layout blieb erhalten; trotz aller Sorgfalt kann es aber passiert sein, dass hierdurch kleinere Diskrepanzen eingeführt wurden.

Mit @ST markierte Fußnoten auf den Seiten wie auch diese Seite, die sie gerade lesen, sind in den Original-Büchern nicht vorhanden, sondern sind Anmerkungen, die bei der Bearbeitung entstanden sind.

Ich wünsche viel Spaß bei der Lektüre des Buches, welches sich auch hervorragend als Nachschlagewerk eignet.

Spiro Trikaliotis, 26. April 2006.

CIP-Kurztitelaufnahme der Deutschen Bibliothe,

Schramm, Karsten:
Die Floppy 1570/1571; d. Handbuch für d. Programmierung d. Floppy 1570 u- 1571 ;
Mit dokumentiertem Listing d. DOS-Betriebssystem-Routinen Karsten Schramm.
Haar bei München : Markt-und-Technik-Verlag, 1986.
ISBN 3-89090-185-9

ISBN 3-89090-185-9

Einbandgestaltung: Grafikdesign Heinz Rauner
Druck: Jantsch. Günzburg
Printed in Germany

Inhaltsverzeichnis

Liebe Leserin, lieber Leser,

das Ziel dieses Buches ist es, Ihnen die Programmierung der beiden Floppystationen 1570 und 1571 von Commodore nahezubringen. Die Vermittlung des nötigen Wissens soll locker und dennoch schnell und umfassend erfolgen. Es spielt hierbei keine Rolle, ob Sie noch Anfänger oder schon Profi sind. Sie erhalten in jedem Fall ein Nachschlagewerk, das Ihnen in jeder erdenklichen Situation weiterhelfen wird.

Da die beiden neuen Floppystationen äußerst komplexe Geräte sind, ist es unumgänglich, das Buch in mehrere Abschnitte zu unterteilen.

Angefangen wird mit grundsätzlichen Daten und Überlegungen zur Arbeit mit den Floppystationen, bevor dann im nächsten Abschnitt auf die Arbeit mit Dateien (Files) eingegangen wird. Sie lernen alles über den Aufbau der verschiedenen Dateitypen und den grundsätzlichen Aufbau von formatierten Disketten. Wenn Sie das alles schon beherrschen, dann erfahren Sie in den Abschnitten für fortgeschrittene Programmierer, wie der Befehlssatz der Floppies aussieht und wie man diese Direktzugriffs-Befehle anwendet und auszunutzt. Schließlich lernen Sie die beiden Diskettenformate der 1570/71 (MFM und GCR-Format) kennen.

Wenn Sie alle diese Kapitel durchgestanden haben, begeben wir uns auf die Ebene der Maschinensprache und erfahren alles Nötige und Wissenswerte über die verschiedenen BURST-Betriebsarten der Floppystationen. Sie lernen die Geheimnisse des DOS kennen und mit Hilfe des sehr ausführlich dokumentierten DOS-Listings werden Sie in die Lage versetzt, auch die schwierigsten Programmieraufgaben sicher und schnell zu bewältigen. Die Kapitel für die Maschinensprache-Programmierer informieren so umfassend und genau über die wichtigen Problemlösungen, daß auch der weniger versierte Leser sicher in der Lage ist, sofort in die Programmierung des DOS einzusteigen. Es werden grundsätzliche Routinen vorgestellt und Ratschl3.ge zur Programmierung gegeben.

Die Krönung bildet dann unter anderem der Diskmonitor zum Abtippen, der in Kapitel 13 ausführlich beschrieben wird. Dieser Diskmonitor zählt sicher mit zu den komfortabelsten Werkzeugen, die es bisher zum Commodore 128 gibt.

Allen Lesern und denen. die es werden wollen, nun viel Spaß und aufregende Stunden bei der Lektüre dieses Buches und der Entdeckung der beiden Floppystationen 1570 und 157l.

Karsten Schramm

Übrigens,

bevor ich mein Vorwort abschließe, möchte ich noch allen denjenigen danken, die direkt oder auch indirekt zum Gelingen dieses Buches beigetragen haben:

Vielen Dank dabei an meine Familie, die mich durch ihr Verständnis für meine Arbeit immer mehr oder weniger stark unterstützte. Ein Dankeschön an Michael Thomas und Boris Schneider, die sich stets bemüht haben, mich bei dem Buch so gut wie möglich zu unterstützen. Schließlich noch ein Dankeschön an die Firma Commodore, die bereitwillig alle Informationen über die Diskcontroller der 1570/71 zur Verfügung stellte.

1 Der Umgang mit der Floppy 1570/71

Bevor wir uns an die Arbeit mit der Floppystation 1570 oder 1571 heranwagen, noch einige Worte zu diesem Buch: Der fortgeschrittene Programmierer unter Ihnen kann die ersten drei Kapitel ohne weiteres überspringen. Arbeiten Sie jedoch das erste Mal mit einer Commodore-Floppy, so sollten Sie sich in den folgenden Kapiteln gleich etwas genauer damit beschäftigen. Zuerst wollen wir die Unterschiede zwischen der 1570 und 1571 betrachten. Dieses Buch wird Ihnen beide der sehr ähnlichen Floppystationen nahebringen.

Generell besteht der Unterschied beider Floppies nur darin, daß die 1571 im Gegensatz zur 1570 über zwei Schreib-/Leseköpfe verfügt, während die 1570 nur einen einzigen Schreib-/Lesekopf besitzt. Dieser Unterschied erlaubt es der 1571, alle Disketten beidseitig zu beschreiben, was sich in einer höheren Speicherkapazität pro Diskette äußert. Diese Tatsache ist jedoch für das Erlernen der Programmierung einer der beiden Floppystationen ohne Belang, so daß wir uns gleich daran machen, die Floppystation in Betrieb zu nehmen.

1.1 Inbetriebnahme

Wenn Sie mit Peripheriegeräten arbeiten, das heißt, mit Geräten. die an den Computer angeschlossen werden, so haben Sie es in der Rege! mir elektromechanischen Teilen zu tun, die einem gewissen Verschleiß unterliegen und besonders vorsichtig zu behandeln sind.

Eine Floppystation enthält beispielsweise mehrere Motoren und genau eingestellte Tonköpfe, die sich bei zu roher Behandlung verstellen oder sogar zerstört werden können. Gehen Sie aus diesem Grund besonders vorsichtig mit Ihrer Floppy um. Das gilt natürlich ebenso für Drucker und Winchester-Plattenlaufwerke.

Wenn Sie Ihr Laufwerk auspacken, so müssen in der Originalschachtel folgende Dinge enthalten sein:

- ein Netzkabel
- - ein Kabel für die Verbindung zum Computer (für C64 und C128 gleich
- - eine Diskette (1570 / 1571 TEST/DEMO)
- - und natürlich die Floppystation.

Stellen Sie die Floppystation an Ihrem Arbeitsplatz neben dem Computer auf. Achten Sie dabei auf einen möglichst großen Sicherheitsabstand zum Fernseher oder zum Monitor, da deren Ablenkeinheit den Betrieb der Floppy stören kann.

Die beigefügte Diskette legen Sie am besten sofort an einen sicheren Ort, womit ein Platz weit weg von Magnetfeldern (Fernseher, Monitor, Lautsprecherboxen) und möglichst ohne Staub oder Zigarettenqualm gemeint ist. Diese Aufbewahrungsvorschrift gilt selbstverständlich auch zukünftig für alle anderen Disketten.

Haben Sie alle Geräte aufgestellt, so achten Sie darauf, daß alle Netzschalter auf OFF stehen, und stellen Sie die Netzverbindungen her. Danach nehmen Sie das Verbindungskabel für die Floppy und den Computer, und stecken es bei der Floppy in eine der beiden vorhandenen Buchsen auf der Rückseite des Geräts (Bild 1.1 zeigt die Rückseiten der 1570 und der 1571; die Buchsen sind mit einem Pfeil gekennzeichnet.). Jetzt stecken Sie das andere Ende des Kabels in die entsprechende Buchse am Computer (entweder ein VC20/C64 oder ein CI28). Auch diese Buchse befindet sich an der Rückseite des entsprechenden Gerätes und ist in Bild 1.2 mit einem Pfeil gekennzeichnet. Ein Einstecken in eine verkehrte Buchse ist bei den Commodore-Computern wegen der unterschiedlichen Formen ausgeschlossen.

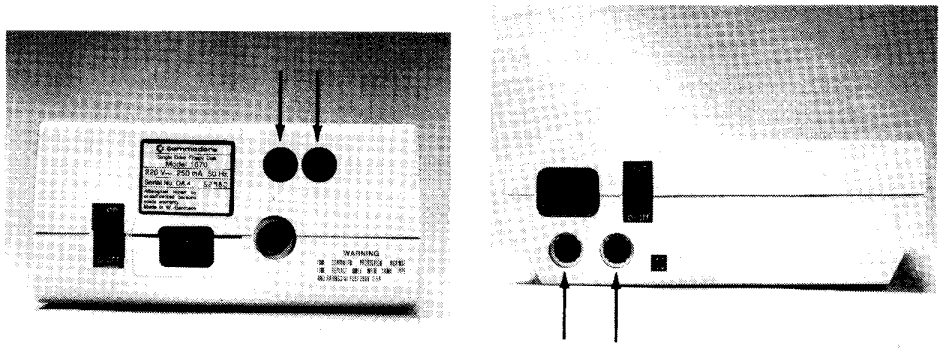


Bild 1.1 *Die Rückseiten der 1570 und 1571*

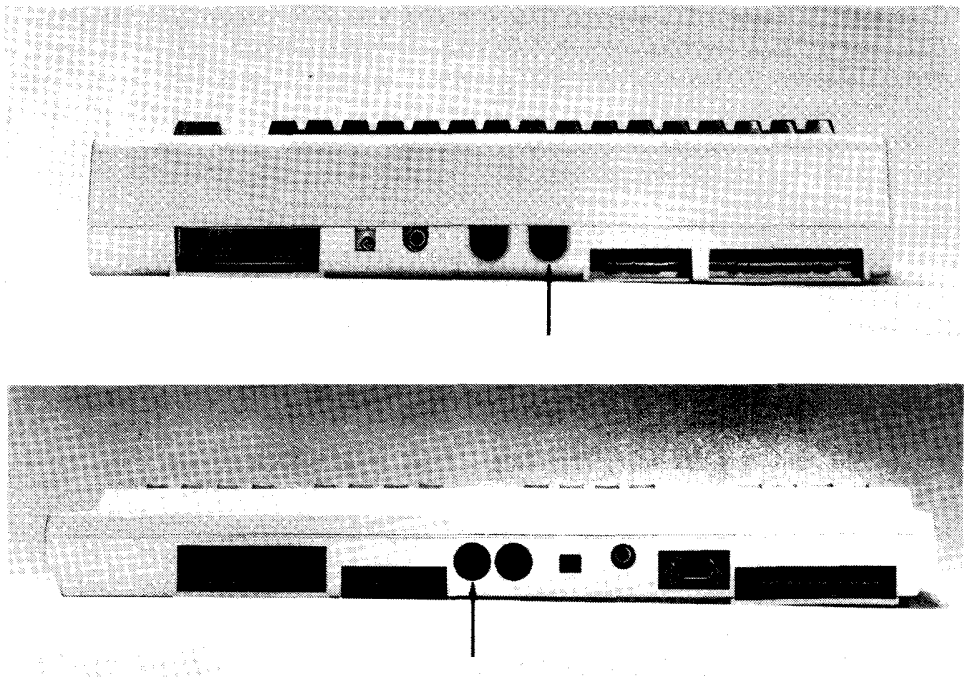


Bild 1.2 *Die Rückseiten von C64 und CI2B*

Bevor Sie die Geräte einschalten, sollten Sie darauf achten, daß Sie die Pappscheibe aus der Öffnung des Floppylaufwerks herausnehmen. Diese Transportsicherung ist gut aufzuheben, um sie bei einem späteren Transport der Floppystation wieder einlegen zu können.

Haben Sie das alles erledigt, so können Sie die Geräte einschalten. Bei einem C64 ist die Reihenfolge des Einschaltens ohne Bedeutung. Bei einem C128 sollten Sie zweckmäßigerweise zuerst die Floppystation und dann den Computer einschalten. Warum das so ist, wird später noch besprochen.

Bei einem C128 läuft die Floppy nach dem Einschalten des Computers an und versucht einen Diskettenzugriff. Wir gehen einmal davon aus. Daß jetzt keine Diskette im Laufwerk liegt. Die Floppy beginnt seltsam zu 'rattern', wobei die Betriebsanzeige in unregelmäßigen Abständen sehr schnell aufblinkt. Kurze Zeit später hat sich die Floppy dann beruhigt und der Computer meldet sich mit READY. Dieser Diskettenzugriff ist vollkommen normal.

Ist alles in Ordnung, so leuchtet bei der 1571 dauernd die rote und bei der 1570 dauernd die grüne Leuchtdiode (LED) am Laufwerk. Diese beiden

LEDs zeigen jedoch bei beiden Geräten das gleiche an: den eingeschalteten Netzschalter. Die jeweils andere LED am Laufwerk zeigt Ihnen an, ob die Floppy bei der Arbeit ist. Sie sollten eine Diskette also nur dann aus dem Laufwerk herausholen oder in das Laufwerk einlegen, wenn diese zweite LED nicht leuchtet. Eine Ausnahme besteht jedoch dann, wenn Sie von dem jeweils laufenden Programm ausdrücklich dazu aufgefordert werden.

1.2 Das Laden und Speichern von Programmen

Nachdem Sie die Computeranlage betriebsbereit gemacht haben, tippen Sie gleich einmal ein Programm ein, das als Dummy zum Erlernen des Diskettenbetriebs dient.

Tippen Sie also zum Beispiel folgende Zeilen in den Computer:

```
10 REM DAS IST EIH TEST
20 REM DAS IST DAS ENDE DES TESTS
30 END
```

Dieses kleine Programm wird bei Arbeitsende, wenn wir den Computer ausschalten, zwangsläufig verlorengehen, da der Computer Programme oder Daten nur im eingeschalteten Zustand behalten kann. Nehmen wir also jetzt einmal an, das sei unser erstes selbstgeschriebenes Programm, das wir unter keinen Umständen verlieren wollen.

Die Lösung des Problems ist eine Floppystation. Mit ihrer Hilfe können Programme auf Disketten abgespeichert und so gegen Verlust geschützt werden. Die Disketten dazu können Sie einzeln oder in Packungen zu mehreren in fast jedem Geschäft kaufen, das auch Computer oder Computerzubehör anbietet.

Wenn Sie sich noch keine Disketten besorgt haben, sollten Sie das schnellstens nachholen. Hier ein paar Tips für den Einkauf von Disketten für die Floppy 1570 und 1571:

Die Disketten sollten:

- 5,25 Zoll Durchmesser haben (es gibt auch 3-, 3,5- und 8-Zoll Disket ten, die jedoch nicht verwendet werden können);
- einen Innenloch- Verstärkungsring enthalten (das erhöht die Lebensdauer der Disketten, ist jedoch nicht unbedingt notwendig);
- für einfache Schreibdichte (single density) ausgelegt sein. (Es gibt auch doppelte oder vierfache Schreib dichte (double density, quattro density).

Das ist für die 1570/71 zwar möglich, aber nicht notwendig. Diese höherwertigen Disketten sind außerdem in der Regel teurer.)

- bei der 1570 einseitig und bei der 1571 zweiseitig zu beschreiben sein
- (1570 single sided, 1571 double sided).

Von welcher Firma die Disketten sind, spielt normalerweise keine Rolle.

Jetzt aber wieder zurück zum Thema. Nehmen Sie eine dieser neuen Disketten aus der Packung, wobei Sie immer besonders sorgfältig vorgehen sollten, und sehen Sie sich diese Diskette einmal an (Bild 1.3).

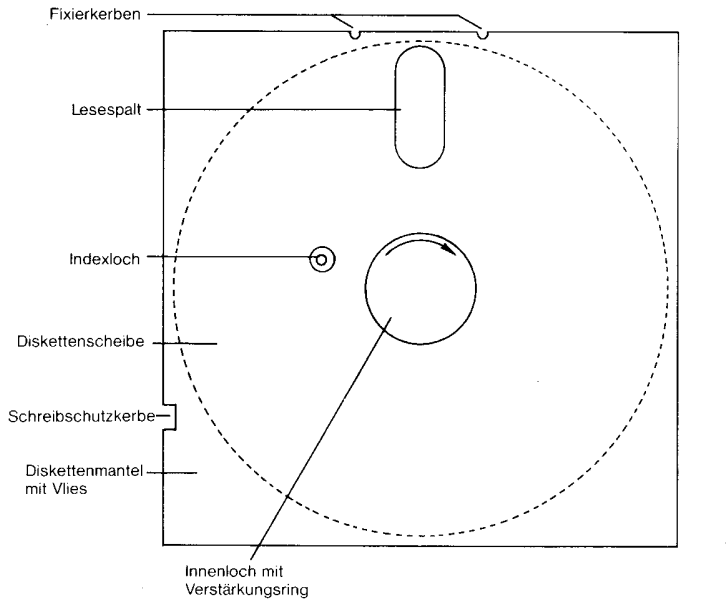


Bild 1.3 *Darstellung einer Diskette*

Eine Diskette ist eine dünne Scheibe aus magnetischem Material (ähnlich einem Tonband), die in einer schwarzen Hülle, dem Diskettenmantel, untergebracht ist. Dieser Mantel ist innen mit einem Vlies ausgelegt, um die Diskette zu schützen und eventuell eintretende Staubteilchen von der Diskettenoberfläche fernzuhalten. In der Mitte der Diskette befindet sich das Innenloch, an dem nachher der Laufwerkmotor angreift, um die Magnetscheibe zu drehen. Die Diskette dreht sich dabei mit 300 Umdrehungen in der Minute in ihrem Mantel, der noch an zwei anderen Stellen Öffnungen aufweist. Das ist einmal ein ganz kleines Loch, nahe dem großen Innenloch, das Indexloch genannt wird und auch auf der Diskettenscheibe an ei

ner Stelle vorhanden ist. Zum anderen befindet sich noch eine große, längliche Schreib-Leseöffnung in dem Diskettenmantel, an der nachher der Schreib- Lesekopf angreift, um die Daten zu schreiben und zu lesen. Die Öffnungen im Diskettenmantel zeigen immer die Magnetschicht der Diskette und sollten nie mit den Fingern berührt werden.

Schieben Sie also nun die Diskette mit der Schreib-/Leseöffnung voran in das Laufwerk, bis sie ganz hinten an den Anschlag stößt. Dabei sollten Sie ein Knicken der Diskette auf jeden Fall vermeiden. Danach schließen Sie die Klappe an der Laufwerköffnung. Bei der 1570 wird eine Klappe nach unten gedrückt; bei der 1571 wird der Drehhebel nach unten gedreht.

In beiden Fällen wird dadurch die Diskette im Laufwerk fixiert.

1.2.1 Formatieren einer Diskette

Nun wollen wir endlich unser Programm auf der Diskette abspeichern, wozu wir vorher noch eine Zusatzmaßnahme ergreifen müssen: Die Diskette wird zuerst formatiert. Mit Formatieren bezeichnet man einen Vorgang, der die Diskette in gewisser Weise überhaupt erst betriebsbereit macht. Dieser Vorgang wird später noch im einzelnen besprochen werden.

Das Formatieren geschieht je nach Computer verschieden. Haben Sie einen C64, so tippen Sie die folgende Zeile ein:

```
OPEN 15,8,15,"N: TESTDISKETTE.,TD": CLOSE 15
```

Nach Eingabe dieser Zeile drücken Sie die RETURN-Taste. Die Floppy läuft jetzt an und die LED am Laufwerk leuchtet auf. Das Formatieren der Diskette dauert jetzt etwa 90 Sekunden. Während dieser Zeit ist der Computer nicht ansprechbar.

Haben Sie einen C128, so tippen Sie entweder die oben angegebene Zeile für den C64 oder aber folgendes:

```
HEADER "TESTDISKETTE",ITD,D0,U8
```

Danach drücken Sie auch hier auf die RETURN- oder auf die ENTERTaste. Der Computer fragt Sie nun:

```
ARE YOU SURE?
```

worauf Sie mit Y (für yes) antworten sollten. Jetzt passiert das gleiche wie beim C64. nur daß das Formatieren erheblich schneller geht. Haben Sie auf dem C128 die Zeile wie für den C64 abgetippt, so unterbleibt die Frage ARE YOU SURE?; ansonsten passiert aber nichts anderes. Diese Frage des

Computers ist eine Sicherheitseinrichtung, die das unbeabsichtigte Formatieren von Disketten verhindern soll. Antworten Sie mit etwas anderem als "Y" oder YES, so geht der Computer in den READY-Modus zurück, und es passiert nichts.

Es sollte Ihnen immer klar sein, daß das Formatieren eine Diskette völlig neu magnetisiert und somit alle Daten und Programme löscht, die eventuell vorher darauf abgelegt waren. Seien Sie also mit diesem Befehl immer sehr vorsichtig, und vergewissern Sie sich vorher lieber zweimal, daß Sie auch die richtige Diskette eingelegt haben.

1.2.2 Das Inhaltsverzeichnis der Diskette

Einer der Vorgänge beim Formatieren ist das Aufbringen eines Inhaltsverzeichnisses auf die Diskette. Man kann so immer nachsehen, welche Programme oder Daten man auf der Diskette abgelegt hat. Außerdem dient das Inhaltsverzeichnis der Floppystation zum Feststellen, ob das Programm, das man in den Computer laden will, überhaupt auf der Diskette vorhanden ist, die sich gerade im Laufwerk befindet.

Das Inhaltsverzeichnis, das oft mit dem englischen Ausdruck "directory" bezeichnet wird, kann wie ein normales Programm von der Diskette geladen werden, um es sich anzusehen. Achtung! Programme, die bis dahin im Speicher standen, werden also gelöscht. Um Ihnen das zu demonstrieren, laden Sie einmal das Directory der neu formatierten Diskette in den Speicher:

```
LOAD "$",8 <RETURN>
```

<RETURN> deutet hierbei an, daß Sie nach Eingabe der Zeile die ReturnTaste am Computer drücken sollen.

Wenn sich der Computer mit READY meldet, tippen Sie

```
LIST <RETURN)
```

Es erscheint folgendes Bild auf dem Bildschirm:

```
0 "TESTDISKETTE    " TD 2A
664 BLOCKS FREE.
```

beziehungsweise:

```
0 "TESTDISKETTE    " TD 2A
1328 BLOCKS FREE.
```

wenn Sie eine Floppy 1571 an einem C 128 betreiben. Sie sehen, wie sich die zweiseitige Betriebsart der 1571 gegenüber der 1570 äußert: Sie erhalten eine Diskette mit einer doppelt so großen Anzahl von freien Blöcken (blocks free). Diese große Kapazität der 1571 wird jedoch nur auf einem C 128 ausgenutzt. Betreiben Sie diese Floppy an einem C64, so simuliert sie eine 1541 und arbeitet ausschließlich einseitig (mit 664 BLOCKS FREE). Sie werden in den Kapiteln für Fortgeschrittene jedoch noch erfahren, wie Sie auch am C64 eine große 1571 betreiben können.

Das Testprogramm, das Sie vorher eingetippt haben, wurde durch das Inhaltsverzeichnis der Diskette im Speicher überschrieben und ist somit gelöscht worden. Für unsere Testzwecke tippen Sie dieses Programm bitte noch einmal ein:

```
NEW  <RETURN>
10 REM DAS IST EIN TEST
20 REM DAS IST DAS ENDE DES TESTS
30 END
```

1.2.3 Das Speichern eines Programms auf Diskette

Wir wollen unser Programm nun einmal auf die Diskette abspeichern. Dazu geben wir dem Programm einen Namen, der aus maximal 16 Zeichen bestehen darf und der sich Filename oder Dateiname nennt. In unserem Beispiel verwenden wir den Namen TEST. Dazu tippen Sie:

```
SAVE "TEST",8 <RETURN>
```

Die Floppy läuft an, und das Programm wird abgespeichert. Zur Kontrolle sehen Sie sich danach wieder das Inhaltsverzeichnis der Diskette an:

```
LOAD "$",8 <RETURN>
LIST <RETURN>
```

Nun erhalten wir folgendes Bild:

```
0 "TESTDISKETTE    " TD 2A
1   "TEST"             PRG
663 BLOCKS FREE.
```

beziehungsweise:

```
0 "TESTDISKETTE    " TD 2A
1   "TEST"             PRG
1327 BLOCKS FREE.
```

Wie Sie sehen, erscheint der Programmname nun im Directory der Diskette. Die Zahl vor dem Programmnamen ist ein Maß für die Größe des abge

speicherten Programms. In unserem Fall beträgt sie 1 Block. (Was das ist, besprechen wir später noch.) Das Kürzel PRG steht für PROGRAMM und zeigt uns an, daß das, was wir abgespeichert haben, ein Programm ist. Wie wir später noch entdecken werden, können nämlich auch ganz andere Sachen auf einer Diskette abgelegt werden, zum Beispiel die Daten von elektronischen Notizbüchern oder von Kochrezepten.

Zum Abspeichern des Programms auf die Diskette haben wir also den SA VE-Befehl benutzt. Dieser Befehl hat folgende Syntax:

```
SAVE "filename",gerätenummer<,sekundäradresse>
```

filename ist der oben angesprochene Name, den man einem Programm geben muß, um es auf der Diskette abspeichern zu können. Dieser Name darf aus 1 bis 16 Zeichen bestehen, wobei folgende Zeichen jedoch nicht gestattet (oder nicht immer gestattet) sind: @, $, *, ?, #, :. Warum diese Zeichen nicht gestattet sind, hat seinen Grund darin, daß die Floppy mit diesen Zeichen Sonderfunktionen ausführt. Bei dem $-Zeichen müßten Sie sich aber schon denken können, warum es nicht erlaubt ist. Natürlich: Das $-Zeichen wird von der Floppy als Name für das Directory der Diskette benutzt.

gerätenummer ist eine Nummer, die der Floppy und jedem anderen, an den Computer angeschlossenen, Gerät zugeteilt wird. Das hat folgenden Grund: alle Geräte (Floppy, Drucker, Plotter, ...) hängen beim Computer an ein und derselben Steckbuchse. Diese Verbindung zum Computer bezeichnet man allgemein als Bus, weil auf dieser Leitung, bildlich gesprochen, Daten zwischen dem Computer und den angeschlossenen Geräten "hin- und herfahren". Wenn der Computer nun aber Daten zur Floppy "fahren" will, wie sollen dann die an den Bus angeschlossenen Geräte wissen, welches gemeint ist? Aus diesem Grund existiert die sogenannte Geräteadresse oder Gerätenummer. Bevor der Computer die Daten auf den Bus schickt, sendet er zuallererst einmal eine Nummer. Anhand dieser Nummer "weiß" jetzt das angeschlossene Gerät mit eben dieser Nummer, daß es gemeint ist, und macht sich für den Empfang von Daten oder Befehlen bereit. Die übrigen angeschlossenen Geräte ignorieren jetzt den Busbetrieb und geben die Leitungen für die Übertragung frei. Eine Floppy hat bei Commodore in der Regel die Nummer 8, ein ein Plotter die 5 und Drucker die Nummer 4. Deshalb haben wir auch immer die Gerätenummer 8 angegeben, wenn wir bis

her mit der Floppy kommuniziert haben. Die Gerätenummer kann bei der 1570/71 auch geändert werden. Auf dieses Problem werden wir in Abschnitt 1.2.7 eingehen; außerdem wird diese Maßnahme natürlich erst dann notwendig, wenn Sie mehr als eine Floppystation an den Bus des Computers anschließen.

<sekundäradresse>

Die Angabe dieses Parameters ist optional. Aus diesem Grund steht er in "<>", die natürlich beim Eintippen weggelassen werden müssen. Eine Sekundäradresse ist ein Wert zwischen 0 und 255, der bei Bedarf mit auf den Bus gesendet werden kann, um das angeschlossene Gerät in eine bestimmte Betriebsart zu versetzen oder Parameter eines Peripheriegeräts zu verändern. Beim Drucker kann beispielsweise mit Sekundäradressen zwischen normalem Druck und tabellarischem Druck umgeschaltet werden. Wird die Sekundäradresse weggelassen, setzt der Computer automatisch die Null. Die Verwendung dieses Wertes wird im Laufe der Arbeit mit der Floppy noch klarer werden. An dieser Stelle sei nur erwähnt, daß Sie die Sekundäradresse bei LOAD oder SAVE generell weglassen können, wenn Sie mit BASIC-Programmen arbeiten, die in den BASIC-Speicher geschrieben werden sollen (also wie in unserem Beispiel). Arbeiten Sie hingegen mit Maschinenprogrammen, die nicht im BASIC-Speicher des Computers sondern an irgendeiner anderen Stelle im Speicher stehen, so ist die Angabe der Sekundäradresse 1 unerläßlich. Prinzipiell sollten Sie sich sogar angewöhnen, immer mit der Sekundäradresse I zu arbeiten; das ist auf jeden Fall nie verkehrt (aber unbequemer, da mehr Tipparbeit zu leisten ist)!

1.2.4 Das Laden eines Programms von der Diskette

Im letzten Abschnitt haben wir besprochen, wie ein Programm auf die Diskette geschrieben werden kann. Wie bekommt man es aber wieder zurück?
Den LOAD-Befehl haben Sie schon kennengelernt, wir wollen ihn dennoch einmal im Detail betrachten.
Wenn Sie Ihr Programm wieder in den Speicher des Computers laden wollen, so tippen Sie:

```
LOAD "TEST",8 <RETURN>
```

oder

```
LOAD "TEST",8,1 <RETURN>
```

Was die zweite Möglichkeit angeht, so haben wir ja im letzten Abschnitt erfahren, welchen Sinn die Angabe der Sekundäradresse hat. Dieser Sinn betrifft eigentlich nur den LOAD-Befehl, da der einfache SAVE-Befehl sowieso nur den BASIC-Speicherbereich auf der Diskette ablegt. In unserem Fall und in allen weiteren Fällen gewöhnen Sie sich also beim Laden das obligatorische ,8,1 an. Bei SAVE reicht in der Regel immer ,8.

Jetzt tippen Sie LIST, und es erscheint:

```
10 REM DAS IST EIN TEST
20 REM DAS IST DAS ENDE DES TESTS
30 END
```

1.2.5 Fehler beim Diskettenbetrieb

Wie Sie gesehen haben, ist der Floppybetrieb denkbar einfach. Programme werden mit SAVE abgespeichert und mit LOAD wieder in den Computer geladen. Auf einer Diskette haben eine Menge Programme Platz. Maximal 144 sind es, je nach Länge der einzelnen Programme. Es existieren ja maximal 664 beziehungsweise 1328 freie Blöcke für Programme.

Beim Floppybetrieb können aber natürlich auch Pannen passieren, weil zum Beispiel ein Programm nicht mit dem Namen auf der Diskette steht, mit dem Sie es laden wollen, oder weil zum Beispiel die eingelegte Diskette defekt ist. In solchen Fällen hält die Floppy Fehlermeldungen bereit, die sich in einem mehr oder minder hektischen Blinken der LED am Laufwerk äußern. Die Leuchtdiode blinkt solange weiter, bis ein neuer fehlerfreier Befehl an die Floppy geschickt wird.

Die Floppy hat leider keinen Bildschirm wie der Computer, so daß Sie mit dem Blinken auf Fehler hinweisen muß. Was aber viele Anwender nicht wissen, ist, daß die Floppy ein vollständiger eigener Computer ist. der eine Unmenge an Fehlermeldungen besitzt. So wie der C64 oder der C128 einen SYNTAX ERROR bei Fehlbedienung ausgibt, so kennt auch die 1570/71 einen SYNTAX ERROR, wenn beispielsweise ein Befehl nicht korrekt eingetippt wurde. Diese Fehlermeldungen werden wir im zweiten Kapitel noch ausführlicher besprechen.

Wichtig ist jetzt Wenn sich die Floppy beim Laden oder Speichern von Programmen "komisch benimmt", das heißt, wenn zum Beispiel auf einmal

ein lautes Rattern hörbar wird und die LED am Laufwerk nicht mehr konstant leuchtet, sondern zuerst unregelmäßig flackert und dann in ein konstantes Blinken übergeht, so ist ein Fehler aufgetreten, der alle möglichen Ursachen haben kann - auch eine defekte Diskette. Um das Risiko solcher Fehler möglichst gering zu halten, kann man nicht oft genug betonen, wie sorgsam man mit Disketten umgehen sollte. Auch sollten Sie von wichtigen Programmen stets noch auf einer zweiten Diskette eine Sicherheitskopie (backup) anfertigen, die genau das gleiche Programm enthält, und diese Sicherheitskopie an einem sicheren Ort verwahren. Fällt nämlich die eine Diskette aus, so können Sie auf Ihre Reserve zurückgreifen und ersparen sich unter Umständen Ärger und Tipparbeit.

Das Fehlerblinken kann aber auch auftreten, ohne daß ein Rattern hörbar wird oder daß die Floppy überhaupt anläuft. In diesem Fall haben Sie meistens eine falsche Eingabe gemacht, oder Sie haben versucht, auf eine schon volle Diskette noch ein Programm zu schreiben. Konnte ein zu langes Programm nicht mehr vollständig auf die Diskette geschrieben werden, so wird das im Directory zum Beispiel folgendermaßen sichtbar:

```
0 "TESTDISKETTE    " TD 2A
1   "TEST"              PRG
2   "NOCH EIN TEST"     PRG
0   "ZU LANGER TEST"   *PRG
0 BLOCKS FREE.
```

Wie Sie sehen, stehen auf dieser Diskette ordnungsgemäß zwei Programme, nämlich TEST und NOCH EIN TEST. Das dritte Programm (ZU LANGER TEST), das auf die Diskette geschrieben werden sollte, hat die Diskettenkapazität überschritten. Als Kennzeichen dafür blinkt die LED der Floppy und vor dem Kürzel PRG steht ein "*". Ein weiteres sicheres Zeichen für diesen Fehler sind die Blockangaben 0 BLOCKS FREE. und Null für die Programmlänge. Wie man einen solchen "faux pas" wieder behebt, denn schließlich ist im Augenblick die ganze Diskette unbrauchbar, das besprechen wir ausführlich im 4. Kapitel.

Sie sollten sich jetzt schon an den Gedanken gewöhnen, daß wir es bei der 1570 und der 1571 nicht nur mit einfachen Diskettenlaufwerken zu tun haben. Es handelt sich vielmehr um eigene kleine Computer, die fast alle Arbeiten selbständig ausführen können, ohne daß dafür der C64 oder der Cl28 benötigt würden. Auch verfügen diese Floppystationen über einen erstaunlichen Befehlsvorrat, was den Diskettenbetrieb betrifft.

1.2.6 Testen auf korrektes Abspeichern vom Programmen

Dieser Abschnitt befaßt sich mit dem VERIFY-Befehl, der allerdings seine Bedeutung bei einer Floppystation fast verloren hat. Vielleicht gehören auch Sie zu denjenigen Personen, die schon einmal mit einem Kassettenrecorder Programme abgespeichert und geladen haben. Dann wissen Sie auch, daß der Kassettenbetrieb eine relativ anfällige Angelegenheit ist, bei der öfter Pannen auftreten (zum Beispiel wegen eine verstellten Tonkopfes).

Für diesen Betrieb ist der VERIFY - Befehl unerläßlich, da man mit ihm nachprüfen kann, ob ein Programm korrekt abgespeichert wurde. Der Computer vergleicht dabei das abgespeicherte Programm direkt mit dem Original im Speicher.

Für den Betrieb einer Floppystation wird dieser Befehl eigentlich nicht mehr benötigt, da die Diskettenspeicherung zum einen eine erheblich sicherere Aufzeichnungsmethode darstellt, als eine Recorderaufzeichnung und zum anderen die 1570/71 ein automatisches Verify durchführt, das bei jedem SAVE-Befehl jeden geschriebenen Block noch einmal Korrektur liest, bevor der nächste geschrieben wird.

Angewendet wird der VERIFY-Befehl wie der LOAD-Befehl:

```
VERIFY"TEST",8 <RETURN>
```

Auch die anzugebenden Parameter sind mit denen des LOAD-Befehls identisch (siehe 1.2.4).

Ist das abgespeicherte Programm mit dem im Computerspeicher identisch, so meldet sich der Computer mit OK, wenn der Vorgang abgeschlossen ist; andernfalls ist ein VERIFY ERROR die Folge.

1.2.7 Das Ändern der Gerätenummer der 1570/71

Wie wir schon erfahren haben, sendet der Computer bei einem Zugriff auf die Floppy eine Nummer auf den Bus, die das Gerät anspricht. Bei unserer Floppy ist das in der Regel die Nummer 8, die normalerweise vom Werk voreingestellt ist.

Nun kann es aber passieren, daß wir uns eine zweite Floppystation anschaffen und diese gleichzeitig mit der ersten betreiben wollen. In diesem Fall gäbe es ein Chaos, wenn der Computer ein Gerät mit der Nummer 8 ansprechen wollte und es würden zwei Geräte mit der Nummer 8 antworten.

Für diesen Zweck kann man die Gerätenummern der Floppystationen zwischen 8 und 11 beliebig verstellen, wobei der Eingriff bei der 1571 problemlos zu bewältigen ist, während Sie bei der 1570 unter Umständen Ihre Garantie verlieren.

Zunächst zur 1571: Hier befinden sich auf der Rückseite (siehe Bild 1.1) Zwei winzige DIP-Schalter, die mit einem spitzen Gegenstand erreicht werden können. Diese beiden Schalter stehen in der Regel beide oben. Wenn Sie die Gerätenummer umstellen wollen, schalten Sie die Floppy aus und stellen sich Ihre Nummer wie folgt ein:

Schalter links	Schalter rechts	Nummer
oben	oben	8
unten	oben	9
oben	unten	10
unten	unten	11

Tabelle 1.1 Schalter für Gerätenummer

Der linke Schalter trägt dabei auch die Nummer 1 und der rechte Schalter die Nummer 2.

Bei der 1570 müssen Sie von Ihrer Floppystation den Gehäusedeckel abnehmen, um an die Schalter heranzukommen. Dazu schalten Sie die Floppystation aus und ziehen den Netzstecker heraus!

Lösen Sie die vier Befestigungsschrauben auf der Unterseite der Floppy und heben den Deckel ab. Wenn Sie die Floppy von der Frontseite betrachten, sehen Sie die große viereckige Platine, auf deren rechter Seite die beiden Schalter angebracht sind. Die Einstellung dieser Schalter ist entsprechend Tabelle 1.1 vorzunehmen.

Auf diese Weise können Sie also bis zu vier Floppystationen mit den Nummern 8 bis 11 an Ihren Computer anschließen. Wenn Sie auf eine Floppystation mit der Nummer 9 zugreifen, dann heißt der Befehl zum Laden des Directory natürlich nicht mehr `LOAD "$",8`, sondern `LOAD "$",9`.

Die weiteren Beispiele in diesem Buch beziehen sich in der Regel auf eine Floppystation mit der Nummer 8.

1.2.8 Das Arbeiten mit der Floppy unter BASIC 7.0

Wir haben uns bisher mit den Standard befehlen des BASIC 2.0 des Commodore 64 beschäftigt. Der Commodore 128 enthält aber das viel leistungsfähigere BASIC 7.0, das auch den Diskettenbetrieb stark vereinfacht.

Da das BASIC 7.0 für fast jede Floppyanwendung einen eigenen Befehl zur Verfügung stellt, wollen wir im 2. Kapitel bei der Besprechung der Floppybefehle immer beide Möglichkeiten der Befehlsübergabe betrachten. Dort wird also jeweils die Methodik unter BASIC 2.0, unter BASIC 7.0 und gegebenenfalls auch für den eingebauten Monitor im Commodore 128 beschrieben. Für C64-Besitzer ist jedoch nur das BASIC 2.0 von Bedeutung; es sei denn, sie verfügen über eine eigene Befehlserweiterung, die die Befehle des BASIC 4.0 enthält. Dieses BASIC stammt von den CBM-Computern der Serien 4000 und 8000 und stellt die leistungsfähigen Floppybefehle ebenfalls zur Verfügung (ausgenommen im Monitor). Das 2. Kapitel wird also ein wenig tabellarisch aufgebaut und erlaubt so, für jedes Computersystem die richtigen Befehle auszuwählen. Wenn im weiteren Verlauf dieses Buches Programme verwendet werden, die sowohl auf dem C64 als auch auf dem C128 lauffähig sind, so wird stets das BASIC 2.0 für deren Erstellung benutzt. Programme, die nur auf dem Commodore 128 lauffähig sind, werden je nach Problemstellung entweder im BASIC 2.0 oder 7.0 geschrieben.

Achtung! Dieses Buch behandelt die Floppystationen 1570 und 1571 parallel, da sich beide nur in geringer Weise unterscheiden. Sind Unterschiede vorhanden, so wird an entsprechender Stelle darauf hingewiesen. Generell gilt jedoch: Jeder Befehl, der sich auf zwei Diskettenseiten bezieht beziehungsweise die Wahl zwischen der einen (Seite 0) oder der anderen (Seite 1) einer Diskette offenläßt, gilt nur für die 1571. Bei der 1570 muß dann immer Seite 0 gewählt werden; ansonsten ist eine Fehlermeldung der Floppystationen die Folge.

2 Der Befehlssatz der 1570/1571

Das folgende Kapitel beschreibt die grundlegenden Befehle der 1570/71 für den Diskettenbetrieb. Es wird dabei auf die verschiedenen Computersysteme und Anwendungen Rücksicht genommen, so daß jeder Anwender die für ihn wichtigen Daten auswählen kann.
Sie können auf Daten stoßen, die Ihnen noch unverständlich sind, weil diese etwa den eingebauten Maschinensprache-Monitor betreffen, den Sie unter Umständen noch nie benutzt haben. Wichtig sind für jeden Anwender erst einmal nur die Informationen, die ihn bei seinen eigenen Problemen unterstützen. Die anderen Angaben werden ihm später vielleicht eine wertvolle Hilfe sein.
Ein paar der Befehle, die hier aufgeführt werden, haben Sie schon kennengelernt. Dieses Kapitel soll jedoch die Rolle eines Nachschlagewerkes übernehmen und Ihnen die Befehle übersichtlich darstellen, mit denen Sie am meisten arbeiten werden.

2.1 Erläuterungen zu den Befehlsbeschreibungen

Im BASIC 2.0 müssen Sie den Befehl direkt über den seriellen Bus an die Floppy schicken. Dazu öffnen Sie einen Kanal mit dem OPEN-Befehl, senden Ihr Kommando mit PRINT# an die Floppy und schließen den Kanal mit CLOSE. Was ein Kanal ist und was eigentlich passiert, wenn Sie einen Befehl senden, wird später erläutert. Wichtig ist erst einmal, wie ein Befehl unter BASIC 2.0 an die Floppy gesendet wird, und das geht folgendermaßen:

```
OPEN 1,8,15      <RETURN>
PRINT#1,"...befehL" <RETURN>
CLOSE 1    <RETURN>
```

Also zum *Beispiel*:

```
OPEN 1,8,15      <RETURN>
PRINT#1, "N:TESTDISKETTE,TD"     <RETURN>
CLOSE 1    <RETURN>
```

Wenn Sie die Zeile mit PRINT# eingetippt haben, wird die Floppy in den meisten Fällen anlaufen und den Befehl ausführen. Tippen Sie dann direkt CLOSE ein, so ist der Computer solange "weg", bis die Floppy den Befehl

beendet hat. In der relativ langen Wartezeit (zum Beispiel beim Formatieren) können vom Computer andere Befehle ausgeführt werden, die nichts mit der Floppy zu tun haben. Diese Arbeitsweise zeigt recht eindrucksvoll, daß der Computer (nur bei BASIC 2.0; bei BASIC 7.0 wartet er automatisch auf das Ende des Befehls bei der Floppy) nichts mit dem Formatieren einer Diskette und auch einigen anderen Floppybefehlen zu tun hat. Die Floppy arbeitet hierbei vollkommen selbständig.
Im allgemeinen haben Basicbefehle, die die Floppy betreffen, mehrere Parameter. Für diese Parameter werden in den Erläuterungen Worte oder Abkürzungen benutzt, die folgende Bedeutungen haben:

Name	steht für einen Namen (I bis 16 Zeichen)
X	steht für ein (beliebiges) Zeichen
file#	steht für eine Dateinummer (I bis 127)
geräte#	steht für die Gerätenummer (in der Regel 8)
sekundär#	steht für die Sekundäradresse (0 bis 255)
drive#	steht für die Laufwerknummer (immer 0)

Diese Wörter beginnen stets mit kleinen Buchstaben. Steht davor noch ein weiteres Zeichen (zum Beispiel: Ugeräte#), so ist das kein Druckfehler: Dieses Zeichen *muß* eingetippt werden. Nachfolgend eine Erklärung solcher Zeichen:

U Dieses U steht für des englische Wort "unit" und ist im BASIC 7.0 allen Angaben der Gerätenummer vorangestellt. Es existiert also nur die Kombination Ugeräte#, wobei Sie dann zum Beispiel U8 für die Geräte nummer 8 der Floppy eingeben müssen.

D Das D steht für "drive" und bezeichnet die Laufwerknummer, die wir bei der Floppy ansprechen wollen. In unserem Fall ist diese Nummer immer Null, das heißt, bei der Kombination Ddrive# kann nur D0 gesetzt werden; es sei denn, Sie betreiben an Ihrem Cl28 ein DoppelLaufwerk. Auch das D gibt es nur im BASIC 7.0.

I Das I steht für "id" oder "identification" und teilt dem Computer mit, daß die beiden folgenden Buchstaben als ID für das Formatieren stehen. Dieses I kommt demnach nur beim HEADER-Befehl vor.

Diese Abkürzungen und englischen Ausdrücke sollten Sie sich einprägen. Die Computersprache schlechthin ist nun einmal Englisch, und man wird kaum darum herumkommen, sie teilweise auch anzuwenden.

2.2 Der Befehl zum Formatieren einer Diskette

Diesen Befehl haben Sie schon im ersten Kapitel kennengelernt. Er hat an dieser Stelle im Buch eigentlich gar nichts verloren, da es sich hier um einen der Systembefehle der Floppystation handelt. Er ist jedoch von so entscheidender Wichtigkeit, daß er in dieser Aufstellung nicht fehlen soll. Die restlichen Systembefehle der 1570/71 werden später dann getrennt erläutert.

Das Formatieren einer Diskette ist der erste wichtige Zugriff, nachdem eine neue Diskette aus ihrer Packung geholt wird. Dieser Vorgang sorgt dafür, daß die Diskette für die Floppystation überhaupt erst lesbar wird, bevor wir unser Programm darauf speichern können. Beim Formatieren wird unter anderem auch das Inhaltsverzeichnis einer Diskette angelegt, das Sie immer über die abgespeicherten Daten und Programme informiert.

Der Formatierungs-Befehl hat den Namen NEW oder N unter BASIC 2.0 und wird unter BASIC 7.0 mit der Anweisung HEADER aufgerufen.

BASIC 7.0	HEADER "name" ,Ixx, U geräte#,Ddrive#
BASIC 2.0	"Ndrive#:name,xx"
Monitor	@geräte#,Ndrive#:name,xx

Beispiele:

```
HEADER "TESTDISKETTE",ITD,U8,D0
OPEN 1,8,15
PRINT#1,"N0:TESTDISKETTE,TD"
CLOSE 1
@8,N0:TESTDISKETTE,TD
```

Oben wurde unter BASIC 2.0 nur der zur Floppy gesendete Befehl dargestellt. Die Anweisungen an den Computer, um diesen Befehl an die Floppy zu übertragen, sind dort weggelassen worden. Die Beispiele zeigen Ihnen die Verwendung der Angaben noch einmal etwas praxisnäher.

Die beiden xx in der Befehlstabelle stellen Platzhalter für die Angabe einer Diskettenidentifikation (kurz ID) dar. Diese ID muß für jede neu zu formatierende Diskette angegeben werden und erlaubt die Unterscheidung der Disketten noch über deren Namen hinaus, zum Beispiel durch Titel wie TESTDISKETTE,01, TESTDISKETTE,02, usw..

Wird die ID bei der Formatierung weggelassen, so erzielt man damit einen Effekt, der auch als "kurze Formatierung" bekannt ist. Durch die kurze Formatierung wird lediglich das Directory der Diskette gelöscht. Das geht natürlich erheblich schneller, als wenn die Diskette insgesamt neu formatiert wird. Das funktioniert jedoch nur dann, wenn die Diskette schon einmal auf der 1570/71 "lang" formatiert wurde. Als Ergebnis der kurzen Formatierung erhält man, wie bei der langen Version, ein leeres Directory mit dem neuen Namen. Alle Blöcke sind wieder frei.

Ein anderes Problem stellt sich im BASIC 7.0, jedoch in Form eines Fehlers des BASIC-Interpreters. Es gibt BASIC-Anweisungen, die nur 2 Zeichen lang sind, beziehungsweise mit 2 Zeichen abgekürzt werden können (IF, ON, GO, Po, Pr,...). Der BASIC-Interpreter versucht nur dann nicht, Zeichen als Anweisungen zu erkennen, wenn sie in Anführungszeichen "" oder hinter dem REM-Befehl stehen.

Wenn Sie nun den HEADER-Befehl verwenden, müssen Sie bei der Angabe der ID auf diese Eigenart des Interpreters Rücksicht nehmen. Die folgende Zeile ruft zum Beispiel einen SYNTAX ERROR hervor:

```
HEADER "TESTDISKETTE",IFE,U8,D0
```

da das I (für ID) zusammen mit dem nachfolgenden F vom Computer als ein IF-Befehl erkannt wird.

Wollen Sie eine solche ID angeben, so kann der Umweg über das BASIC 2.0 helfen. Hier steht der Befehl in Anführungszeichen und Sie können prinzipiell jedes vorhandene Zeichen als ID verwenden; sogar Cursor-Steuerzeichen.

Haben Sie bei BASIC 7.0 alles richtig gemacht, dann kommt die Sicherheitsnachfrage vom Computer: `ARE YOU SURE?`

Diese im BASIC 2.0 fehlende Einrichtung wird bei allen diskettengefährdenden Befehlen verwendet, die unter Umständen wichtige Daten zerstören können. Nach dieser Frage wird nur bei der ausdrücklichen Angabe von Y oder YES weitergemacht, sonst wird abgebrochen. Bei Verwendung des Header-Befehls innerhalb eines Basic-Programms wird diese Abfrage nicht durchgeführt.

Wird das Formatieren ordnungsgemäß beendet, dann meldet sich der Computer wieder mit READY. Ansonsten erfolgt bei BASIC 7.0 die Ausgabe der Meldung BAD DISK und das Formatieren wird abgebrochen. Im BASIC 2.0 bleibt die Floppy mit einer blinkenden LED stehen. Beiden Fälle bedeuten das gleiche: Die Diskette, die formatiert werden sollte, ist nicht in

Ordnung. Ändert sich dieser Sachverhalt nicht nach zwei weiteren Versuchen, so ist entweder der Schreib-/Lesekopf der Floppy verstellt oder sie können die Diskette wegwerfen, da sie einen Herstellungsfehler aufweist. Im ersten Fall sollten Sie sich mit weiteren Versuchen an anderen Disketten von der Richtigkeit der Annahme überzeugen.

Die Befehlsangabe für den Monitor entspricht von ihrer Verarbeitung her immer der für BASIC 2.0. Nur BASIC 7.0 ist so komfortabel, daß es über eine spezielle Diskettenbehandlung außer dem üblichen Datenaustausch verfügt (Umsetzen der Fehlermeldungen der Floppy, Sicherheitsabfragen, Auslesen des Fehlerkanals, usw.).

2.3 Das Inhaltsverzeichnis der Diskette

Auch das Directory der Diskette haben wir uns schon einmal angesehen. Hier gibt es jedoch grundlegende Unterschiede zwischen BASIC 2.0 und BASIC 7.0. In der Tabelle wird dies deutlich:

BASIC 7.0	DIRECTORY Ddrive# ON Ugeräte#, "x...x" CATALOG Ddrive# ON Ugeräte#, "x...x"
BASIC 2.0	LOAD "$drive#:x...x",geräte#
Monitor	@geräte#,$drive#:x...x

Beispiele:

```
DIRECTORY
DIRECTORY D0 ON U8
DIRECTORY D0 ON U9, "*=p"
DIRECTORY U8
DIRECTORY "TEST=U"
LOAD "$0",8
LOAD "$0:*=S",8
LOAD "$0",9
@8,$
@8,$0:TEST
```

In BASIC 2.0 wird das Inhaltsverzeichnis wie ein Programm (siehe 1.2.2) in den BASIC-Speicher geladen und durch LIST angesehen. Alle im Speicher vorhandenen BASIC-Programme werden dadurch jeweils gelöscht. Diese sehr ärgerliche Tatsache kann man nur durch ein Programm beheben oder sich eine BASIC-Erweiterung zulegen.

Im BASIC 7.0 existieren dazu zwei Befehle, die beide jedoch prinzipiell das gleiche tun. Nun, warum denn zwei Befehle? Hier hat Commodore darauf Rücksicht genommen, daß es Anwender aus sehr unterschiedlichen Kreisen gibt. Die einen nennen das Inhaltsverzeichnis einer Diskette Directory, die anderen sprechen von einem Catalog; beide meinen jedoch das gleiche, und so ist auch die Syntax dieser beiden Befehle gleich.

Hier wird das Directory nicht, wie bei LOAD, in den Speicher des Computers geladen, sondern lediglich auf dem Bildschirm angezeigt, wobei keine Programme oder Daten verlorengehen. Der Vorteil liegt klar auf der Hand: Sie können nach dem Schreiben eines Programms erst einmal auf verschiedenen Disketten nachsehen, ob Sie überhaupt noch Platz darauf haben und das Programm danach abspeichern. Versuchen Sie so etwas einmal bei BASIC 2.0 (lieber nicht!).

Der Befehl DIRECTORY ist so nützlich, daß der Cl28 nach dem Einschalten automatisch die Funktionstaste F3 damit belegt hat. Es genügt also ein Tastendruck, und schon erscheint das Directory auf dem Bildschirm - ohne Programmverlust.

Hier sind der Vollständigkeit halber noch einmal ein paar Informationen zum Aussehen eines Directory.

In der ersten Zeile steht nach der Laufwerknummer (bei Einzellaufwerken immer 0) der Diskettenname, den Sie beim Formatieren festgelegt haben. Dieser Name steht in Anführungszeichen und ist invers (dunkel auf hellem Grund) dargestellt. Danach folgt die zweistellige ID der Diskette, wiederum gefolgt von zwei Buchstaben (2A). Diese letzten zwei Buchstaben geben das Diskettenformat an, damit man feststellen kann, auf welcher Commodore-Floppy die Diskette formatiert wurde (die Commodore 8050 hat beispielsweise die Kennung 2C).

Danach folgen untereinander die Dateieinträge. Sie beginnen jeweils mit einer Zahl als Längenangabe. Danach kommt der Programmname und am Ende der Zeile ein Kürzel für die Art der Daten, die auf der Diskette stehen. Für Programme haben wir das Kürzel schon kennengelernt: PRG. Insgesamt gibt es davon fünf verschiedene:

PRG	für Programmdateien
SEQ	für sequentielle Datenfiles
USR	für Programmdatenfiles
REL	für relative Datenfiles
DEL	für gelöschte Dateien (wird normalerweise nicht angezeigt)

Am Ende des Directory steht jeweils eine Mitteilung, die angibt, wieviel Platz auf der Diskette noch frei ist. Maximal sind das 664 (oder 1328) Diskettenblöcke, wobei wir noch erfahren werden, was ein Block im einzelnen ist.

Wie Sie in der Tabelle gesehen haben und auch aus den Beispielen sehen können, kann man dem DIRECTORY-Befehl (oder CATALOG-Befehl; das ist egal) noch eine Zeichenkette in Anführungszeichen folgen lassen. Das "x...x" soll dabei ausdrücken, daß es sich um eine Zeichenkette variabler Länge handeln kann. Diese Zusätze erlauben ein selektives Anzeigen des Directory. Wie Sie wissen, können auch noch andere Daten, außer Programmen, auf einer Diskette abgelegt werden. Mit dieser Methode ist es nun möglich, zum Beispiel nur die Programme auf der Diskette anzuzeigen oder nur die Datenfelder, die mit den Buchstaben DATEN beginnen. Wie das im einzelnen funktioniert, das müssen wir später etwas genauer besprechen, da hier noch einige Sonderfunktionen der Floppy zu beachten sind.

2.4 Laden eines Programms

Dieser Befehl erklärt sich eigentlich von selbst. Hier ist die Befehlssyntax:

BASIC 7.0	DLOAD "name" ,Ddrive#,Ugeräte#
	RUN "name",Ddrive#,Ugeräte#
BASIC 2.0	LOAD "drive#:name",geräte#,sekundär#
Monitor	L "drive#:name",geräte#,xxxx
	xxxx gibt hierbei eine hexadezimale Startadresse an, ab der das Programm geladen werden soll.

Beispiele:

```
DLOAD "TEST",D0,U8
DLOAD "PROGRAMM"
RUN "AUTOSTART",U9
LOAD "BASIC-PROGRAMM",8
LOAD "0:MASCHINENPRG",8,1
L "ASSEMBLER",8
L "0:BILD",8,12000
```

Der prinzipielle Unterschied zwischen RUN und DLOAD besteht darin, daß der RUN-Befehl das Programm nach dem Laden direkt startet; der DLOAD-Befehl jedoch nicht.

Wie Sie sehen, kann man im BASIC 7.0 bei DLOAD und RUN keine Sekundäradresse angeben, wie das beim LOAD-Befehl möglich ist.

Wenn der Computer ein Programm auf die Diskette schreibt, dann fügt er außerdem die Anfangsadresse des Programms im Speicher dazu. Für ein BASIC- Programm ist das in der Regel 2049 ($0801) beim C64 oder 7169 ($lC01) beim C128. Arbeiten Sie jedoch auch im Maschinensprachemonitor, so wollen Sie vielleicht einmal den Speicherbereich von $2000 bis $3000 abspeichern. In diesem Fall haben Sie also eine andere Startadresse, als ein BASIC-Programm.

Wenn Sie im BASIC 2.0 keine Sekundäradresse oder die Null angeben, so wird jedes direkt von der Diskette geladene Programm an den BASIC-Start gebracht - egal, wo es vor dem Abspeichern stand. Im ersten Kapitel erfuhren Sie, warum Sie beim Laden eines Programms generell die Sekundäradresse 1 angeben sollten. Hier wird ein Programm an den Platz im Speicher eingeladen, den es auch beim Abspeichern innehatte, das heißt, der Computer richtet sich jetzt nach der angegebenen Startadresse des Programms. Für ein BASIC-Programm ist diese immer $0801 bzw. $1c01; in unserem Beispiel mit dem Maschinensprache-Monitor ist sie $2000.

In BASIC 7.0 können Sie diese Sekundäradresse beim DLOAD oder RUN-Befehl, wie gesagt, nicht angeben. Hier gibt es für das Laden von Programmen in andere Speicherbereiche einen speziellen Befehl (BLOAD), den wir in Kapitel 2.7 besprechen werden.

2.5 Speichern eines Programms

Den SAVE-Befehl haben wir ebenfalls schon kennengelernt. Hier ist seine Syntax:

BASIC 7.0	DSAVE "name",Ddrive#,Ugeräte#
BASIC 2.0	SAVE "drive#:name",geräte#,sekundär#
Monitor	S "drive#:name",geräte#,xxxx,yyyy xxxx gibt hierbei eine hexadezimale Startadresse und yyyy die hexadezimale Endadresse plus 1 des Speicherbereichs an, der abgespeichert werden soll.

Beispiele:

```
DSAVE "TEST",D0,U8
DSAVE "OHNE PARAMETER"
DSAVE "FLOPPY NR.9",U9
SAVE "0:TEST",8,1
SAVE "IN BASIC 2.0",8
S "0:6502 ASS",08,0801,6000
```

Die Sekundäradresse bei SAVE ist im Gegensatz zu LOAD ohne Belang und kann angegeben werden oder auch nicht. Wenn wir vom Monitor einmal absehen, der ohnehin für die Behandlung spezieller Speicherbereiche zuständig ist, können mit diesen Befehlen nur BASIC-Programme abgespeichert werden. Will man ein BASIC-Programm im Monitor abspeichern, so ist nur die Startadresse gleich $1C01 zu setzen (BASIC-Speicher des C128; der C64 besitzt ja keinen eingebauten Monitor). Das Ende hängt von der Länge des BASIC-Programms ab.

Bei Maschinenprogrammen, die von BASIC aus abgespeichert werden sollen, gibt es in BASIC 2.0 überhaupt keine Möglichkeit (ohne spezielle Tricks). In BASIC 7.0 gibt es dafür den BSAVE-Befehl, den wir im Kapitel 2.8 besprechen.

2.6 Prüfen auf fehlerfreies Abspeichern

Der VERIFY-Befehl wird bei einer Floppy zwar nicht benötigt, der Vollständigkeit halber ist hier jedoch seine Syntax:

BASIC 7.0	DVERIFY "name",Ddrive#,Ugeräte#
BASIC 2.0	VERIFY "drive#:name",geräte#,sekundär#
Monitor	V "drive#:name",geräte#

Beispiele:

```
DVERIFY "FEHLER?",D0,U8
DVERIFY "TEST"
VERIFY "PRÜFUNG",8
V "AUCH IM MONITOR",8
```

Dieser Befehl dient der Prüfung, ob ein Programm richtig abgespeichert wurde. Hierbei wird das Original im Speicher direkt mit dem Disketteninhalt verglichen. Ist alles on Ordnung, wird OK ausgegeben; ansonsten ist ein VERIFY ERROR die Folge.

Der Befehl stellt eigentlich ein Relikt aus der Zeit der Datasette dar, bei der eine Datenspeicherung oft mit sehr hoher Fehleranfälligkeit verbunden war. Floppystationen benötigen den Befehl nicht mehr, da sie erstens sehr sicher aufzeichnen und zweitens bei jedem Speichervorgang ein automatisches VERIFY durchführen.

2.7 Laden eines Maschinenprogramms

Dieser Befehl behandelt das Laden von Programmen, die nicht im Speicherbereich des BASIC-Interpreters stehen.

BASIC 7.0	BLOAD "name",Ddrive#,Ugeräte#,ON Bxx,Pyyyyy xx bedeutet die Nummer der Speicherbank, in die das Programm geladen werden soll (0 bis 15). yyyyy gibt die dezimale Adresse an, ab der das Programm abgelegt werden soll (0 bis 65535).
BASIC 2.0	LOAD "drive#:name",geräte#,l
Monitor	L "drive#:name",geräte#,xxxxx xxxxx gibt hierbei eine hexadezimale Adresse an, ab der geladen werden soll.

Beispiele:

```
BLOAD "ASSEMBLER",D0,U8,ON B1,P2049
BLOAD "SOURCE CODE",ON B1,P8192
LOAD "MASCHINE",9,1
L "0:TEST",0,008001
```

Den BLOAD-Befehl gibt es in dieser Form nicht im BASIC 2.0. Er erlaubt das Laden von Programmen an eine beliebige andere Stelle als die Originaladresse, mit der sie abgespeichert wurden.

2.8 Speichern eines Maschinenprogramms

Analog zum BLOAD-Befehl gibt es BSAVE, um Graphikbilder oder Maschinenprogramme sogar von BASIC aus abspeichern zu können.

BASIC 7.0 BSAVE "name",Ddrive#,Ugeräte#,ON Bxx,Pyyyyy TO Pzzzzz

xx steht für die Nummer der Speicherbank (0 - 15) und yyyyy und zzzzz jeweils für Startadresse und Endadresse plus 1 des Programms (dezimal).

BASIC 2.0 Befehl existiert nicht in BASIC 2.0

Monitor S "drive#:name",geräte#,xxxxx,yyyyy

xxxxx bedeutet die Startadresse und yyyyy die Endadresse plus 1 (hexadezimal) des Programms.

Beispiele:

```
BSAVE "ZEICHEN",D0,U8,0N B1,P8192,P16384
BSAVE "PROGRAMM", ON B1,P4096
S "0:TEST",8,12000,14000
```

Will man Speicherbereiche in BASIC 2.0 abspeichern, so ist dazu entweder eine Befehlserweiterung oder ein Maschinensprache-Monitor notwendig.

2.9 Abfragen der Floppy-Fehlermeldungen

Es wurde schon einmal darauf hingewiesen, daß die Commodore-Floppystationen mehr als nur einfache Massenspeicher sind. Das Besondere an ihnen ist zum Beispiel der große Vorrat an Fehlermeldungen. (Eine Liste befindet sich im Anhang dieses Buches.) Diese Fehlermeldungen können vom Benutzer abgefragt werden.

BASIC 7.0 PRINT DS$ für die Fehlermeldung

PRINT DS für die Fehlernummer

BASIC 2.0 10 OPEN 1,geräte#,15

20 GET#l,A$: PRINT A$: IF ST <> 64 THEN 20

30 CLOSE 1

Monitor @geräte#

Wenn die Floppy eine Fehlermeldung für Sie bereitgestellt hat, dann erkennen Sie das am Blinken der LED am Laufwerk. Sobald die Fehlermeldung abgeholt oder ein neuer Befehl zur Floppy geschickt wird, erlischt die LED.

Diese Fehlermeldungen sind für Programme mit intensiver Diskettenbenutzung außerordentlich wichtig, so daß das BASIC 7.0 extra zwei Variablen dafür reserviert hat: DS und DS$.

DS enthält stets die Nummer der letzten aufgetretenen Meldung (das muß keine Fehlermeldung sein). DS$ enthält darüberhinaus den gesamten String der Rückmeldung der Floppystation (inklusive der Rückmeldungsnummer).

Ist die Nummer der Rückmeldung kleiner als 2, so handelt es sich grundsätzlich um keine Fehlermeldung sondern um eine Statusanzeige der Floppystation. Eine Null bedeutet zum Beispiel die Meldung 00, OK,00,00, was soviel wie "Alles in Ordnung" bedeutet. Diese Meldung wird immer dann ausgegeben, wenn ein Befehl ordnungsgemäß, ohne Fehler, ausgeführt wurde.

Die Rückmeldung mit der Nummer 1 hat eine besondere Bedeutung beim SCRATCH-Befehl, der uns später interessieren wird.

In BASIC 2.0 ist die Handhabung dieser Rückmeldung nicht so einfach. Hier muß extra ein Programm geschrieben werden, das die Rückmeldung abholt und auf dem Bildschirm ausgibt. Bemerkenswert ist hier die Verwendung der ST-(Status)-Variablen. Diese Variable ist gewissermaßen eine Überwachungsvariable für den gesamten "Verkehr" zwischen Computer und Peripherie. Sie enthält den Wert eines Bytes, dessen Bits folgende Bedeutung haben, wenn sie gesetzt, das heißt auf "1" sind:

Bit 0	Time-Out beim Schreiben von Daten
Bit 1	Time-Out beim Lesen von Daten
Bit 2	Block beim Laden zu kurz (nur Kassette)
Bit 3	Block beim Laden zu lang (nur Kassette)
Bit 4	Fehler beim Lesen (nur Kassette)
Bit 5	Fehler in Prüfsumme (nur Kassette)
Bit 6	letztes Datenbyte übertragen (EOI)
Bit 7	keine weitere Übertragung mehr (EOT)

Deshalb kann unser Programm auch feststellen, wann kein Byte von der Floppy mehr ankommt. Das ist notwendig, da die Rückmeldung unterschiedlich lang sein kann. Bei der Übertragung des letzten Bytes gibt die Floppy außerdem ein EOI-Signal (end of information) an den Computer aus, und der setzt dann das Bit 6 im Statusbyte (ergibt den Wert 64).

Nun haben Sie also den grundsätzlichen Befehlssatz der 1570/71 kennengelernt. Diese Befehle sind nur ein kleiner Teil dessen, was die Floppy noch alles kann. Sie erlauben Ihnen aber schon die einfache Arbeit mit der Floppystation: Das Laden und Speichern von Programmen. Da die Commodore-Floppies aber noch um einiges mehr beherrschen als nur das Laden und Speichern von Programmen, wollen wir uns als nächstes ein bißchen mehr mit den Interna und wichtigen Begriffen in der Verbindung zwischen Floppy und Computer beschäftigen. Einige dieser Dinge haben Sie bewußt oder unbewußt schon angewendet.

3 Informationen zum Arbeiten mit der 1570/71

Dieses Kapitel bringt nun gewissermaßen die ersten Voraussetzungen für ein effektiveres Arbeiten mit der Floppystation. Sie werden etwas über den grundsätzlichen Aufbau von Disketten, über unterschiedliche Diskettenformate und über die Kommunikation zwischen dem Computer und der Floppy erfahren.

3.1 Der Aufbau einer formatierten Diskette

Im Laufe dieses Buches haben Sie schon einiges über das Formatieren gehört, jedoch noch nichts darüber erfahren, was dieser Vorgang eigentlich mit den Disketten macht.

Wenn Sie sich eine neue, unbespielte Compactkassette auf Ihrer Stereoanlage anhören, dann hören Sie im ersten Moment gar nichts. Wenn Sie lauter aufdrehen, können Sie vielleicht ein Rauschen vernehmen: Das ist gewissermaßen der Leerinhalt der Kassette.

Bei einem Kassettenrecorder kann man auf eine neue Kassette direkt Musik aufzeichnen, ohne daß dafür irgendwelche Vorkehrungen notwendig wären. Bei einer Diskette sieht das anders aus, und zwar aus folgenden Gründen:

Sie können davon ausgehen, daß auch eine Diskette einen ähnlichen "Leerinhalt" aufweist wie eine Compactkassette. Nun arbeitet eine Floppystation jedoch nicht analog, sondern digital. Das bedeutet, daß eine Floppy nur zwischen zwei Signalen auf der Diskette unterscheiden kann: entweder High-Pegel oder Low-Pegel. Auch der Lese-Schreibverstärker in der Floppy ist darauf ausgelegt. Bei einem Kassettenrecorder wird einfach verstärkt und weitergegeben, was vom Band kommt; egal, was das ist. Bei einer Floppy wird nur nach gültigen Pegeln - entweder High-(1)-Pegel oder Low-(0)-Pegel - gesucht, und diese Pegel werden in eine Folge von Bits umgewandelt (entweder kommt ein Bit=l oder ein Bit=0).

Wenn Sie einer Floppy nun eine neue Diskette "servieren", so kann die Elektronik in der Floppy und auch das Betriebssystem im ROM der Floppystation (DOS oder Disk Operating System) mit dem darauf vorhandenen "Rauschen" nicht viel anfangen. Eine Floppystation ist auf das Vorhandensein von gewissem Markierungen auf der Diskette angewiesen. Durch unterschiedliche Art und Ausführung der Markierungen entstehen verschiedene Diskettenformate. Hier ist das Wort Format schon enthalten,

denn das bedeutet letztendlich nichts anderes, als das Aufbringen eines Diskettenformates auf die Diskette.

3.1.1 Spuren oder Tracks

Wie Sie wissen, enthält eine herkömmliche Schallplatte spiralförmig angeordneten Rillen. Diese Rillen (oder Spuren) haben zwei Funktionen:
1. sie enthalten die Musikinformation und
2. sie führen den Tonarm des Spielers immer weiter nach innen, zur jeweils folgenden Information.

Hätte also eine Schallplatte nicht diese Rillen, so würde der Plattenspieler hoffnungslos verwirrt werden, da er keine Führung zur nächsten Musikinformation mehr hätte.

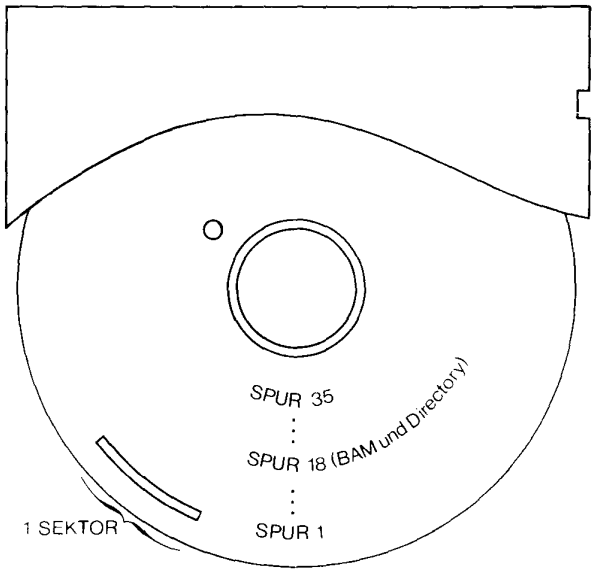


Bild 3.1 *Aufbau einer formatierten Diskette*

Einer Floppystation geht es genauso, wenn man ihr eine unformatierte Diskette einlegt. Auch sie ist auf eine gewisse Art von Rillen angewiesen, die es ihr gestatten, die jeweilige Position des Lese- /Schreibkopfes auf der Diskettenscheibe festzustellen. Der Unterschied zu den Rillen auf der

Schallplatte ist dabei folgender: Erstens sind es keine echten Rillen sondern nur magnetische Signale, die aufgebracht werden, und zweitens sind diese "Rillen" nicht spiralförmig angeordnet, sondern bestehen aus konzentrischen Ringen um den Diskettenmittelpunkt (siehe Bild 3.1). Diese Ringe nennt man bei der Floppystation Spuren oder man bezeichnet sie auch oft mit dem englischen Ausdruck Tracks.

Die Floppystationen 1570 und 1571 arbeiten dabei mit maximal 40 solcher Spuren pro Diskettenseite: Ganz außen liegt dabei Spur 0 und ganz innen liegt Spur 39. Floppystationen mit maximal 40 Spuren auf einer Seite schreiben meistens mit einfacher Dichte (single density); das kann aber auch noch von anderen Faktoren abhängen. Es gibt jedoch Floppystationen, die bis zu 80 Spuren auf einer Diskettenseite verarbeiten. Hier handelt es sich in jedem Fall um Laufwerke mit doppelter Schreibdichte (double density).

Da die Spuren untereinander in keiner Verbindung stehen, muß der Schreib-/Lesekopf der Floppy immer um ein kleines Stück nach innen oder nach außen auf der Diskette bewegt werden, wenn er auf die nächste Spur zugreifen will. Für diesen Zweck hat die Floppy noch einen zweiten Motor, außer dem, der die Diskette dreht. Dieser Motor ist in der Lage, sehr winzige, ruckartige Drehungen zu machen. Er wird daher Schritt- oder Steppermotor genannt. Er ist für die Positionierung des Schreib-/Lesekopfes auf jede Spur der Diskette zuständig und kann diese Aufgabe mit erstaunlicher Präzision lösen.

Wenn Sie eine Diskette formatieren, dann hören Sie in der Regel in schneller Folge ein Klicken aus der Floppystation. Mit jedem Klicken rückt der Schrittmotor den Schreib-/Lesekopf um eine Spur weiter in Richtung Diskettenmitte, um die Formatierung aufzubringen. Wenn Sie ein etwas länger dauerndes Schnarren hören, dann ist das der Steppermotor, wenn er den Kopf über mehrere Spuren hintereinander bewegt.

Die erste Grundvoraussetzung für das Arbeiten mit einer Diskette ist damit geschaffen. Sie verfügen nun über eine Diskette, die so magnetisiert ist. daß die Floppy immer weiß, auf welcher Spur sich der Schreib- /Lesekopf gerade befindet und auch dementsprechend den Kopf auf eine andere gewünschte Spur bewegen kann.

3.1.2 Die Sektoren auf einer Diskette

Jetzt haben wir unsere Diskette zwar in Spuren unterteilt. geholfen ist uns aber damit im Endeffekt wenig. Da eine Spur eine sehr große Speicherkapazität besitzt, ist es sinnvoll, eine Diskette in noch kleinere Einheiten zu zerlegen, auf die noch schneller und genauer zugegriffen werden kann.

Zu diesem Zweck benutzt man Sektoren. Jede Spur einer Diskette besitzt noch eine gewisse Anzahl von Sektoren, die auf der Diskette vorerst die kleinste einzeln ansprechbare Einheit darstellen.

Nehmen wir einmal Spur 1 einer Diskette. Im Commodore-Diskettenformat, das auch als GCR-Format bezeichnet wird, besteht dieser Spur wiederum aus 21 Sektoren. Diese Sektoren sind von 0 bis 20 durchnummeriert. Bei Commodore wurde festgelegt, daß ein Sektor immer genau 256 Bytes an Daten aufnehmen kann. Das entspricht gerade einer Speicherseite im Computer (zum Beispiel dem Bereich $0000 bis $00ff, der Zeropage).

Eine Einheit von 256 Datenbytes wird oft auch als Block bezeichnet. Sie sollten sich daher nicht durcheinanderbringen lassen, wenn im Commodore-Format von Blöcken die Rede ist, obwohl Sektoren gemeint sind.

Wie Sie sehen, kann man nun auf alle 256-Byte-Einheiten der Diskette gezielt zugreifen. Es reicht die Angabe der Spur- und der Sektornummer, und schon weiß die Floppy, wo die entsprechenden Daten stehen. Zum Beispiel ist eine Angabe wie 1,18 üblich. An erster Stelle steht die Spurnummer und hinter dem Komma die entsprechende Sektornummer. In unserem Beispiel soll also ein Zugriff auf Spur 1 Sektor 18 stattfinden.

Wenn im Commodore-Format 256 Bytes als Sektorgröße üblich sind, so gilt das noch lange nicht für die anderen Diskettenformate, die es sonst noch gibt. In der Regel beschränkt man sich bei der Herstellung einer Floppystation auf das eigene, spezifische Format. Bei der 1570/71 ist das jedoch anders. Diese Floppystation kann auch noch andere Formate lesen und sogar schreiben, wodurch dem Benutzer der Zugriff auf fast alle bekannten Diskettenformate gestattet wird. Bei diesen anderen Formaten kann sowohl die Anzahl der Spuren als auch die Sektoranzahl stark voneinander abweichen.

Es gibt zum Beispiel Formate, die besitzen nur noch 5 Sektoren auf einer Spur, dafür enthält aber jeder Sektor 1024 Bytes.

Commodore verwendet in seinem Format nur die Spuren 1 bis 35; andere Hersteller nutzen alle möglichen Spuren von 0 bis 39 für ihr Format aus und beschreiben jede Spur mit Sektoren der Größe von 512 Byte.

Um Ihnen das Verständnis dieser doch ziemlich verworrenen Tatsachen zu erleichtern, wollen wir uns gleich einmal ein bißehen genauer das Commodore GCR-Format betrachten und uns ansehen, wie eine Diskette darin aufgebaut ist.

3.2 Das Commodore GCR-Format

Wie Sie schon im letzten Abschnitt erfahren haben, benutzt Commodore für sein Format nur die Spuren 1 bis 35 auf einer Seite der Diskette. Bei der doppelseitigen 1571 wird auch die zweite Seite verwendet. Die hier vorhanden Spuren (1 bis 35) werden dann einfach mit 36 bis 70 durchnummeriert, so daß auch bei der zweiseitigen Diskette die Angabe einer Spur- und einer Sektornummer genügt, um jede Spur und jeden Sektor einer Diskette zu erreichen.

Commodore verwendet, wie schon erwähnt, immer Sektoren, die 256 Datenbytes enthalten. Diese Sektoren werden von Commodore auch als Blöcke (blocks) bezeichnet.

Da das Commodore-Diskettenformat sehr flexibel ist, sind Disketten auf eine Art und Weise aufgebaut, die man woanders in der Regel nicht findet. So nimmt zum Beispiel die Anzahl der Sektoren von außen nach innen (also von Spur 1 bis Spur 35) ab, da der Durchmesser der Diskette kleiner wird. Die folgende Tabelle zeigt die genauen Zahlen:

Spur 01 bis 17	21 Sektoren
Spur 18 bis 24	19 Sektoren
Spur 25 bis 30	18 Sektoren
Spur 31 bis 35	17 Sektoren
Spur 36 bis 52	21 Sektoren (nur 1571)
Spur 53 bis 59	19 Sektoren (nur 1571)
Spur 60 bis 65	18 Sektoren (nur 1571)
Spur 66 bis 70	17 Sektoren (nur 1571)

Tabelle 3.1 Anzahl der Sektoren pro Spur im GCR-Format

Die Spuren 36 bis 70 entsprechen den Spuren 1 bis 35 auf der ersten Seite (Seite 0) der Diskette und haben dementsprechend die gleichen Sektorzahlen. Die 1571 errechnet also intern die Position für ihren Schreib-/Lesekopf auf der anderen Seite (1) der Diskette dadurch, daß sie einfach von der Spurnummer 35 subtrahiert. Spur 53 auf der Diskette ist also nichts anderes, als Spur 18 auf Seite 1.

An dieser Stelle sei auf die Zählweise eingegangen, wie sie beim Computer üblich ist. Ein Computer zählt nicht von 1 an, sondern er benutzt die 0 als erste Zahl. Wenn wir also 21 Sektoren auf einer Spur haben, so sind das die Sektoren von 0 bis 20. Auch die erste Seite einer Diskette (Vorderseite) ist grundsätzlich Seite 0, die andere Seite demnach Seite 1 (Rückseite).

Bei den Spuren im Commodore-Format verhält es sich genauso, obwohl wir von Spur 1 bis 35 zählen. In Wirklichkeit besteht das Commodore-Format

nämlich aus 36 Spuren (0 bis 35). Die Spur 0 wird lediglich nicht beschrieben, da sie als Anschlagpunkt für den Schreib-jLesekopf definiert ist.

3.2.1 Die Diskettenorganisation im Commodore-Format

Nachdem Sie nun wissen, wie eine Diskette aufgeteilt ist, heißt das noch lange nicht, daß auch deutlich wird, warum eine Floppy so schnell und effektiv arbeitet. Wir wollen uns deshalb in diesem Abschnitt mit der Organisation der Diskette ein wenig genauer beschäftigen.

3.2.1.1 Das Directory der Diskette

Können Sie sich noch an Kapitel 1 und 2 erinnern? Wir haben dort etwas über das Inhaltsverzeichnis, das Directory, einer Diskette erfahren.

Wenn Sie nach Tabelle 3.1 alle Sektoren auf einer Diskette zusammenzählen, dann müßten Sie auf eine Anzahl von 683 beziehungsweise 1366 Sektoren pro Diskette kommen. Das ist die theoretische Gesamtkapazität einer Diskette im Commodore-1570/71-Format. Wollen Sie die Anzahl der möglichen Bytes ermitteln, so brauchen Sie diese Anzahl einfach nur mal 256 (ein Sektor hat 256 Bytes) zu nehmen, und Sie erhalten 163328 beziehungsweise 326656 BYTES FREE für Ihre Disketten.

Das entspricht jedoch nicht ganz der Wahrheit. Formatieren Sie sich einmal eine Diskette, und sehen Sie sich dann das Inhaltsverzeichnis an. Sie lesen die Meldung 664 BLOCKS FREE beziehungsweise 1328 BLOCKS FREE. Wie Sie wissen, entspricht im Commodore-Format ein Block gleich einem Sektorinhalt, so daß Sie sich fragen müssen, wo denn die restlichen 19 (38) Sektoren geblieben sind.

Die Antwort ist einfach. Für die Organisation einer Diskette benötigt das Betriebssystem der Floppy (DOS) einen gewissen Teil der Diskette als eigenen reservierten Arbeitsbereich. In diesem Bereich - es ist jeweils Spur 18 beziehungsweise Spur 18 und 53 einer Diskette - stehen unter anderem das Directory der Diskette und die BAM, die wir im nächsten Abschnitt besprechen.

Wenn Sie also ein Programm auf eine Diskette schreiben, so wird der Programmname in das Directory auf Spur 18 eingetragen (und noch ein paar zusätzliche Informationen), dann wird das Programm auf den Rest der Diskette geschrieben. Wie das genau aussieht, erfahren Sie ab Abschnitt 3.2.2.

Das Directory kann bis zu 144 Dateinamen unterbringen und benötigt deshalb eine Menge Platz. Auf Spur 18 werden deshalb allein die Sektoren I bis 18 für das Directory benötigt.

3.2.1.2 Die BAM der Diskette

Das Kürzel BAM steht für den englischen Ausdruck "block availability map", was übersetzt etwa Blockbelegungsplan heißt. Die BAM, die den Sektor 0 auf Spur 18 (beziehungsweise auf Spur 18 und 53) belegt, gibt Auskunft über das Vorhandensein von freien und belegten Blöcken der Diskette.

Die ganze Angelegenheit funktioniert dabei folgendermaßen: In der BAM ist für jeden Block auf der Diskette ein Bit reserviert. Steht dieses Bit auf logisch 1, so ist der entsprechende Block/Sektor frei. Wird nun ein Programm auf die Diskette geschrieben, so belegt dieses natürlich einen oder mehrere Diskettenblöcke. Damit das DOS nun weiß, welche der Blöcke auf einer Diskette belegt sind, werden diese Blöcke in der BAM entsprechend gekennzeichnet; das entsprechende Bit wird auf 0 gesetzt.

Durch diese Methode weiß das DOS also bei einem nächsten Diskettenzugriff, daß der entsprechende Sektor mit einem Programm oder mit Daten belegt ist und sucht für das neue Programm andere Sektoren, die noch frei sind.

Jetzt werden Sie auch die Angaben im Directory verstehen. BLOCKS FREE gibt an, wie viele Blöcke/Sektoren der Diskette noch nicht als belegt gekennzeichnet wurden, und die Zahl vor jedem Programmnamen gibt die Länge dieses Programms in Blöcken/Sektoren auf der Diskette an. Addieren Sie alle Zahlen vor den Programmnamen mit dem BLOCKS FREE- Wert, so müßten Sie auf die Maximalzahl der freien Blöcke der Diskette (664 oder 1328) kommen.

Diese Bemerkungen zum Commodore-Diskettenformat sollen für den Augenblick genügen. Wir werden eine Diskette natürlich später noch ganz genau unter die Lupe nehmen. Damit das jedoch überhaupt möglich ist, müssen wir uns erst einmal überlegen, wie man mit einer Floppystation kommuniziert.

3.3 Kommunikation mit der 1570/1571

Wir haben im Laufe dieses Buches schon ein paar Mal Zugriffe auf die Diskette vorgenommen. Diese Zugriffe bestanden aber jedesmal aus einem ganzen Komplex von Operationen, die der Computer nach dem Eingeben des Befehls selbsttätig durchgeführt hat (zum Beispiel LOAD).

Uns soll jetzt interessieren, wie die Kommunikation zwischen Computer und Floppy auf niedrigerer Ebene möglich ist. Wie sendet man zum Bei-

spiel nur einzelne Bytes zur Floppy, oder wie legt man Daten auf einer Diskette ab.

3.3.1 Der Kommandokanal

Im Laufe des ersten Kapitels haben Sie schon ein wenig über die Verbindung zwischen Floppy und Computer, den seriellen Bus, erfahren. Der Computer muß eine Geräteadresse über diesen Bus schicken, um das gewünschte Gerät (Floppy, Drucker, usw.) anzusprechen.

Wurde also diese Geräteadresse über den Bus gesendet, dann muß man der Floppy noch mitteilen, was es zu tun gibt. Wir müssen an die Floppy zum Beispiel einen Befehl schicken, der besagt, daß das Inhaltsverzeichnis einer Diskette gelesen oder daß ein Programm geladen werden soll.

Zu diesem Zweck verwenden wir den sogenannten Kommandokanal. Die Floppy verfügt intern über 16 verschiedene Kanäle, die man mit verschiedenen Gängen vergleichen kann, wie in einem großen Gebäude. Jeder dieser Kanäle hat dabei eine bestimmte Funktion. Die einen sind dafür zuständig, daß Daten gelesen werden, die anderen sorgen dafür, daß ein Programm von der Diskette geladen wird und wieder andere besorgen das Schreiben eines Programms auf die Diskette.

Bei der 1570/1571 ist es so organisiert, daß der Kanal Nummer 0 grundsätzlich für das Laden eines Programms verantwortlich ist. Spricht der Computer also Kanal 0 an, so "weiß" die Floppy, daß jetzt ein Programm geladen werden soll. Sie fordert vom Computer dann nur noch den Programmnamen an und schickt ihm daraufhin jedes Byte dieses Programms zu.

Kanal Nummer 1 ist für das Speichern eines Programms zuständig. Eröffnet der Computer eine Leitung für Kanal 1, so erwartet die Floppy zunächst den Programmnamen, der dann ins Directory geschrieben wird. Danach holt sie sich vom Computer Byte für Byte des betreffenden Programms und schreibt diese in die einzelnen Sektoren der Diskette.

Die Kanäle Nummer 2 bis 14 sind für den Benutzer reserviert, das heißt, hier kann der Benutzer bestimmen, was er mit diesen Kanälen anfangen will. Wir werden bald lernen, wie wir auch diese Kanäle zum "Rauchen" bringen.

Der Kanal 15, der uns schon in der Überschrift dieses Abschnittes vorgestellt wurde, ist der sogenannte Kommandokanal. Da die Floppy über eine Vielzahl von Befehlen verfügt, muß ihr irgendwie mitgeteilt werden, welchen Befehl sie auszuführen hat. Bei LOAD und SAVE ist das nicht not-

wendig. Hier reicht es, wenn der Computer einfach einen Dateinamen über Kanal 0 oder 1 schickt, und schon "weiß" die Floppy, daß entweder ein Programm geladen oder gespeichert werden soll.

3.3.2 Die Verwendung des Kommandokanals

Sie sollen jetzt natürlich erfahren, wie man denn nun an die vielen Befehle der Floppy herankommt; wie man sie an die Floppy schickt und vor allem, welcher Art die Befehle überhaupt sind.

Betrachten wir zuerst das Senden von Befehlen an die Floppy. Rufen Sie sich einmal das Formatieren einer Diskette ins Gedächtnis zurück, wie es unter BASIC 2.0 stattfindet:

```
OPEN 1,8,15
PRINT#1,"N0:TESTDISKETTE,TD"
CLOSE 1
```

Wenn man den OPEN-Befehl verwendet, so gibt man zuerst eine Dateinummer an. Diese Nummer hat nur für den Computer eine Bedeutung und dient computerintern zur Organisation von mehreren offenen Dateien. Der Computer muß ja wissen, welche Datei gemeint ist, wenn man mehrere geöffnet hat. In unserem Fall ist das eine Datei mit der Nummer 1. und wie Sie sehen, werden auch über diese Nummer entsprechend (PRINT#1), Daten zur Floppy gesendet.

Die zweite Angabe kennen Sie schon: Es ist die Gerätenummer des Geräts. daß Sie ansprechen wollen. In unserem Fall die Floppy mit der Nummer 8.

Die dritte Angabe, und jetzt wird's interessant, ist die Sekundäradresse oder Kanalnummer. Diese dritte Angabe hat nun mit dem Computer nichts mehr zu tun, sondern sie wird direkt an das Gerät geschickt, das wir angesprochen haben. Es wurde in Kapitel 1 schon erwähnt, daß die Sekundäradresse (oder Kanalnummer) in der Regel Steuerfunktionen hat, um ein bestimmtes Gerät in bestimmte Betriebsarten zu versetzen. So ist es z.B. bei der Floppy.

Die Angabe der Sekundäradresse im OPEN-Befehl ist beim Umgang mit Floppies immer zwingend erforderlich, da hiermit der entsprechende Kanal für die Bearbeitung ausgewählt wird. In unserem Fall - beim Schicken eines Befehls zur Floppy - ist das natürlich der Kommandokanal mit der Nummer 15.

Außer dem Senden von Befehlen zur Floppy können wir über den Kommandokanal aber auch etwas empfangen:

```
10 OPEN 1,8,15
20 GET#1,A$: PRINT A$;: IF ST<>64 THEN 20
30 CLOSE 1
```

Wie Sie sehen, gibt die Floppy über den Kommandokanal auch eine Status- oder Fehlermeldung heraus. Wenn Sie nicht in BASIC 7.0 arbeiten, ist es ein Kinderspiel, einen Befehl zur Floppy zu senden, dessen Ausführung abzuwarten und schließlich nachzusehen, ob alles zur Zufriedenheit erledigt wurde. Zum Beispiel beim Formatieren einer Diskette:

```
10 OPEN 1,8,15  :REM Kommandokanal öffnen
20 PRINT#1, "N0:DISKETTENNAME,DN"     :REM Befehl senden
30 GET#1,A$     :REM Rück_eldung holen
40 PRINT A$;    :REM Zeichen auf Bildschirm
50 IF ST<>64 THEN 30  :REM Ende der Meldung abwarten
60 CLOSE 1 :REM Kanal schließen
```

Wir können also direkt nach dem Senden eines Kommandos den Status der Floppy abfragen. Dauert der Befehl eine Weile, so hält der Computer beim Holen der Rückmeldung solange an, bis die Floppy den Befehl ausgeführt hat und die Meldung herausgibt. Hat bei unserem Programm alles einwandfrei funktioniert, so muß auf Ihrem Bildschirm die Meldung

```
00, OK,00,00
READY
```

erscheinen. Ist ein Fehler aufgetreten, so kann dort zum Beispiel stehen:

```
24, READ ERROR,18,00
READY
```

Diese Meldung ist das typische Zeichen für eine defekte Diskette, wobei hinter dem Text natürlich auch andere Spur- und Sektorangaben stehen können. Im Anhang finden Sie eine Aufstellung aller Fehlermeldungen mit den Bedeutungen der einzelnen Parameter.

Jetzt wissen Sie also, wie man den Kommandokanal bedient. Hierbei muß man leider auf die Ebene des BASIC 2.0 heruntersteigen, da das BASIC 7.0 viele der einzelnen Befehle (zum Beispiel Lesen der Rückmeldung) schon implementiert hat und die Bedienung des Kommandokanals dabei automatisch übernimmt. Die Variable DS$ wird zum Beispiel automatisch aktualisiert, sobald die Floppy über BASIC angesprochen wurde.

4 Die Dateibehandlung auf der Diskette

Nachdem wir nun wissen, wie wir Meldungen und Befehle von der Floppy erhalten, beziehungsweise an sie senden, wollen wir das neue Wissen gleich einmal anwenden, indem wir neue Befehle der Floppystation und mehrere neue Dateitypen (außer den schon bekannten Programmdateien) kennenlernen.

An dieser Stelle ist eine Bemerkung angebracht: Natürlich ist es ein leichtes, dieses Buch mit den Eigenheiten der BASIC-Dialekte von BASIC 2.0 und BASIC 7.0 zu füllen. Das ist aber nicht der Sinn dieses Buches. Wenn also künftig Beispiele angegeben werden, die sich direkt auf den Kommandokanal oder die Dateibearbeitung beziehen, dann werden diese Beispiele in dem auf beiden Computern (C64 und C128) lauffähigen BASIC 2.0 angegeben. Für die Cl28 Besitzer, die auch nur geringe Kenntnisse besitzen (oder zumindest das Handbuch) und bei den kurzen Beispielen Wert darauf legen, dürfte es kein Problem sein, einen OPEN-Befehl in einen DOPEN-Befehl umzuwandeln. Im Prinzip sind im BASIC 7.0 sowieso nur einige Erleichterungen hinzugekommen, die auf weniger Tipparbeit hinauslaufen. Gibt es jedoch spezielle Befehle des C128, die für den Aufruf von floppyinternen Funktionen verantwortlich sind, dann werden diese selbstverständlich dargestellt. Es soll hier nur eine Spitzfindigkeit zwischen OPEN und DOPEN oder CLOSE und DCLOSE vermieden werden, die außer gefülltem Papier nichts bringt.

Jetzt kommen wir aber zu ein paar neuen Befehlen und Tips auf der 1570/71, die bei intensiverer Arbeit mit diesen Geräten sicher unumgänglich sind.

4.1 Die Joker im Diskettenbetrieb

Die Commodore-Floppies haben alle eine Eigenschaft, die sie von vielen anderen Produkten unterscheidet und die das Arbeiten mit einer Floppystation in vielen Fällen stark erleichtert.

Die Rede ist von sogenannten Jokern (im englischen mit "wild cards" bezeichnet). Diese Joker sind bei der Floppy die beiden Zeichen "*", und "?".

4.1.1 Das Sternchen im Dateinamen

Haben Sie schon einmal ein Programm von der Diskette laden wollen, das irgendeinen exotischen Namen besitzt, der noch dazu 16 Zeichen lang ist und mit SPE beginnt? Wenn ja, dann haben Sie sicher in mühevoller Arbeit das Directory geladen, um nachzusehen, wie der Name genau geschrieben wird, um beim Eintippen keinen Fehler zu machen.

Diese Arbeit hätten Sie sich sparen können, wenn Sie nur gewußt hätten, daß Ihr Programm das einzige auf der Diskette ist, dessen Name mit SPE beginnt; oder zumindest das erste der Programme mit SPE. Für solche Fälle hat sich Commodore den Joker einfallen lassen (Achtung: Die folgenden Hinweise gelten nicht für den Betrieb unter CP/M!).

Wenn Sie von einem Programm nur die ersten paar oder nur den ersten Buchstaben kennen, dann können Sie den Programmnamen abkürzen. Die Floppy sucht dann im Directory nach dieser ersten Zeichenfolge und lädt das Programm (oder die Daten).

Sie haben z.B. eine Diskette, auf der sich zwei Programme befinden: 1. TESTPROGRAMM und 2. TESTPROGRAMM. Auf dieser Diskette haben Sie alle Programme nach dem gleichen Muster benannt. Es reicht nun, wenn Sie die Nummer des TESTPROGRAMMs wissen, um das richtige Programm in den Computer zu laden. In unserem Fall vielleicht das zweite: Anstatt

```
LOAD "2. TESTPROGRAMM",8,1
```

tippen Sie einfach

```
LOAD "2*",8,1
```

oder

```
LOAD "2.*",8,1
```

je nachdem, wie Sie gerade wollen. Damit haben wir die Funktion des ersten Jokers, nämlich des "*", kennengelernt. Dieses Zeichen versetzt Sie in die glückliche Lage, Programm- oder Dateinamen abzukürzen, um überflüssige Tipparbeit zu sparen.

Zu beachten ist dabei: Die Floppy holt immer die erste Datei, das der Abkürzung entspricht. Bei einer Diskette mit den Programmen TESTPROGRAMM 1 und TESTPROGRAMM 2 wäre eine Abkürzung sicher nicht sinnvoll.

Wollen Sie das allererste Programm von einer Diskette laden, dann heißt der Befehl dazu

```
LOAD ":*",8
```

Der Doppelpunkt steht dabei für ein Zeichen, daß die Floppy im Dateinamen ignoriert. Es soll also die erste Datei geladen werden, die überhaupt einen Dateinamen besitzt, und das ist immer die allererste Datei einer Diskette (eine Datei ohne Dateiname existiert ohnehin nicht!). Nun werden Sie sich fragen, warum man dazu nicht die Befehlsfolge

```
LOAD "*",8
```

verwendet. Dieser Befehl wird jedoch nicht fehlerfrei vom Betriebssystem interpretiert. Ursprünglich war er dazu gedacht, eine Datei zu laden, die den gleichen Namen der vorher geladenen besitzt, um so immer wieder auf die gleiche Datei zugreifen zu können. Von dieser Idee ist Commodore jedoch dann abgekommen, das heißt, es gibt Floppystationen, die laden mit diesem Befehl die erste Datei einwandfrei; wieder andere bringen bei einem Diskettenwechsel einen FILE NOT FOUND ERROR, da die Datei mit gleichem Namen auf der anderen Diskette nicht existiert. Bei den neuen Floppies 1570/71 kann man allerdings davon ausgehen, daß ein LOAD "*",8 wie das LOAD ":*",8 arbeitet.

4.1.2 Das Fragezeichen im Dateinamen

Neben dem "*" gibt es auch noch einen anderen Joker bei der 1570/71.
Dieser Joker ist das Fragezeichen "?".

Während das "*" einen Dateinamen abkürzt - wobei die Floppy alles, was nach dem "*" als Dateiname folgt, ignoriert -, steht das "?" immer für ein beliebiges Zeichen im Dateinamen.

Nehmen wir an, Sie wollen ein Programm laden, dessen Name aus 4 Zeichen besteht, wobei die letzten beiden Zeichen ".2" lauten. Anstatt nachzusehen, wie der komplette Dateiname lautet, können Sie jetzt die fehlenden ersten Zeichen durch einen Joker ersetzen:

```
LOAD "??.2",8,1
```

Diese Befehlskombination lädt das erste Programm von der Diskette, das im Directory mit einem vier Zeichen langen Namen eingetragen ist und dessen zwei letzte Zeichen" .2" lauten.

Wollen Sie das erste Programm von der Diskette holen, das einen Dateinamen mit 8 Zeichen Länge hat, so tippen Sie:

```
LOAD "????????",8,1
```

Natürlich kann man sämtliche Joker miteinander kombinieren.

4.1.3 Die Kombination von Jokern im Dateinamen

Es steht Ihnen frei, wie Sie das "*" und das "?" miteinander kombinieren; Sie sollten nur beachten, daß für die Floppy ein Dateiname bei einem eventuell vorhandenen "*" beendet ist.

```
LOAD "4??? TEST*",8,1
```

lädt zum Beispiel ein Programm, dessen Dateiname eine 4 am Anfang, danach noch 3 beliebige Zeichen mit der Folge" TEST" und danach ebenfalls wieder beliebige Zeichen besitzt. Eine Angabe, wie

```
LOAD "TEST*.???",8,1
```

lädt das erste Programm, das gefunden wird und dessen Name mit TEST
beginnt. Die Zeichen hinter dem "*" werden ignoriert.

Generell gilt: Die Floppy ignoriert alle Zeichen hinter dem "*", die darauf hindeuten, daß sie zu einem Dateinamen gehören. Dazu zählen nicht die Sonderzeichen, wie "," oder ":" oder "=".

4.2 Das Löschen einer Datei (SCRATCH-Befehl)

Sicherlich kommt es im Lauf der Zeit einmal vor, daß Sie ein abgespeichertes Programm später nicht mehr brauchen, oder Daten liegen auf einer Diskette, die unnütz sind, aber Platz wegnehmen.

Für diesen Fall gibt es den SCRATCH-Befehl. Hiermit ist es möglich, unerwünschte Dateien von der Diskette wieder zu löschen.

BASIC 7.0 SC RA TCH "name",Ddrive#,Ugeräte#

BASIC 2.0 OPEN l,geräte#,15,"Sdrive#:name"

Monitor @geräte#,Sdrive#:name

Beispiele:

```
SCRATCH "WEG DAMIT", D0, U8
SCRATCH "*"
```

```
OPEN 1,8,15, "S0:TEST*"
@8,S:????TEST*
```

Die Bedeutungen der einzelnen Abkürzungen finden Sie in Kapitel 2.1.

Wenden Sie den SCRATCH-Befehl unter BASIC 7.0 an, dann folgt nach Drücken der RETURN-Taste die Frage ARE YOU SURE, die mit "Y" oder YES zu beantworten ist. Ansonsten wird der Befehl abgebrochen. Nachdem der Befehl ausgeführt wurde, wird unter BASIC 7.0 automatisch eine Meldung der Floppy angezeigt:

```
01, FILES SCRATCHED,xx,00
```

Diese Meldung kann auch unter BASIC 2.0 (siehe Kapitel 2.9) abgeholt werden. Sie gibt an, wieviele Dateien mit dem entsprechenden Namen auf der Diskette gelöscht wurden (xx steht dabei für die Anzahl). Normalerweise ist xx immer 1. Es können aber im SCRA TCH-Befehl die schon bekannten Joker verwendet werden, wobei eine Befehlsfolge, wie

```
SCRATCH "TEST*"
```

beziehungsweise

```
OPEN 1,8,15, "S:TEST*"
```

alle Dateien löscht, deren Namen mit TEST beginnen. Hier ist die Rückmeldung über die Anzahl der gelöschten Dateien durchaus sinnvoll. Auf diese Weise ist es natürlich möglich, eine gesamte Diskette zu löschen; hierzu verwendet man jedoch bevorzugt den NEW- oder HEADER-Befehl ohne ID, der in Kapitel 2.2 beschrieben ist.

4.3 Aufräumen auf der Diskette (VALIDATE-Befehl)

Dieser Befehl ist vor allem dann sinnvoll, wenn durch einen Fehler beim Diskettenzugriff die BAM und das Directory in Unordnung geraten sind. Hiermit werden alle Dateien auf der Diskette abgetastet und deren Blöcke in der BAM als belegt gekennzeichnet. Der Rest der Blöcke in der BAM wird freigegeben. Die BAM wird hierbei also neu geschrieben.

BASIC 7.0	COLLECT Ddrive# ON Ugeräte#
BASIC 2.0	OPEN 1 ,geräte#, 15, "V drive#"
Monitor	@geräte#,Vdrive#

Beispiele:

```
COLLECT
COLLECT D0 ON U9

OPEN 1,9,15, "V"
OPEN 1,8,15, "V0"

@8,V
```

Wann die BAM durch einen Fehler zerstört werden kann, haben Sie schon in Kapitel 1 erfahren. Es kann zum Beispiel passieren, daß ein Programm bei SAVE nicht mehr auf die Diskette paßt. In diesem Fall wird der SAVE-Befehl abgebrochen und die Diskette auf 0 BLOCKS FREE gesetzt. Gleichzeitig erscheint vor dem Dateinamen ein "*", der darauf hindeutet, daß mit dem Programm etwas nicht stimmt. Außerdem ist noch die Anzahl der vom Programm belegten Blöcke ebenfalls auf 0 gesetzt worden.

In diesem Fall hilft der COLLECT- oder VALIDATE-Befehl, der die Diskette aufräumt und die verlorengegangenen Blöcke der zu langen Datei wieder freigibt.

Je nach Belegung der Diskette kann der Befehl einige Zeit für die Ausführung brauchen. Es ist aber auf jeden Fall darauf zu achten, daß die Diskette keine Schreibschutz-Plakette trägt, da sonst eine Fehlermeldung die Folge ist.

4.4 Das Umbenennen von Dateien (RENAME-Befehl)

Dieser Befehl ist selten wirklich nötig; oft aber eine Frage des guten Geschmacks. Hiermit kann der Name einer gespeicherten Datei nachträglich im Inhaltsverzeichnis geändert werden.

BASIC 7.0	RENAME "a-name" TO "n-name",Ddrive#,Ugeräte# wobei a-name der bisherige Name der Datei und n-name der neue Name der Datei sein soll.
BASIC 2.0	OPEN 1,geräte#,15,"Rdrive#:n-name=a-name"
Monitor	@geräte#,Rdrive#:n-name=a-name

Beispiele:

```
RENAME "TEST" TO "KEIN TEST",D0,U8
RENAME "ALTER NAME" TO "NEUER NAME"
```

```
OPEN 1,8,15, "R0:NEUER NAME=ALTER NAME"
@9,R:NEU=ALT
```

Zu beachten ist bei dem Wechsel zwischen BASIC 7.0 und BASIC 2.0 die unterschiedliche Reihenfolge der Namen in der Befehlszeile!

4.5 Das Kopieren von Dateien (COPY -Befehl)

Dieser Befehl wurde eigentlich für Doppellaufwerke entwickelt und hat seine Bedeutung bei den Einzellaufwerken eigentlich verloren. Er kann jedoch noch für das Kopieren innerhalb einer Diskette verwendet werden, wenn zum Beispiel mehrere Dateien zu einer einzigen verknüpft werden sollen, oder wenn von einer Datei eine Sicherheitskopie (backup) angefertigt werden soll.

An dieser Stelle ist zu beachten, daß im BASIC 7.0 für das Anhängen einer Datei an eine andere und für das Kopieren von einer Datei in eine andere zwei verschiedene Befehle vorhanden sind, die jedoch prinzipiell das gleiche bewirken: Sie senden den COPY - oder C- Befehl an die Floppy. Der einzige Unterschied zwischen beiden Befehlen besteht darin, daß die Zieldatei bei CONCAT schon auf der Diskette sein muß, während sie bei COPY noch nicht vorhanden sein darf. COPY fertigt also eine Sicherheitskopie an, während CONCAT eine Datei an eine schon bestehende anhängt.

BASIC 7.0 COPY Ddrive#2,"name2" TO Ddrive#l,
"name1" ON Ugeräte#
CONCAT Ddrive#2,"name2" TO Ddrive#l,
"name1" ON Ugeräte#

Die Nummern 2 und 1 hinter den Drivenummern und den Dateinamen stehen für Quelle und Ziel (Laufwerk und Datei).

BASIC 2.0 OPEN 1,geräte#,15,"Cdrive#1:name1=drive#2:name2,
drive#3:name3,..."

Monitor @geräte#,Cdrive#l:name1=drive#2:name2,drive#3:name3,...

Beispiele:

```
COPY "TEST1" TO "TEST2"
CONCAT D0,"NEUES ZUM" TO "ALTEH" OH U8
```

```
OPEN 1,8,15,"C:NEUE8 FILE=DAS,UND DAS,UND DAS AUCH"
OPEN 1,8,15,"C0:BACKUP=0:0RIGINAL"

@8.C0:TEST=0:TEST1,0:TEST2
```

Auch bei diesen Befehlen muß wieder auf die unterschiedliche Reihenfolge der Dateinamen geachtet werden. Verwendet man BASIC 7.0, dann kopiert man die eine Datei zum anderen (Name 2 TO Name 1); bei BASIC 2.0 erzeugt man eine neue Datei aus einem oder mehreren alten Dateien (Name 1=Name 2,Name 3,Name 4).

4.6 Das Kopieren einer Diskette (DUPLICATE-Befehl)

Dieser Befehl ist bei der 1570/71 nicht vorhanden, da er eine gesamte Diskette von einem Laufwerk auf ein anderes kopiert, was bei einem EinzeILaufwerk nicht so ohne weiters möglich ist. Prinzipiell sieht die Syntax des Befehls folgendermaßen aus:

BASIC 7.0 BACKUP Ddrive#2 TO Ddrive#1, Ugeräte#

BASIC 2.0 OPEN 1,geräte#,15,"Ddrive#1=drive#2"

Monitor @geräte#,Ddrive#1=drive#2

Beispiele:

```
BACKUP D0 TO D1
OPEN 1,8,15,"D1=0“
@8,D1=0
```

Die Nummern hinter der Laufwerksangabe (drive#1,drive#2) stehen jeweils
für Ziel- (1) und Quellaufwerk (2).

5 Die verschiedenen Dateitypen der 1570/71

Bisher haben wir uns mit allen möglichen Befehlen der Floppy zur Filebzw. Dateiverwaltung beschäftigt. Sie haben etwas über die BAM und das Directory erfahren und haben Programme geladen sowie ab gespeichert. Nun sollen uns andere Anwendungen der Floppystation interessieren.

5.1 Die sequentielle Datei (SEQ)

In Kapitel 2.3 haben Sie schon erfahren, daß es auch noch andere Dateitypen, als die schon bekannte PRG-(Programm)-Datei gibt, deren Kennung im Directory immer hinter dem Dateinamen steht.

Als erstes soll uns ein Dateityp mit der Kennung SEQ interessieren. Das SEQ steht für sequentielle Datei und bedeutet folgendes:

Wenn Sie ein Buch aufschlagen, dann lesen Sie dieses Buch (zum Beispiel einen Roman) immer der Reihe nach, von vorne bis hinten durch. Dieses "der Reihe nach", wird auch als sequentiell bezeichnet. Wenn Sie jetzt zum Beispiel eine bestimmte Stelle suchen und keinen Anhaltspunkt haben, wo sich diese Stelle (zum Beispiel ein Fremdwort) im Buch befindet, dann bleibt Ihnen nichts anderes übrig, als das Buch noch einmal durchzulesen (der Reihe nach), bis Sie die betreffende Stelle gefunden haben.

Jetzt kann man aber im Zuge der zunehmenden Verbreitung der Computer auch auf die Idee kommen, ein solches Buch auf eine Diskette zu speichern und vielleicht eine Bibliothek mit Bildschirmterminals auszustatten, wo jeder Band nur noch auf Diskette existiert. Für die Speicherung solcher Romane ist die sequentielle Datei wie geschaffen.

5.1.1 Das Eröffnen einer sequentiellen Datei

Eine sequentielle Datei enthält alle Datenbytes hintereinander in einer Datei. Um eine solche Datei anzulegen, benötigt man fast keinen Aufwand, weshalb diese Form der Datenverwaltung sehr häufig anzutreffen ist.

Wir wollen jetzt eine sequentielle Datei mit dem Namen SEQ-TEST anlegen. Dazu rufen Sie sich noch einmal die Kanalstruktur der Floppy ins Gedächtnis zurück (Kapitel 3.3). Wir wissen, daß die Floppy über 16 Kanäle verfügt, wobei die Nummern 2 bis 14 für den Benutzer frei verwendbar

sind. Wählen Sie für die sequentielle Datei beispielsweise den Kanal mit der Nummer 2:

```
OPEN 1,8,2,"SEQ-TEST,S,U"
```

So einfach ist das. Die LED am Laufwerk der 1570/71 geht an und die Datei wird auf der Diskette eröffnet, das heißt im Inhaltsverzeichnis eingetragen.

Beim Eröffnen einer Datei muß man der Floppy immer mitteilen, welchen Typ man wünscht; die Kanäle können ja jeden beliebigen Dateityp verwalten. In unserem Fall sind das zwei Parameter, die an den Dateinamen angehängt werden. Die Bedeutung des ersten Parameters ist folgende:

S SEQ- Datei
P PRG- Datei
U USR-Datei
L REL-Datei

In unserem Falle ist also ein S zu setzen. Wie Sie sehen, gibt es auch ein Kürzel für Programmdateien. Wenn ein Programm abgespeichert werden soll, nimmt man aber im allgemeinen den Kanal 1, da hier die Angabe P nicht mehr notwendig ist - Kanal 1 ist schon von vornherein für Programmdateien ausgelegt. Das SAVE-Kommando ist also letztendlich nichts anderes, als das Anlegen einer Programmdatei, in der dann die Programmbytes untergebracht werden.

Nun aber wieder zur sequentiellen Datei. Haben wir den Dateityp angegeben, so müssen wir der Floppy nur noch mitteilen, für welchen Zweck wir die Datei öffnen: zum Schreiben, Lesen oder zum Anhängen von Daten.

W Schreiben von Daten in eine neue Datei
R Lesen von Daten aus einer bestehenden Datei
A Anhängen (Append) von Daten an eine bestehende Datei

In unserem Beispiel haben wir also eine neue Datei zum Schreiben von Daten eröffnet und der Eintrag in das Inhaltsverzeichnis der Diskette ist nun erstellt.

5.1.2 Das Schreiben von Daten in eine Datei

Dieser Vorgang ist für alle Dateitypen identisch. Hier geht es um das
Übertragen von Datenbytes in die entsprechende Datei.

```
PRINT#1,"DIES IST EIN SEQUENTIELLES FILE"
```

Hier wird gleich eine ganze Anzahl von Bytes (eine Zeichenkette) in unsere Datei geschrieben. Sie können natürlich auch nur ein einzelnes Byte in die Datei eintragen, wenn Sie z.B. irgend welche Parameter von technischen Messungen ablegen wollen. Das Byte wird folgendermaßen in diese Datei geschrieben:

```
PRINT#1,CHR$(byte);
```

An dieser Stelle eine Bemerkung zu dem Semikolon am Ende des Befehls. Steht bei einem PRINT-Befehl auf den Bildschirm des Computers kein Semikolon am Ende der Zeichenkette, so wird automatisch ein Zeilenvorschub ausgeführt, was gleichbedeutend mit der Ausgabe des ASCII-Zeichens 13 ($0d = RETURN) ist. Dieses ASCII-Zeichen sorgt dafür, daß der nächste PRINT-Befehl in die nächste Bildschirmzeile schreibt. Das gleiche gilt für den PRINT#-Befehl in eine Datei. Hier wird das ASCII-Zeichen ebenfalls mit ausgegeben und hinter die Zeichenkette in die Datei geschrieben. Hat man Text, den man auf Diskette ablegen will, so ist dieser Effekt erwünscht. Hier kann man nämlich dann die einzelnen Zeichenketten mit INPUT# wieder in den Computer holen, da der INPUT#-Befehl das Zeichen $0d als Endekennzeichen für eine Zeichenkette nimmt. Wäre es nicht vorhanden, so würde der Computer mit INPUT# immer weiterlesen, bis er mehr als 255 Zeichen bekommt und mit einem STRING TOO LONG ERROR aussteigt.

Will man jedoch keinen Text auf der Diskette ablegen, sondern zum Beispiel Daten oder Programmbytes, so ist das RETURN störend, da es erstens Speicherplatz verbraucht (hinter jedem einzeln abgespeicherten Byte kommt ein RETURN) und zweitens die Struktur des Datenfeldes zerstört.

Aus diesem Grund ist hinter der BASIC-Zeile mit dem CHR$-Befehl ein Semikolon eingegeben. Es verhindert die Ausgabe des RETURNs und sorgt so für die gleiche Struktur der Daten auf der Diskette wie im Computerspeicher. Für den Bildschirm gilt übrigens das gleiche. Auch hier können Sie durch die Angabe eines Semikolons den Zeilenvorschub hinter PRINT verhindern.

In jedem Fall haben Sie jetzt Ihre Zeichenkette oder Ihr einzelnes Byte in die Datei geschrieben. Dabei fällt Ihnen vielleicht auf, daß die Floppy nach diesem Schreibvorgang nicht angelaufen ist, um die Daten auf die Diskette zu schreiben. Der Grund dafür ist folgender:

Wenn die Floppy Daten für eine Datei bekommt, so werden diese Daten zuerst einmal in einem floppyinternen Pufferspeicher zwischengelagert, und zwar solange, bis ein ganzer Block (256 Byte) erhalten wurde, um auf die Diskette geschrieben zu werden. Die 1570/71 erspart sich damit allzu häu

fige Diskettenzugriffe. Würden Sie jetzt immer wieder Daten zur Floppy schicken, so würde der Laufwerkmotor auf einmal anlaufen, um die Daten auf die Diskette zu schreiben. Sie wissen dann, daß Sie mehr als 256 Bytes zur Floppy geschickt haben. Danach beginnt das Füllen des Pufferspeichers wieder von neuem, bis der Dateizugriff beendet wird.

5.1.3 Das Schließen einer Datei

Ein Dateizugriff wird durch das Schließen der betreffenden Datei beendet.

```
CLOSE 1
```

Das ist alles! Jetzt kann es sein, daß der Motor der Floppy erneut anläuft, damit ein eventuell noch vorhandener halbvoller Puffer auf die Diskette geschrieben werden kann. Die Datei wird im Directory abgeschlossen und die LED am Laufwerk erlischt.

Achtung! Das Schließen einer Datei nach der Bearbeitung ist unbedingt notwendig, da sonst diese sonst "offen" stehen bleibt und die restlichen Daten im Pufferspeicher der Floppy verlorengehen. Eine offene Datei können Sie im Inhaltsverzeichnis ganz einfach erkennen: Sie hat ein Sternchen vor der Angabe des Dateityps und die Anzahl der belegten Blöcke der Datei steht auf 0. Wir werden später noch erfahren, wie man auch einen solchen Fehler wieder behebt. Sie sollten es jedoch auf keinen Fall darauf ankommen lassen.

5.1.4 Die Arbeit mit einer sequentiellen Datei

Wie Sie in den vorherigen Abschnitten gesehen haben, ist die Handhabung einer sequentiellen Datei im Prinzip recht einfach: Datei öffnen, Daten hineinschreiben und schließlich Datei wieder schließen. Wie der Name dieses Dateityps schon sagt, werden die Daten genau in der Reihenfolge auf die Diskette geschrieben, in der sie bei der Floppy ankommen: also sequentiell. Zum Lesen dieser Daten wird in der Floppy ein spezieller Zeiger auf den Anfang der Datei gesetzt, der die Position in der Datei angibt.

Stellen Sie eine kleine sequentielle Datei her:

```
OPEN 1,8,2,"SEQ-DATEI,S,W"
PRINT#1,"1. EINTRAG"
PRINT#1,CHR$(1);
PRINT#1,"2. EINTRAG"
PRINT#1,CHR$(2);
PRINT#1,"ENDE DER DATEI"
PRINT#1,CHR$(3)
CLOSE 1
```

Die Datei ist folgendermaßen aufgebaut: Zuerst kommt der Text "1. EINTRAG", der mit dem ASCII-Zeichen für RETURN ($0d) auf die Diskette geschrieben wird (zur Erklärung siehe 5.1.2). Danach folgt ein numerischer Wert in Form eines Bytes, das einzig und allein noch einmal die Nummer des Eintrags enthält. Es wird ohne anschließendes RETURN auf die Diskette geschrieben, da es sich nicht um eine Text-Zeichenkette handelt. Der "2. EINTRAG" ist genauso aufgebaut. Der Schluß der Datei wird durch das Byte 3 gekennzeichnet.

Da in einer sequentiellen Datei die Reihenfolge der aufgezeichneten Daten berücksichtigt werden muß, können die Daten auch nur in der gleichen Reihenfolge gelesen werden wie sie geschrieben wurden. Dieses Problem kann man mit der Anwendung der relativen Datei umgehen, die wir in Abschnitt 5.3 besprechen werden. Deren Handhabung ist jedoch um einiges komplizierter, als die der sequentiellen Datei, weshalb vor allem für den Anfänger die Verwendung der relativen Datei nicht zu empfehlen ist.

Wir können das "sequentielle Problem" auch ein wenig anders lösen. Haben Sie sich vielleicht gefragt, warum im Beispiel immer ein Byte nach einer Zeichenkette kommt, das die Nummer des Eintrags angibt? Nun, stellen Sie sich vor, Sie hätten anstelle des Textes "1. EINTRAG" und "2. EINTRAG" zum Beispiel den Titel eines Kochrezeptes stehen. Sie wären dann nur sehr schwer in der Lage, auf einen bestimmten Eintrag zuzugreifen.

Mit Hilfe der Nummer können Sie jedoch einigermaßen schnell zum Beispiel auf den 2. Eintrag der Datei zugreifen, selbst wenn die Reihenfolge durcheinandergewürfelt ist.

Zum Auslesen der Datei benötigen wir allerdings ein kleines Programm, da INPUT und GET im Direktmodus nicht funktionieren:

```
10 OPEN 1,8,2,"SEQ-DATEI,S,R"
20 INPUT#1.A1$
30 GET#1,N1$
40 INPUT#1,A2$
50 GET#1,N2$
60 INPUT#1,AX$
70 GET#1,NX$
80 CLOSE 1
```

In der ersten Zeile (10) wird Ihre vorhin angelegte Datei als sequentielle Datei zum Lesen geöffnet. Dann können die Daten wieder fast genauso aus der Datei herauslesen werden, wie sie hineingeschrieben wurden: nur mit dem Unterschied, daß die Zeichenketten mit dem RETURN-Zeichen am Ende jeweils mit INPUT# und das einzelne Byte jeweils mit GET# ein

lesen wird. Sie können eine Zeichenkette natürlich auch mit GET# einlesen; das funktioniert folgendermaßen:

```
10 OPEN 1,8,2,"SEQ-DATEI,S,R": A1$=""
20 GET#1,A$
30 IF A$=CHR$(13) THEN 50
40 A1$=A1$+A$: GOTO 20
50 GET#1,N1$
60 ...
.. CLOSE 1
```

Diese Methode ist aber natürlich ziemlich langsam, da sehr viele BASICBefehle für das Lesen einer Zeichenkette ausgeführt werden müssen. Achtung bei Zeichen ketten über 255 Zeichen Länge. In diesem Fall muß eine Spezialvorrichtung eingebaut werden, die solche Zeichenketten auf mehrere Variablen verteilt, da sonst ein STRING TOO LONG ERROR die Folge ist. Solche Zeichenketten können außerdem nur mit PRINT# und anschließendem Semikolon erzeugt werden.

Wenn Sie eine sequentielle Datei also öffnen, dann wird der Zeiger in diese Datei bei jedem Schreib- oder Lesezugriff weitergestellt, so daß pro Zugriff auf eine sequentielle Datei jedes Element nur maximal einmal gelesen oder geschrieben werden kann. Haben Sie jetzt zum Beispiel zehn Elemente überlesen, um vor das elfte zu gelangen und stellen dann fest, daß Sie das neunte Element hätten haben wollen, so müssen Sie die Datei wieder schließen und danach erneut öffnen. Dann müssen Sie wieder die ersten acht Elemente überlesen, um nun das neunte zu erhalten.

Diese Methode ist natürlich ziemlich kompliziert und damit auch sehr zeitraubend, besonders wenn Sie eine sehr große Datei zu verwalten haben, bei der nicht alle Elemente gleichzeitig in den Computer geladen werden können. Die relative Datei bietet dann eine echte Alternative, doch dazu mehr in Abschnitt 5.3.

5.1.5 Das Anhängen von Daten in einer sequentiellen Datei

Natürlich kommt es bei richtiger Datenverarbeitung häufig vor, daß eine Datei zum Beispiel ergänzt werden muß. Auch dafür gibt es bei der 1570/71 einen Befehl. Nehmen wir einmal an, Sie wollen ein Adreßbuch verwalten, bei dem laufend neue Adresse hinzukommen. In diesem Fall wird die Datei beim allerersten Mal angelegt wie in Abschnitt 5.1.1 beschrieben, damit ein Directory-Eintrag vorhanden ist. Nach dem Zugriff wird die Datei natürlich wie immer ordnungsgemäß geschlossen. Jetzt hängen Sie neue Daten an Ihre Datei, die sich ADRESSBUCH nennt. Das geht folgendermaßen:

```
OPEN 1,8,2,"ADRESSBUCH,S,A"
```

Wie Sie sehen, wird die Datei wieder geöffnet - diesmal mit dem A-Kommando (A für Append). Dieses Kommando stellt den Zeiger in die Datei automatisch hinter den letzten bisher vorhandenen Eintrag und macht die Floppy somit bereit für neue Daten, die hinten an die Datei angehängt werden.

Dieser Vorgang kann theoretisch beliebig oft erfolgen; jedoch können damit keine Daten gelöscht werden. Will man Daten löschen, so muß die Datei völlig neu aufgebaut und in eine andere Datei umgeladen werden.

5.2 Die Benutzerdatei (User- Datei. USR)

Dieser Dateityp unterscheidet sich in der Handhabung überhaupt nicht von der sequentiellen Datei und ist auch genauso aufgebaut. Er unterscheidet sich lediglich in der Anwendung.

Während die sequentielle Datei in der Floppy generell für die Datenaufzeichnung verwendet wird, soll die User-Datei für das Aufnehmen von Maschinenprogrammen dienen. Die sequentielle Datei ist also ein Datenfile, während es sich bei der User-Datei (wie auch bei der Programm-Datei) um ein Programmfile handelt.

Diese Vereinbarung wurde von Commodore getroffen, damit sich die Dateien zumindest äußerlich voneinander unterscheiden. Der innere Aufbau ist nämlich vollkommen identisch. Sie können also theoretisch jederzeit ein Kochbuch in einer USR - Datei und ein Maschinenprogramm in einer SEQDatei ablegen.

Wir werden die User-Datei später noch im Zusammenhang mit einem Spezialbefehl der Floppy kennenlernen.

5.3 Die relative Datei (REL)

Dieser Dateityp wurde vorhin schon einmal erwähnt. Er stellt den schnellsten und zugleich auch kompliziertesten Dateityp dar und sollte nur verwendet werden, wenn komplexe Dateien möglichst schnell verwaltet werden sollen und wenn der Programmierer schon Erfahrung im Umgang mit der 1570/71 besitzt.

Die relative Datei wurde ursprünglich für das BASIC 4.0 der Commodore-Serien 4000 und 8000 vorgesehen, da deren BASIC die Handhabung unterstützt und erleichtert. Da dieses BASIC auch im BASIC 7.0 des C 128 enthalten ist, kommen dem Commodore-128-Anwender die Erleichterungen zugute. Aber auch C64-Besitzer sollten sich nicht davor scheuen, diese komfortable und schnelle Datenverarbeitung kennenzulernen; auch wenn das mit ein bißchen mehr Aufwand verbunden ist.

5.3.1 Eröffnen einer relativen Datei

Der Nachteil der sequentiellen Datenspeicherung besteht eigentlich nur darin, daß man nicht auf einen bestimmten Dateieintrag willkürlich zuzugreifen kann. Die ganze Datei besteht aus einem einzigen Datensatz, der solange durchsucht werden muß, bis man seinen entsprechenden Eintrag gefunden hat.

Die relative Datenspeicherung geht einen anderen Weg. Es wird hier davon ausgegangen, daß jede Datei aus vielen Einträgen besteht, die alle eine gewisse Maximalgröße nicht überschreiten. Bei der relativen Datei wird also für jeden dieser Einträge, deren Größe vom Benutzer definiert wird, ein eigener Datensatz (Record) angelegt, auf den bei späterer Bearbeitung direkt zugegriffen werden kann.

Haben Sie beispielsweise eine Adreßkartei, in der Name, Adresse und Telefonnummer jedes Kunden abgelegt werden, so teilen Sie der Floppy beim Eröffnen der Datei mit, daß Sie für jede Eintragung 40 Zeichen Länge gewähren. Die Floppy legt danach eine relative Datei an, die in Datensätze zu je 40 Zeichen aufgeteilt wird. Auf jeden dieser Datensätze kann der Benutzer dann willkürlich zugreifen, ohne sich um irgendeine Reihenfolge kümmern zu müssen. In der Praxis sieht das so aus:

Sie eröffnen eine Datei ADRESSEN, bei der jeder Eintrag bis zu 40 Zeichen lang sein darf:

```
OPEN 1,8,2,"ADRESSEN,L,"+CHR$(41)
```

beziehungsweise

```
DOPEN#1,"ADRESSEN",L41
```

für BASIC 7.0.

Wie Sie sehen geschieht das Öffnen analog zur sequentiellen Datei, nur daß anstelle der Betriebsart der Datei die Länge der Datensätze (Recordlänge) steht. Bei einer Länge der Einträge von 40 Zeichen wurden hier 41 Zeichen angegeben, da wir jetzt einmal davon ausgehen, daß jeder Eintrag mit ei

nem PRINT#-Befehl ohne anschließendes Semikolon gesendet wird. Wie Sie wissen, schickt der Computer dann ein RETURN-Zeichen automatisch hinterher, so daß Sie die Länge um eins erhöhen müssen.

Ist die Datei, auf die Sie zugreifen wollen, schon vorhanden, so kann die Angabe der Recordlänge entfallen. Wird sie dennoch angegeben, so muß die Länge aber in jedem Fall mit der beim allerersten Zugriff übereinstimmen.

Die Datei ist jetzt in jedem Fall im Directory eingetragen, und wir gehen davon aus, daß dies der erste Zugriff auf die relative Datei ist. In diesem Fall sollten Sie sich aus folgendem Grund Gedanken machen, wie groß die Datei später eventuell einmal werden soll: Die Datei ist zwar jetzt im Inhaltsverzeichnis eingetragen; die Datensätze sind aber auf der Diskette noch nicht angelegt beziehungsweise für den Zugriff freigegeben. Wollten Sie jetzt Daten in die Datei schreiben, so würde die Floppy eine Fehlermeldung ausgeben. Aus diesem Grund sollte man als erstes den Datensatz freigeben, dessen Nummer man für die höchste hält, die später in der Datei verwendet werden wird. Es werden dann automatisch auch alle niedrigeren Datensätze freigegeben, so daß anschließend die gesamte relative Datei auf der Diskette organisiert vorliegt. Erweitern kann man die Datei zu gegebener Zeit dann immer noch. Da das Freigeben aber eine ganze Weile dauern kann, sollte man diesen Vorgang lieber möglichst früh hinter sich bringen.

Das Freigeben geschieht durch das Anwählen des entsprechenden Datensatzes und das Schreiben eines Leerbytes in diesen Datensatz. Dieses Leerbyte ist für die 1570/71 das Byte 255 ($ff). Wir wollen eine Datei mit 500 Datensätzen anlegen, wobei das Freigeben folgendermaßen aussieht:

```
OPEN 2,8,15      :REM Kommandokanal für Befehl öffnen
PRINT#2,"P";CHR$(2+96),CHR$(244);CHR$(1);CHR$(1)
PRINT#1,CHR$(255) :REM Leerbyte senden
CLOSE1:CLOSE2
```

beziehungsweise

```
RECORD#1,500,1
PRINT#1,CHR$(255)
DCLOSE#1
```

für BASIC 7.0 (das OPEN 1,8,2,... ist schon erfolgt). Beachten Sie im Basic-2.0-Betrieb, daß beim Positionieren immer die Zahl 96 zur Kanalnummer des Datenkanals addiert werden muß!

Wenn Sie die zweite Zeile eingetippt haben, beginnt die LED an der Floppy zu blinken, da Sie auf einen Datensatz zugreifen wollen. der noch

nicht freigegeben ist. Das soll Sie nicht stören. Sie tippen einfach die dritte Zeile ein und mit dieser wird der entsprechende Datensatz und alle niedrigeren sofort freigegeben, was ein Weile dauern kann. Es soll Sie auch nicht irritieren, daß die LED während des Freigebens manchmal leuchtet und manchmal nicht. Das ist normal und hängt davon ab, ob Sie die Fehlermeldung bei der blinkenden LED ausgelesen haben oder nicht. Haben Sie die Meldung nicht ausgelesen, so blinkt die LED nach dem Freigeben wiederum. Sie bekommen dann aber unter Umständen keinen sinnvollen Fehlertext mehr. Nun aber zur Erklärung der einzelnen Schritte.

5.3.2 Das Positionieren auf einen Datensatz (Record)

In der ersten Zeile der letzten Befehlsfolge haben Sie den Kommandokanal mit der Dateinummer 2 geöffnet. Das ist notwendig, da im folgenden ein Befehl an die Floppy geschickt wird, und wie wir wissen, müssen Befehle über den Kommandokanal geschickt werden. Der folgende Befehl ist der P- oder POSITION-Befehl, der uns in die Lage versetzt, den Zeiger innerhalb der Datei auf einen gewünschten Datensatz zu setzen (zu positionieren). Die Syntax dieses Befehls ist folgende:

BASIC 7.0	RECORD file#,datensatz#,byte# datensatz# bezeichnet dabei die Nummer des gewünschten Datensatzes und byte# die Nummer des Bytes in diesem Datensatz.
BASIC 2.0	OPEN 1,geräte#, 15, "P"+CHR$(sekundär#+96) +CHR$(rec Lo)+CHR$(rec Hi)+CHR$(byte#) rec Lo bezeichnet dabei die Nummer Lo des Records rec Hi die Nummer Hi des Records. Zur Berechnung dieser Nummern siehe folgenden Text.
Monitor	nicht anwendbar, wegen fehlender CHR$-Umrechnung

Wie Sie sehen, ist die Anwendung unter BASIC 7.0 sehr viel kürzer und einfacher; wir werden uns jedoch jetzt mit beiden Methoden beschäftigen.

Unter BASIC 2.0 müssen Sie den Kommandokanal öffnen, da der P-Befehl an die Floppy gesendet werden muß. Dieser Befehl hat folgende Parameter: sekundär# gibt die Kanalnummer oder Sekundäradresse an, wie Sie beim Öffnen der relativen Datei festgelegt wurde (in unserem Fall 2). Zu dieser Kanalnummer wird 96 addiert, um die Kompatibilität der Vorgehensweise mit dem Vorgehen des RECORD-Befehls unter BASIC 7.0 zu erhalten. Dies ist eine Commodore-Empfehlung, der man sich nicht verschließen sollte, auch wenn es ein wenig unbequem erscheint. Danach muß die

Nummer des gewünschten Records übergeben werden. Da ein Byte maximal den Wert 255 aufnehmen kann, wurde diese Angabe in 2 Bytes, das Lo- und das Hi-Byte zerlegt. Die Werte berechnen sich daraus wie folgt:

rec Hi	INT(Nummer des Records/256)
rec Lo	Nummer des Records AND 255

Schließlich muß noch das Byte innerhalb des Datensatzes angegeben werden, auf das positioniert werden soll. Das kann maximal den Wert 1 bis 254 erhalten, woraus Sie schon schließen können, daß die Länge eines Datensatzes einer relativen Datei höchstens 254 Zeichen betragen darf. Die Berechnung der Recordnummer läßt den maximalen Wert von 65535 zu. Eine relative Datei kann also höchstens aus 65535 Datensätzen bestehen. Durch das Betriebssystem der Floppy ist auch die Gesamtlänge einer relativen Datei begrenzt. Sie beträgt 182880 Bytes. Das ist bei der 1570 sicherlich keine Einschränkung, da hier die Diskettenkapazität auf maximal 169329 Bytes beschränkt ist. Bei der 1571 sieht die Sache etwas anders aus, da sie eine Kapazität von 338640 Bytes hat.

Unter BASIC 7.0 funktioniert das Freigeben der Datensätze etwas anders. Hier braucht der Kommandokanal nicht extra geöffnet zu werden, da der Computer die Angelegenheit mit dem RECORD-Befehl intern regelt. Zwei Besonderheiten besitzt der RECORD-Befehl gegenüber BASIC 2.0. Erstens wird nicht, wie beim P-Befehl, die Kanalnummer der relativen Datei, sondern dessen Dateinummer (in unserem Beispiel die 1) angegeben, und zweitens kann man die Recordnummer als einen ganzen Wert zwischen 1 und 65535 angeben, ohne sich über eine Aufteilung Gedanken machen zu müssen.

Das Freigeben der Datensätze durch das Senden des Leerbytes CHR$(255) ist bei beiden BASIC-Versionen gleich. Durch diesen Befehl werden jeweils alle Datensätze bis zum angegebenen vollständig mit dem Wert 255 ($ff) gefüllt. In unserem Fall besteht die Datei also aus 500 Datensätzen, wobei jeder 41 Bytes mit dem Inhalt 255 enthält.

5.3.3 Das Schreiben eines Datensatzes

Wenn Sie bisher alles richtig gemacht haben, so müßte auf Ihrer Diskette jetzt eine Datei ADRESSEN stehen, die die Kennung REL trägt und 82 Diskettenblöcke belegt.

Nun wollen wir einen beliebigen Datensatz dieser Datei beschreiben. Verwenden Sie dazu folgende Daten:

```
NA$="HUBERT MEIER"
WE$="HERZOGWEG 4"
OR$="9999 BEISPIEL"
```

Aus diesen Daten bilden wir unseren Datensatz:

```
RE$=NA$+WE$+OR$
```

Der Datensatz heißt nun

```
"HUBERT MEIERHERZOGWEG 49999 BEISPIEL"
```

und soll auf die Diskette unter der Datensatznummer 10 abgelegt werden. Das geschieht folgendermaßen:

Zuerst müssen Sie die Datei wieder öffnen, wobei Sie bei einer relativen Datei nicht angeben müssen, zu welchem Zweck - Schreiben oder Lesen von Daten - die Datei geöffnet wird:

```
OPEN 1,8,2,"ADRESSEN,L"
```

Beziehungsweise

```
DOPEN#1,"ADRESSEN"
```

für BASIC 7.0.

Danach positionieren Sie auf den zehnten Datensatz und dort auf das erste Byte

```
OPEN 2,8,15
PRINT#2,"P"CHR$(2+96)CHR$(10)CHR$(0)CHR$(1)
CLOSE 2
```

beziehungsweise

```
RECORD#1,10,1
```

für BASIC 7.0.

Jetzt können Sie den Datensatz in die Datei eintragen:

```
PRINT#1,RE$
```

wobei das RETURN-Zeichen automatisch hinter die Zeichenkette in die
Datei geschrieben wird. Durch

```
CLOSE 1
```

Beziehungsweise

```
DCLOSE#1
```

für BASIC 7.0 sichern Sie Ihre Eingabe und schließen den Zugriff ab. Das Zugreifen auf einen Datensatz und das Schreiben des Inhaltes kann man vor dem CLOSE-Kommando natürlich beliebig wiederholen und dabei kreuz und quer durch die Datei springen.

Der Vorteil der relativen Datei wird sehr schnell deutlich, wenn man viele Dateizugriffe machen muß, bei denen sich auch noch das Schreiben und Lesen abwechseln.

5.3.4 Das Lesen eines Datensatzes

Analog zum Schreiben von Daten erfolgt deren Lesen. Wenn wir davon ausgehen, daß die Daten als Zeichenkette mit dem RETURN-Zeichen am Ende abgespeichert wurden, dann kann ein Datensatz ganz einfach mit INPUT# wieder eingelesen werden. Ansonsten verwendet man den GET#-Befehl, wobei eine Abfrage auf das Leerbyte 255 in jedem Datensatz dafür sorgt, daß das Lesen bei kürzeren Zeichenketten rechtzeitig abgebrochen wird. Bei kürzeren Zeichenketten ist nämlich der Rest des Datensatzes immer noch mit 255 ($ff) Bytes gefüllt. Zuerst das Beispiel mit INPUT#:

```
10 OPEN 1,8,2,"ADRESSEN,L"
20 OPEN 2,8,15
30 PRINT#2,"P"CHR$(2+96)CHR$(10)CHR$(0)CHR$(1)
40 INPUT#1,A$
50 PRINT A$
60 CLOSE1: CLOSE2
```

bezieh ungsweise

```
10 DOPEN#1,"ADRESSEH"
20 RECORD#1,10,1
30 IHPUT#1,A$
40 PRINT A$
50 DCLOSE#1
```

für BASIC 7.0. Der Datenzeiger wurde so positioniert, daß der gleiche Datensatz ausgelesen wird, der zuvor geschrieben wurde. Dieses Auslesen muß natürlich in einem Programm erfolgen, da ein INPUT im Direktmodus nicht erlaubt ist. Bei beiden Programmen muß der folgende Bildschirmausdruck erscheinen:

```
HUBERT MEIERHERZOGWEG 49999 BEISPIEL
READY.
```

Mit einem entsprechenden Rahmenprogramm kann man jetzt die Einzelheiten des Datensatzes wieder in Namen und Adresse zerlegen und mit dem Beispiel aus Abschnitt 5.3.3 eine komplette Adreßverwaltung schreiben.

Nun noch das Beispiel mit GET#. Wir haben die vorherige Zeichenkette mit dem RETURN-Zeichen 13 ($0d) auf die Diskette geschrieben, deshalb fragt das Programm nach 13 als Endekennzeichen. Im anderen Fall, wenn wir die Zeichenkette mit einem PRINT# und abschließendem Semikolon geschrieben hätten, müßte der Computer das Leerkennzeichen als Endekennzeichen abfragen. Eines muß ja auf jeden Fall vorhanden sein, da wir eine Datensatzlänge von 41 angegeben haben und mit 40 Zeichen pro Datensatz arbeiten:

```
10 OPEN 1,8,2,"ADRESSEN,L": RE$=""
20 OPEN 2,8,15
30 PRINT#2,"P"CHR$(2+96)CHR$(10)CHR$(0)CHR$(1)
40 GET#1,A$
50 IF A$=CHR$(13) THEN 60: REM IF A$=CHR$(255) THEN 60
60 RE$=RE$+A$: GOTO 40
70 PRINT RE$
80 CLOSE 1: CLOSE 2
```

beziehungsweise

```
10 DOPEN#1,"ADRESSEN"
20 RECORD#1,10,1
30 RE$=""
40 GET#1,A$
50 IF A$=CHR$(13) THEN 60: REM IF A$=CHR$(255) THEN 60
60 RE$=RE$+A$: GOTO 40
70 PRINT RE$
80 DCLOSE#1
```

für BASIC 7.0. In Zeile 50 müssen Sie sich also jeweils entscheiden, ob Sie auf das RETURN-Zeichen oder auf das Leerzeichen achten.

Wie Sie sehen, läßt sich mit einer relativen Datei recht gut leben; vor allem unter BASIC 7.0. Haben Sie sich einmal eine riesige Datei angelegt, dann werden Sie vor allem in den Genuß der Geschwindigkeit dieser Datenverarbeitungsmethode kommen. Während Sie bei der sequentiellen Datei jeden Buchstaben erst einmal überspringen müssen, können Sie bei der relativen Datei direkt auf jeden Eintrag zugreifen. Eine Voraussetzung dabei ist allerdings zu beachten. Die relative Datei ist nur dann effektiv, wenn die Datensätze sortiert vorliegen, das heißt, daß Sie sich die Datensatznummer eines Eintrags zum Beispiel aus der Nummer eines anderen Eintrags errechnen können und dann direkt zugreifen. Müssen Sie einen Datensatz erst

suchen, weil ein wirres Durcheinander vorherrscht, so kann auch die relative Datei nicht zaubern; dann braucht eben alles seine Zeit.

5.4 Die Programmdatei (PRG)

Diese Datei zu erwähnen ist fast überflüssig, da wir wohl meistens damit arbeiten. Man kann jedoch nicht nur LOAD und SAVE darauf anwenden.

5.4.1 Das Inhaltsverzeichnis als Programmdatei

Wie Sie das Inhaltsverzeichnis bzw. Directory unter BASIC 2.0 laden können, das wissen Sie ja:

```
LOAD"$",8…
```

Das heißt aber, daß das Inhaltsverzeichnis einem Programm entspricht. Die Floppy verwaltet das Directory intern auf eine andere Weise als die übrigen Programmdateien. Sie richtet es in einer Art her, daß der Computer "meint", er habe es mit einem BASIC-Programm zu tun.

Diese Eigenschaft des Directory kann mit dem Wissen über die Dateitypen der 1570/71 zunutze machen. Es wurde schon einmal angesprochen, daß beim C64 das Laden des Directory jedesmal mit einem Programmverlust verbunden ist. Wir wollen nun einmal sehen, wie wir das Directory auf einem C64 anzeigen können, ohne daß ein Programm dadurch verloren geht. Diese Möglichkeit kann zum Beispiel in einem BASIC-Programm mit Diskettenbetrieb verwendet werden, damit der Benutzer jedesmal nachsehen kann, was für eine Diskette er eingelegt hat.

Das einzige, was Sie zu diesem Thema noch wissen müssen, ist, wie ein BASIC- Programm im Speicher des Computers aussieht. Das ist recht schnell erklärt.

Der BASIC-Speicher beginnt beim C64 bei der Adresse 2049, was hexadezimal $0801 entspricht. An dieser Adresse beginnt die erste BASIC-Zeile. Jede Zeile besteht zuerst aus 2 Bytes, die die Adresse der nächsten BASICZeile angeben. Existiert keine weitere BASIC-Zeile, stehen diese zwei Bytes auf $00. In der ersten Zeile steht also die Adresse der nächsten BASIC-Zeile bei $0801 und $0802; und zwar zuerst das niederwertige (Lo-) Byte und dann das höherwertige (Hi-) Byte. Anschließend folgen wieder zwei Bytes, die die Zeilennummer bilden. In diesem Fall zuerst das Hiund dann das Lo-Byte. Die Adressen und Zeilennummern errechnen sich folgendermaßen (die Bytes in dezimaler Angabe):

Adresse = AdresseHi * 256 + AdresseLo

Nummer = NummerHi * 256 + NummerLo

Nach diesen 4 Bytes der BASIC-Zeile folgt der eigentliche BASIC-Text, der durch ein Nullbyte ($00) abgeschlossen wird. Danach folgt die zweite Zeile und so weiter...

Das Ende eines BASIC-Programms besteht also aus drei hintereinander stehenden Nullbytes: das Ende der letzten BASIC-Zeile und die Adresse der nächsten Zeile, die ja aus zwei Nullbytes besteht, wenn keine Zeile folgt.

5.4.2 Anzeige des Inhaltsverzeichnis unter BASIC 2.0

Die Kenntnis über den Aufbau eines BASIC-Programms ist notwendig, wenn wir uns das Directory mit einem Programm anzeigen lassen wollen.

Zu diesem Zweck eröffnen wir das Inhaltsverzeichnis als Programmdatei und lesen dessen Daten mit GET# aus (wie aus einer sequentiellen Datei, da der Aufbau fast identisch ist). Wir müssen dabei nur den Aufbau eines BASIC-Programms berücksichtigen, da wir erstens wissen müssen, wann das Directory zu Ende ist und da wir zweitens die Adressen, die zwischen zwei BASIC-Programmzeilen stehen, ausfiltern müssen - sie gehören ja nicht zur eigentlichen Information des Directory. Schließlich müssen wir noch die Zeilennummern umrechnen und als Blockanzahl der einzelnen Dateien ausgeben.

Das folgende BASIC-Programm sollten Sie sich einmal ansehen. Es holt die Daten aus dem Directory und zeigt sie auf dem Bildschirm an. Durch seine Kompaktheit ist es für einen Einbau in ein größeres Programm wie geschaffen.

Wie Sie sehen, wird die Directorydatei im Programm nicht mit

```
OPEN 1,8,2,"$,P,R"
```

geöffnet, sondern mit

```
OPEN 1,8,0,"$"
```

In Kapitel 3 haben wir schon gelernt, daß so etwas bei PRG-Dateien möglich ist, da die Kanalnummern 0 und 1 für PRG-Dateien reserviert sind und demzufolge immer eine PRG-Datei angenommen wird, wenn keine andere Angabe gemacht wird. Die Kanalnummer 0 entspricht dabei dem ",P,R" und die Kanalnummer 1 dem ",P,W". Nun aber zum Programm:

```
1000 OPEN 1,8,0,"$": GET#1,A$: GET#1,A$
1010 GET#1,D$: GET#1,A$: PRINT VAL(D$)
1020 FOR X=0 TO 27: GET#1,A$: PRINT A$;: NEXT: PRINT
1030 IF ST=64 THEM 1080
1040 GET#1,A$: GET#1,A$
1050 GET#1,ZL$: GET#1,ZH$
1060 PRIMT ASC(ZL$+CHR$(0))+256*ASC(ZH$+CHR$(0));
1070 FORX=0 TO 27: GET#1,A$: PRINT A$;: NEXT: PRINT
1080 GOTO 1030
1090 CLOSE 1
```

Es kann durchaus sein, daß Sie in diesem Programm noch nicht alle Einzelheiten verstehen. Wir werden jedoch im nächsten Kapitel auf den Aufbau von Dateien und auch vom Directory und der BAM eingehen; danach müßte sich der Schleier eigentlich heben.

5.5 Einige Zusatzbemerkungen zur Dateibehandlung

Wie Sie sehen, kann man im Prinzip alle Dateitypen der 1570/71, außer den relativen Dateien, gleich behandeln. Das liegt an der internen Bearbeitung der Dateien durch die Floppy und an dem Aufbau der einzelnen Dateitypen. Wie wir im nächsten Kapitel sehen werden, sind die Dateien im Aufbau nämlich relativ ähnlich.

Nun noch etwas zum Öffnen von Dateien und/oder Kanälen bei der 1570/71. Bei relativen Dateien haben wir schon einmal 2 Kanäle gleichzeitig geöffnet, nämlich den Kommandokanal und den Kanal für die relative Datei. Im Prinzip kann man auch mehrere Dateien gleichzeitig öffnen. Das sind

3 Dateien bei PRG, SEQ oder USR - Dateien
1 Datei bei REL-Dateien
1 REL-Datei und eine Datei anderen Typs

Sie können also drei sequentielle Dateien gleichzeitig offenhalten und bearbeiten, während nur eine einzige relative Datei offen sein darf.

Diese Zahlen hängen mit den schon erwähnten Pufferspeichern der Floppy zusammen. Die Floppy hat insgesamt 2 KByte RAM. Dieser ist unter anderem in fünf Pufferspeicher zu je 256 Byte Größe aufgeteilt.

Wenn eine sequentielle oder eine Programmdatei eröffnet wird, "verschlingt" das jeweils einen Pufferspeicher, wobei noch zwei andere für die übrige Organisation benötigt werden. Es können also drei Dateien geöffnet werden.

Eine relative Datei benötigt indessen drei Pufferspeicher und einen weiteren für die übrige Organisation. Deshalb kann zu einer relativen Datei nur maximal ein anderer Dateityp geöffnet werden; dann ist der gesamte Pufferspeicher belegt.

Öffnen Sie einmal aus Versehen zuviele Dateien, dann meldet sich die Floppy mit der Fehlermeldung 70, NO CHANNEL,00,00, die besagt, daß kein weiterer Kanal mehr reserviert werden kann.

6 Die Organisation einer Diskette

Die folgenden Abschnitte dieses Buches sollen Ihnen die Arbeitsweise des Betriebssystems der 1570/71 (DOS 3.0) ein wenig deutlicher machen. Dazu nehmen wir die Organisation einer Diskette, das heißt den Aufbau der BAM, des Directory und der einzelnen Dateien einmal genauer unter die Lupe, wobei wir mit der BAM anfangen wollen. Den Sinn und Zweck dieser Einrichtung haben Sie ja schon kennengelernt. Jetzt wollen wir die Organisation in Block 18,0 genauer betrachten.

6.1 Der Aufbau von Block 18,0 (BAM)

Wie Sie sicherlich noch wissen, ist in der BAM das Verzeichnis über belegte und freie Blöcke/Sektoren der entsprechenden Diskette abgelegt. Tabelle 6.1 zeigt die Bedeutung der einzelnen Bytes dieses Sektors für den Betrieb der 1570/71:

Byte	Bedeutung
000	enthält 18 ($12); Spurnummer für Diredory
001	enthält 1 ($01); Startsektor für Diredory
002	enthält 65 ($41); Formatkennzeichen "A"
003	Flag für doppelseitige Disketten (1 = doppelseitige Disk, keine Bedeutung im 1541-Modus)
004	Anzahl der freien Blöcke/Sektoren für Spur 1
005-007	Bitmuster der Blockbelegung für Spur 1: Bit=l bedeutet "Sektor/Block frei" Bit=0 bedeutet "Sektor/Block belegt" Byte 005 enthält die Belegung für Sektor 0 - 7 Byte 006 enthält die Belegung für Sektor 8 - 16 Byte 007 enthält die Belegung für Sektor 17 - 23 (Sektor 21-23 sind natürlich nie vorhanden)
008-011	s.o. 004-007 für Spur 2
...	
140-143	s.o. 004-007 für Spur 35
144-159	Diskettenname, der bei der Formatierung angegeben wird; aufgefüllt mit 160 ($a0)
160-161	zweimal 160 ($a0) SHIFT SPACE
162-163	ID der Diskette
164	160 ($a0) SHIFT SPACE
165-166	$32 und $41 "2A"; Formatangabe der Diskette, wobei es eigentlich "3A" heißen müßte, da die 1570/1571 mit DOS 3.0 arbeitet
167-170	160 ($aO) SHIFT SPACE
171-179	$00 bei 1541-Modus; $ao bei 1570/71-Modus
180-220	0 ($00); nicht benutzter Bereich
221-255	1541/1570: restlicher Bereich nicht verwendet ...

bei 1571:

221-237	Anzahl der freien Blöcke für Spur 36-52
238	Anzahl der freien Blöcke für Spur 53 (immer 0)
239-244	Anzahl der freien Blöcke für Spur 54-59
245-250	Anzahl der freien Blöcke für Spur 60-65
251-255	Anzahl der freien Blöcke für Spur 66-70

Die 1571 enthält zusätzlich noch ein Verzeichnis in Block 53,0:

Byte	Bedeutung
000	enthält 0 ($00)
001-003	s.o. 005-007 für Spur 36
...	
102-104	s.o. 005-007 für Spur 70
105-255	restlicher Bereich nicht verwendet

Tabelle 6.1 Aufbau der BAM bei der 1541/1570/1571

Die Belegungstabelle spricht eigentlich für sich. Hier sind jedoch noch ein paar kleine Anmerkungen:

Die Bytes 0 und 1 eines jeden Sektors haben in der Regel eine Bedeutung, auf die wir nachher - beim Aufbau der Dateien - noch intensiver eingehen werden. Sie zeigen normalerweise immer auf den nächsten Sektor einer Datei, in diesem Fall auf den ersten Sektor des Directory, da BAM und Inhaltsverzeichnis eine zusammengehörige Einheit darstellen.

Die Bytes 004 bis 007 stellen eine der Vier-Byte-Einheiten dar, die die Blockbelegung für jeden Track bzw. jede Spur anzeigen. Ein gesetztes Bit heißt dabei "Block frei", während ein gelöschtes Bit auf einen belegten Sektor hindeutet. Da in diesem Bitmuster 24 Sektoren Platz haben (Sektoren 0 bis 23), wir jedoch über maximal 21 Sektoren pro Spur verfügen, sind die Bits für die Sektoren 21 bis 23 generell gelöscht (Block belegt). Das erste Byte der Vier-Byte-Einheit stellt jeweils die Summe der gesetzten Bits, also die Anzahl der freien Blöcke für eine Spur dar. Dieser Belegungsplan gilt für sämtliche Spuren einer Diskette auf der 1570/71. Spur 18 hat dabei stets die Kapazität 0, das heißt keinen Block frei, da hier die BAM und das Inhaltsverzeichnis stehen. Bei der 1571 gilt das gleiche für Spur 53.

6.2 Der Aufbau des Directory (Block 18,1 ...)

Nun soll uns der Aufbau des Inhaltsverzeichnisses der Diskette interessieren. In jedem Sektor auf Spur 18 (außer 18,0) können acht Dateieinträge, das heißt acht Dateinamen mit Parametern, abgespeichert werden. Da auf Spur 18 noch 18 Sektoren (18,1 bis 18,18) frei sind, entspricht das einer Kapazität von 18*8=144 Einträgen. Tabelle 6.2 zeigt den Aufbau eines Directoryblocks.

Byte	Bedeutung
000	Spurnummer des nächsten Directoryblocks; ggf. 0
001	Sektornummer des nächsten Directoryblocks
002-031	Eintrag der 1. Datei
034-063	Eintrag der 2. Datei
066-095	Eintrag der 3. Datei
098-121	Eintrag der 4. Datei
130-159	Eintrag der 5. Datei
162-191	Eintrag der 6. Datei
194-223	Eintrag der 1. Datei
226-255	Eintrag der 8. Datei

Tabelle 6.2 Aufbau eines Blocks des Directory

Ein Directory-Eintrag ist folgendermaßen aufgebaut:

Byte	Bedeutung
000	Kennzeichen des Dateityps; die Bits haben dabei folgende Bedeutung: Bit 0: die Bits 0-3 kennzeichnen den Dateityp: Bit 1: 0000 – DEL 0011 – USR Bit 2: 0001 – SEQ 0100 – REL Bit 3: 0010 – PRG Bit 4: keine Bedeutung Bit 5: keine Bedeutung Bit 6: Bit=l zeigt SCRATCH-Schutz an Bit 1: Bit=0 zigt offene (ungültige) Datei an; dieses Bit muß also normalerweise immer auf 1 stehen.
001-002	Spur und Sektor des ersten Blocks der Datei
003-018	Dateiname; mit 160 ($a0) aufgefüllt
019-020	Spur und Sektor des ersten Side-Sektor-Blocks (nur bei relativen Dateien)
021	Recordlänge (nur bei relativen Dateien)
022-025	Zwischenspeicher bei DOS-Operationen
026-027	Spur und Sektor der neuen Datei beim Überschreiben mit REPLACE (@)
028-029	Anzahl der Blöcke der Datei (L/H)

Tabelle 6.3 Struktur eines Dateieintrags im Directory

Zur Struktur der Dateieinträge einige Erklärungen:

Wie Sie sehen, besteht der Dateityp aus einzelnen Bits, die alle eine bestimmte Bedeutung haben. Die Bits 0 bis 3 bestimmen den Dateityp. Inter

essant sind für uns auch die Bits 6 und 7. Bit 6 enthält eine Funktion, die das Schützen einer Datei vor dem SCRATCH-Befehl erlaubt. Ist dieses Bit gesetzt, so erscheint im Directory ein "<" hinter der Angabe des Dateityps und der SCRATCH-Befehl wird unwirksam.

Bit 7 kennzeichnet eine nicht ordnungsgemäß geschlossene Datei. An dieser Stelle müßte Ihnen eigentlich sofort Kapitel 5.1.3 ins Gedächtnis kommen. Dort haben wir die Wichtigkeit des CLOSE-Befehls nach dem Bearbeiten einer Datei kennengelernt. Wird dieser CLOSE-Befehl nicht eingesetzt, so bleibt Bit 7 im Dateityp gelöscht, was sich durch ein Sternchen vor dem Typ im Directory äußert. Wir werden später eine Möglichkeit kennenlernen, die einen solchen "Defekt" nachträglich beheben kann (indem das betreffende Bit gesetzt wird). Voraussetzung ist allerdings, daß bis dahin kein weiterer Schreibzugriff auf die Diskette erfolgt. Wollen Sie eine solche offengebliebene Datei löschen, so müssen Sie dafür den VALIDA TE-Befehl (siehe 4.3) verwenden.

Wie Sie sehen, bedeutet auch eine DEL-Datei (also eine gelöschte Datei) nichts weiter, als einen bestimmten Dateityp. Genauso wie der NEW-Befehl im Computer funktioniert auch der SCRATCH-Befehl in der Floppy. Es wird lediglich der Dateityp auf $00 gesetzt und die Blöcke dieser Datei in der BAM wieder freigegeben. Findet nun kein weiterer Schreibzugriff auf die Diskette statt, so kann man den SCRATCH-Befehl jederzeit durch Setzen des ursprünglichen Dateityps wieder rückgängig machen (genauso wie ein OLD beim Computer ein BASIC-Programm wieder restaurieren kann). Wir werden auch das später noch genauer erläutern.

6.2.1 Der REPLACE-Befehl (@:)

Wie Sie aus der Bedeutung der Bytes 26 und 27 ersehen können, existiert bei der 1570/71 ein Befehl, der das Überschreiben einer "alten" Datei durch eine neue gleichen Namens erlaubt. Dieser Befehl nennt sich REPLACE und wurde in diesem Buch bisher deshalb nicht erläutert, da er einen ganz fatalen Betriebssystemfehler enthält, der unter Umständen mehrere Dateien auf einer Diskette zerstören kann. Es wird daher an dieser Stelle ausdrücklich empfohlen, den Befehl nicht anzuwenden, sondern ihn durch die Kombination SCRATCH und anschließendes Neuschreiben der Datei zu ersetzen. Anderslautende Meldungen, die besagen, daß der Fehler inzwischen von Commodore behoben wurde, entsprechen nicht den Tatsachen; sie übersehen nur den Effekt, daß dieser Fehler ganz sporadisch auftreten kann und nur selten (dann aber gründlich) Disketten zerstört. Die Kombination SCRA TCH und normales neues Abspeichern ist in jedem Fall sicherer. Wer den Befehl dennoch ausprobieren möchte, für den ist hier kurz

die Syntax dargestellt: Der Unterschied zum normalen Abspeichern besteht im Setzen eines @drive#: vor den Dateinamen, zum Beispiel

```
DSAVE "@:UEBERSCHREIBEN"
SAVE "@0:TEST",8
OPEN "@:SEQ-FILE,SW"
```

Es wird dann automatisch die bisher vorhandene Datei gleichen Namens mit der neuen Datei überschrieben, wobei zuerst die neue Datei vollständig auf Diskette geschrieben und dann die "alte" gelöscht wird.

6.3 Der Aufbau von Dateien auf der Diskette

Nachdem wir nun über den Aufbau der BAM und des Directory informiert sind, interessiert uns natürlich auch der Aufbau der einzelnen Dateitypen. Dieser ist bei allen Typen ähnlich (ausgenommen relative Dateien) und sehr einfach. Die einzige Angabe, die dabei im Inhaltsverzeichnis enthalten ist, ist die Spur- und Sektornummer des jeweils ersten Blocks in der Datei.

6.3.1 Der Aufbau von sequentiellen Dateien (SEQ)

Die sequentiellen Dateien haben den einfachsten Aufbau aller Dateitypen, der jedoch einem ganz bestimmten Schema der Blockverkettung entspricht. Diese Blockverkettung ist grundlegend für das Commodore-Diskettenformat.

6.3.1.1 Das Prinzip der Blockverkettung

In einem Fall, nämlich beim Aufbau von Directory und BAM, haben wir uns schon die einzelnen Bytes von Sektoren angesehen. Dabei sind Ihnen die beiden ersten Bytes (0 und I) sicher ganz besonders aufgefallen. Sie enthielten jeweils eine Spur- und eine Sektornummer.

Dieser Aufbau eines Blocks/Sektors ist für die gesamte Diskette gleich. Wie Sie wissen, gibt es im Directory in jedem Dateieintrag zwei Bytes, die die Spur- und Sektornummer des jeweils ersten Blocks einer Datei angeben. Nun haben die meisten Dateien aber mehr als einen Block. Um den jeweils folgenden Block einer Datei zu finden, hat man bei Commodore vereinbart. die beiden ersten Bytes eines Blocks generell für die Spur- und Sektornummer des Folgeblocks zu reservieren. Dieses Prinzip nennt man "Linken" oder Blockverkettung. Die Spur- und Sektorangaben nennt man demzufolge Linkbytes oder Linker.

Wie unschwer zu erkennen ist, verringert sich dabei die Kapazität eines einzelnen Blocks auf nur mehr 254 Datenbytes; dafür ist jedoch ein sehr schnelles sequentielles Abarbeiten einer Datei möglich.

Da aber jede Datei ein Ende hat, muß das DOS wissen, wann der letzte Block einer Datei gelesen wurde. Das Kennzeichen hierfür ist die Spurnummer des Folgeblocks, die jetzt auf Null ($00) steht (es gibt ja im Commodore-Format keine Spur 0).

Nun muß aber dieser letzte Sektor nicht mehr voll belegt sein; er kann zum Beispiel nur noch 5 Datenbytes enthalten. Um das zu erkennen, bekommt auch die Sektornummer des Folgeblocks im letzten Block eine andere Bedeutung; sie gibt jetzt an, wieviele Bytes dieses Blocks noch zur Datei gehören ($01 bis $fd). Die restlichen Bytes dieses Sektors sind wohl oder übel verschwendet.

Das Laden der Daten einer Datei ist also recht einfach aber raffiniert gelöst. Es wurde schon erwähnt, daß die sequentielle Datei den einfachsten Aufbau aller Dateitypen hat. Sie besteht lediglich aus Blöcken, die durch die eben angesprochenen Linkbytes miteinander verknüpft sind. Das DOS liest nun jeweils einen Block der Datei in den Pufferspeicher. Werden mehr Daten benötigt, so wird der nächste Block der Datei anhand der Linker ebenfalls in den Puffer gelesen, und so weiter. Wenn das DOS auf eine Null als Spurnummer stößt, weiß es, daß die Datei zu Ende ist, und hört auf.

6.3.1.2 Der Abstand der Sektoren beim Schreiben

Wie Sie wissen, enthält eine Spur zum Beispiel 21 Sektoren, die der Reihe nach von 0 bis 20 durchnummeriert sind, wobei Sektor 1 neben Sektor 0, Sektor 2 neben Sektor 1, ... und Sektor 0 wiederum neben Sektor 20 liegt; damit ist der Kreis geschlossen.

Wenn die Floppy nun ein Programm von der Diskette lädt, so ist es nicht möglich, die Sektoren direkt von der Diskette zum Computer zu übertragen. Es wird vielmehr der erste Block vollständig in den floppyinternen Pufferspeicher gelesen und erst danach zum Computer geschickt. Dann wird der zweite Block anhand des Linkers ebenfalls gelesen und zum Computer geschickt, und so weiter, bis das gesamte Programm geladen wurde.

Nun benötigt dieses Schicken eines Blocks zum Computer natürlich eine gewisse Zeit, in der sich die Diskette jedoch weiterdreht. Aus diesem Grund wäre es zeitlich nicht sehr klug, wenn der Zeiger des nächsten Blocks direkt auf den physikalisch nächsten Block zeigen würde. Während der Übertragung des einen Sektors würde sich die Diskette nämlich so

weiterdrehen, daß der direkt im Anschluß kommende Sektor erst nach einer ganzen Umdrehung wieder gelesen werden könnte.

Die Erfahrung zeigt jedoch, daß die Übertragung des Blocks vom Pufferspeicher der Floppy zum Computer im 1541-Modus etwa zweieinhalb Diskettenumdrehungen - die Diskette dreht sich 5 mal in der Sekunde, das bedeutet also 1/2 Sekunde - und im l570/7l-Modus etwa eine Viertelumdrehung - das entspricht 1/20 Sekunde - benötigt.

Aus diesem Grund legt man den Folgeblock im l54l-Modus im 10-Sektoren-Abstand und im 1570/71-Modus im 6-Sektoren-Abstand an. Wenn wir also auf der 1571 eine Datei haben, die zufällig bei Sektor 17,0 beginnt, so werden beim Abspeichern die Zeiger auf die Folgesektoren, auf 17,6 und 17,12 und 17,18 und 17,3 und so weiter gelegt, so daß sich die Floppy nach dem Senden eines Blocks ziemlich genau vor dem Folgeblock befindet und diesen sofort - ohne langes Suchen - in den Pufferspeicher lesen kann.

Auch im Directory existiert ein solcher Sektorabstand, der übrigens im englischen mit "sector interleave" bezeichnet wird. Dort wird jedoch meistens nur intern nach einem Dateinamen gesucht, so daß der Sektorabstand hier kürzer gewählt werden kann. Er beträgt in der Regel drei Sektoren, so daß auf Sektor 18,1 der Sektor 18,4 folgt, und so weiter...

Um die Angaben nachprüfen zu können, müssen Sie sich noch ein wenig gedulden. Wir werden nämlich im nächsten Kapitel auf den Direktzugriff eingehen, und dort werden Sie unter anderem einen Diskettenmonitor an die Hand bekommen, der es Ihnen erlaubt, Disketten anzusehen und das Gesagte zu überprüfen. Außerdem werden Sie in die Lage versetzt, auf jedes Byte einer Diskette zuzugreifen, um eigene Eingriffe, wie das Schützen einer Datei vor SC RA TCH, vorzunehmen.

Jetzt aber erst wieder zum Aufbau der einzelnen Dateitypen.

6.3.2 Der Aufbau von Benutzer-Dateien (User-Files, USR)

Dieser Dateityp entspricht, wie schon in Kapitel 5.2 besprochen, prinzipiell der sequentiellen Datei und wird nur anderen Aufgabengebieten zugewiesen. Er besteht, wie auch die sequentielle Datei, nur aus zusammengeketteten Sektoren, wobei jeder Sektor 254 Datenbytes aufnimmt.

6.3.3 Der Aufbau von Programmdateien (PRG)

Auch dieser Dateityp ist der sequentiellen Datei sehr ähnlich. Er
unterscheidet sich lediglich im ersten Block jedes Programms.

Im ersten Block einer Programmdatei haben, im Gegensatz zur sequentiellen Datei, nur 252 Bytes Platz, da weitere 2 Bytes für die Dateiverwaltung benötigt werden. Das sind die Bytes 2 und 3, die in diesem Fall die Startadresse des Programms im Speicher des Computers enthalten (zuerst das Lo- und dann das Hi-Byte). Für die Floppy macht das intern zwar keinen Unterschied (ihr ist "egal", was in welchem Byte außer Byte 0 und 1 steht). Der Computer richtet sich jedoch sehr wohl danach:

Bei LOAD"name",8,1 nimmt der Computer als Programmadresse im Speicher die Adresse, die er bei SAVE zur Floppy geschickt und auf Diskette geschrieben hat, also die Bytes 2 und 3.

Lädt man dagegen mit LOAD"name",8, so ignoriert der Computer die Adresse, die im ersten Sektor der Programmdateien steht und nimmt als Adresse generell 2049 ($0801), da das der BASIC-Startadresse entspricht (beim C64). Beim Commodore 128 ist die BASIC-Startadresse entweder $1c01 (ohne HiRes-Grafic) oder $4001 (mit HiRes-Grafik).

Wichtig ist in diesem Zusammenhang noch: der Computer hat von den Linkbytes der Sektoren (also Byte 0 und 1) "keine Ahnung". Sie werden nur von der Floppy benötigt. Das erste Datenbyte einer Datei ist für den Computer also jeweils das Byte 2.

6.4 Der Aufbau von relativen Dateien (REL)

Wegen der unterschiedlichen Arbeitsweise mit relativen Dateien im Gegensatz zu allen anderen Dateitypen ergibt sich fast zwangsläufig, daß dieser Dateityp auch einen anderen Aufbau besitzt, als beispielsweise eine sequentielle Datei. Dieser Aufbau ist um einiges komplexer und soll im folgenden besprochen werden.

Wie Sie wissen, besteht eine relative Datei aus mehreren Datensätzen, die alle ein Länge bis zu 254 Bytes haben können. Die Längenangabe entspricht dabei genau dem Dateninhalt eines Sektors. Sie schließen daraus ganz richtig, daß ein Datensatz in einer relativen Datei maximal 1 Sektor lang sein darf.

Zur Verwaltung einer relativen Datei benötigt die Floppy jedoch noch einen Zusatz, dem wir uns zuerst widmen wollen.

6.4.1 Die Funktion des Side-Sektors

Bei einer relativen Datei kann, auf jeden beliebigen Datensatz nur durch die Angabe der Nummer dieses Datensatzes zugegriffen werden. Wie weiß

die Floppy aber nun, wo dieser Datensatz auf der Diskette steht? Theoretisch könnte Sie die gesamte relative Datei anhand der Linker in den Date nb löcken nach dem gewünschten Record durchsuchen. Diese Methode wäre jedoch sehr zeitaufwendig und würde den Zeitvorteil der relativen Datenverarbeitung gegenüber der sequentiellen Verarbeitung wieder zunichte machen. Aus diesem Grund geht man einen anderen Weg.

Die Floppy legt neben der Sektorfolge der Datensätze noch einen sogenannten Side-Sektor an.

Wie Sie aus Abschnitt 6.2 wissen, steht bei relativen Dateien im Dateieintrag noch zusätzlich die Angabe der Recordlänge und die Spur- und Sektornummer des ersten Side-Sektor-Blocks.

Ein Side-Sektor ist eine Tabelle, die die Positionen der einzelnen Datensätze auf der Diskette angibt. Das heißt im Klartext, für jeden Datensatz der relativen Datei ist im Side-Sektor eine Spur- und eine Sektornummer abgelegt, die den Block auf der Diskette angibt, in dem der gewünschte Datensatz zu finden ist. Ein Side-Sektor-Block kann dabei bis zu 120 Einträge aufnehmen. Werden mehr Angaben benötigt, so legt die Floppy einen weiteren der 6 möglichen Side-Sektor-Blöcke an. Die Gesamtheit aller dieser Side-Sektor-Blöcke bildet dann den Side-Sektor.

Aus diesen Faktoren errechnet sich auch die maximale Kapazität einer relativen Datei. Sie kann aus 6* 120=720 Datensätzen bestehen. Jeder Datensatz kann dabei bis zu 254 Bytes lang sein. Das entspricht einer Maximalzahl von 182880 Bytes pro relativer Datei.

Tabelle 6.4 zeigt den genauen Aufbau eines solchen Side-Sektors-Blocks.

Byte	Bedeutung
000	Spurnummer des nächsten Side-Sektor-Blocks
001	Sektornummer des nächsten Side-Sektor- Blocks
002	Nummer dieses Side-Sektor-Blocks (0 bis 5)
003	Recordlänge in der Datei
004-015	Spur- und Sektornummern der übrigen angelegten Side-Sektor-Blöcke; sie sind 0, wenn der entsprechende Block nicht angelegt worden ist.
016-255	Verzeichnis der Spur- und Sektornummern für die entsprechenden 120 (oder weniger) Datensätze

Tabelle 6.4 Aufbau eines Side-Sektor-Blocks

Die Spurnummer des folgenden Side-Sektor-Blocks steht natürlich auch hier auf 0, wenn kein weiterer Block vorhanden ist.

6.4.2 Wie die Floppy einen Datensatz findet

Sie wissen nun, daß eine relative Datei im Prinzip aus einer sequentiellen Datei besteht, die noch zusätzlich einen Side-Sektor besitzt. Die Datenblöcke der relativen Datei sind genauso verkettet wie eine sequentielle Datei und besitzen jeweils 254 Datenbytes.

Es stellt sich nun natürlich die Frage, wie die Floppy auf einen gewünschten Datensatz zugreift. Durch den Side-Sektor ist der relative Datenzugriff keine Suchangelegenheit mehr, sondern ein rein mathematisches Problem und Rechnen geht allemal schneller als ein Diskettenzugriff.

Betrachten wir wieder die relative Datei aus Kapitel 5.3. Sie besitzt den Namen ADRESSEN und besteht aus 500 Datensätzen zu je 41 Bytes. Bedingt durch die Länge eines Records, können bei dieser Datei mehr als 6 Datensätze in einem Datenblock untergebracht werden, womit die Anzahl der benötigten Sektoren der Datei auf 81 festlegt ist. Dazu benötigen wir noch einen einzigen Side-Sektor-Block (er faßt theoretisch 120 Datenblöcke) und bekommen die Datei mit 82 Blöcken auf Diskette. Wenn Sie sich das Directory ansehen, finden Sie das bestätigt.

Wir wollen nun auf den 398. Datensatz zugreifen und in diesem Datensatz auf das 4. Byte.

In der Floppy startet jetzt folgende Rechnung:

Ein Datensatz besteht aus 41 Bytes. Wenn wir den 398. Datensatz benötigen, sind das bis dahin 398*41=16318 Bytes. Dazu zählen wir noch 4 Bytes, da der Zeiger auf das 4. Byte positioniert werden soll. Das ergibt nun 16322 Bytes. Da ein Datenblock aus 254 Bytes besteht, errechnet sich die Nummer des betreffenden Blocks aus 16322/254=64 (das genaue Ergebnis beträgt ungefähr 64.259, wobei wir den Nachkommawert später noch benötigen, um die Byteposition im entsprechenden Datenblock festzustellen.).

Unser Datensatz steht also im 64. Datenblock der relativen Datei. Anhand des Dateieintrags im Directory wird nun auf den benötigten Side-SektorBlock zugegriffen. In unserem Fall ist das nur einer. Hätten wir auf den 164. Block zugreifen wollen, so hätte sich der betreffende Side-Sektor-Block aus 164/120=1 (genauer: 1.367) errechnet, was dem zweiten SideSektor- Block entspricht.

Wir müssen den entsprechenden Zeiger auf unseren Datenblock jedoch im ersten Side-Sektor-Block mit der Nummer 0 suchen, wobei sich die Byteposition aus

(64 * 2) + 16 = 144

errechnet: plus 16, da die Tabelle im Side-Sektor-Block ab Byte 16 beginnt; mal 2, da jeder Eintrag in der Tabelle aus 2 Bytes (Spur und Sektor) besteht. Die Spur- und Sektornummer unseres Datenblocks steht also bei Position 144 und 145 in Side-Sektor-Block 0.

Jetzt fehlt nur noch die Byteposition unseres Datensatzes im Datenblock. Dazu nehmen wir das Ergebnis unserer ersten Rechnung (64.259) und holen davon den Nachkommarest (0.259). Aus diesem Wert errechnen wir mit

0,259 * 254 + 2 = 67,786

die Position unseres Bytes im Datenblock, die also 68 beträgt. Das "+2" ist natürlich nötig, da die beiden ersten Bytes des Datenblocks keine Datenbytes, sondern Linker sind.

Mit dieser Rechnung haben Sie die Position des Datensatzes also gefunden. Dies mag im ersten Augenblick zwar kompliziert erscheinen, ist es aber nicht. Es geht in der Praxis noch dazu um einiges schneller, als jedes andere Verfahren, da diese Rechnungen vom DOS der 1570/71 in ein paar Millisekunden erledigt sind. Das Durchsuchen der Diskettenblöcke und Durchzählen der Bytes hätte mehrere Sekunden benötigt.

6.5 Der Aufbau von gelöschten Dateien (Deleted Files, DEL)

Dieser Dateityp hat genaugenommen gar keinen Aufbau, da es sich hier um eine gelöschte Datei handelt. Da das Löschen einer Datei bei der Floppy aber lediglich durch Ändern des Dateityps und Freigeben der Sektoren in der BAM besteht, hat eine DEL-Datei natürlich immer den Aufbau der Datei, die gelöscht wurde. Ändert man den Dateityp "von Hand" wieder auf den vorherigen Wert und führt ein VALIDATE aus, damit die Sektoren der Datei wieder als belegt gekennzeichnet werden, so hat man den gesamten Vorgang des SCRATCH-Befehls wieder rückgängig gemacht.

Eine Besonderheit dieses Dateityps darf aber dennoch nicht unerwähnt bleiben: wird ein SCRATCH durchgeführt, so sucht die Floppy im Inhaltsverzeichnis die betreffende Datei und ersetzt den bisherigen Dateityp (das erste Byte des Eintrags) durch 0 ($00). Das heißt eigentlich, daß es sich nun um eine DEL-Datei handelt, die nicht ordnungsgemäß geschlossen wurde (Bit 7 des Dateityps steht auf 0). Will man sich nun das Inhaltsverzeichnis ansehen, so ist die gelöschte Datei nicht mehr darin vorhanden.

Mit einem Trick kann man die Anzeige einer DEL-Datei jedoch erzwingen. Dazu muß man das Bit 7 des Dateityps setzen (also den Dateityp auf ordnungsgemäß geschlossen stellen; entspricht 128 bzw. $80). Im Directory erscheint die Datei nun wieder ganz normal; jedoch mit der Dateikennung DEL.

Diesen letzten Vorschlag können Sie natürlich im Augenblick noch nicht praktizieren. Er motiviert Sie vielleicht ein bißchen für das folgende Kapitel, in dem wir uns dem Direktzugriff auf eine Diskette zuwenden.

Mit dem Ende dieses Kapitels verlassen Sie nun endgültig die Abschnitte des Buches für den weniger versierten Anwender. Wir steigen nun sehr tief in die Interna der 1570/71 ein und werden Befehle und Geheimnisse entdecken, die Sie Ihrer Floppy im ersten Moment sicherlich nicht zugetraut haben. Erst diese Befehle jedoch machen die Floppy zu einem Gerät, das jede nur erdenkliche Anwendung unterstützen kann.

7 Der Direktzugriff auf die Floppy 1570/71

Wir wollen uns nun mit dem Bereich der beiden Floppystationen beschäftigen, der wohl den größten Unterschied zu herkömmlichen Floppylaufwerken darstellen dürfte. Einige Befehle der 1570/71 haben Sie schon kennengelernt. Diese Befehle stellen jedoch nur einen geringen Teil des tatsächlichen Befehlsvorrates der Floppy dar.

Sie lernen jetzt Befehle kennen, die sich nicht mehr an eine Dateistruktur halten. Diese Befehle erlauben den direkten Zugriff auf die Diskette (auf jedes einzelne Byte) und auf das Betriebssystem der Floppy, das DOS.

Zuerst wollen wir die Befehle behandeln, die sich mit dem Direktzugriff auf die Diskette beschäftigen. Wenn Sie einen solchen Befehl an die Floppy schicken, so muß von der Floppy intern ein Pufferspeicher bereitgestellt und ein Kanal reserviert werden. Das ist nötig, damit die Floppy den zu behandelnden Sektor in den Speicher liest und Sie danach die Inhalte dieses Sektors ungestört bearbeiten können.

7.1 Das Eröffnen eines Direktzugriffs-Kanals

Die folgenden Ausführungen dürften Sie sehr stark an die Arbeit mit Dateien erinnern. Auch hier wird nämlich eine Art Datei eröffnet, die jedoch nicht auf einer Diskette existiert, sondern lediglich für die interne FloppyOrganisation nötig ist. Diese Datei nennt sich Direktzugriffs-Datei und hat auch einen Namen, nämlich das Numerus-Zeichen "#". Wenn dieses Zeichen als Dateiname ankommt, dann stellt die Floppy sofort einen Puffer mit 256 Bytes Kapazität und einen Kanal für den Datenaustausch zur Verfügung.

Die Syntax beim Eröffnen einer Direktzugriffs-Datei lautet:

```
OPEN 1,geräte#,sekundär#,"#puffer"
```

Dazu einige Beispiele:

```
OPEN 1,8,2,"#"
OPEN 3,9,3,"#0"
OPEN 1,8,4,"#3"
```

In der Regel wird die zu bearbeitende Diskette nach dem Einlegen sofort initialisiert. Sie sollte sich also beim Eröffnen der Direktzugriffs-Datei schon im Laufwerk befinden.

Wenn Sie die Syntax des Befehls noch einmal betrachten, so werden Sie feststellen, daß hinter dem Dateinamen noch eine Nummer angegeben werden kann. Mit dieser Nummer kann man der Floppy vorschreiben, welchen Puffer man für den Direktzugriff reservieren möchte (0 bis 3). Diese Angabe sollten Sie aber nur dann verwenden, wenn sie wirklich nötig ist (das kann bei späteren Anwendungen passieren); ansonsten sollten Sie die Zahl weglassen und der Floppy die Wahl des geeigneten Puffers überlassen.

Der Rest des Eröffnens entspricht dem einer ganz normalen Datei, wobei wieder eine Kanalnummer zwischen 2 und 14 gewählt werden kann.

Es sei an dieser Stelle darauf hingewiesen, daß das BASIC 7.0 die Direktzugriffs-Befehle nicht unterstützt. Hier müssen Sie also auf die Ebene des BASIC 2.0 heruntersteigen.

Nun wollen wir uns aber den einzelnen Befehlen zuwenden, wobei zwei Kategorien unterschieden werden müssen: Die BLOCK-Befehle und die MEMORY -Befehle. Die BLOCK-Befehle beziehen sich immer auf einen Sektor der Diskette im Commodore-(GCR)-Format. Anschließend widmen wir uns den MEMORY-Befehlen, die für Anwendungen im Speicher der Floppy geschaffen sind.

7.2 Der BlOCK-READ-Befehl (B-R, U1)

Dieser Befehl erlaubt es Ihnen, einen beliebigen der 683 (1366) vorhandenen Sektoren einer Diskette in den reservierten Pufferspeicher zu lesen, wo er dann von Ihnen bearbeitet werden kann. Wie Sie sehen, können Sie auf sämtliche Blöcke der Diskette zugreifen, also auch auf die Spuren 18 und 53, die die BAM und das Directory beinhalten. Sie haben damit enorme Möglichkeiten, direkt auf die Organisation einer Diskette einzuwirken und diese bei entsprechender Behandlung zur bis absoluten Unkenntnis zu verstümmeln. Seien Sie also besonders vorsichtig, wenn Sie Disketten verwenden, die unter Umständen wichtige Daten beinhalten.

Der BLOCK-READ-Befehl sieht folgendermaßen aus:

```
B-R Kanal# Drive# Spur# Sektor#
```

Da dieser Befehl jedoch eine, für unsere Zwecke unerwünschte, Begleiterscheinung hat (siehe auch "Fehler im DOS der 1570/71"), wird er durch

einen anderen Befehl ersetzt, der sich U1 nennt und dessen Syntax genau die gleiche ist:

```
U1 Kanal# Drive# Spur# Sektor#
```

Hierzu ein paar *Beispiele*:

```
OPEN 1,8,15,"U1 2 0 18 0"
PRINT#2, "U1";Kanal;Drive;Track;Sector
OPEN 8,8,15,"U1 3 8";T;S
```

Wie Sie an den Beispielen erkennen, handelt es sich natürlich auch bei den Direktzugriffs-Befehlen um Befehle, die über den Kommandokanal geschickt werden müssen, wobei jeder Befehl die Angabe der Nummer des Direktzugriffs- Kanals verlangt. (Sie können also auch mehrere gleichzeitig offenhalten.)

Haben Sie den Block im Pufferspeicher, so können Sie dessen Inhalt mit GET# oder INPUT# einlesen, wobei bei INPUT# natürlich wieder auf die Stringlänge zu achten ist (siehe auch 5.2 und 5.3). Das folgende kleine Programm liest die beiden ersten Bytes aus Block 18,0 einer Diskette und zeigt diese an. Wie Sie wissen, befindet sich in Block 18,0 die BAM einer Diskette, wobei die beiden ersten Bytes den Zeiger auf den ersten Block des nachfolgenden Directory enthalten (siehe auch 6.1). Das Ergebnis muß also im Normalfall die Zahlen 18 und 1 liefern.

```
10 OPEN 1,8,2,"#": OPEN 2,8,15
20 PRINT#2,"U1 2 0 18 0"
30 GETB1,T$,S$
40 PRINT ASC(T$),ASC(S$)
50 CLOSE1: CLOSE2
```

Wenn Sie dieses Programm starten, muß folgender Ausdruck auf dem Bildschirm erscheinen, wenn an der eingelegten Diskette nichts manipuliert worden ist:

```
18 1
```

Wie Sie sehen, ist das Einlesen eines Sektors sehr einfach. Es gibt nur ein Problem im Zusammenhang von GET# und ASC.

Der ASC-Befehl stellt die Umkehrung des CHR$-Befehls dar und wandelt demzufolge ein Zeichen wieder in einen Zahlenwert um. Aus dem Zeichen A wird also die Zahl 65, die dem Zeichen entspricht. Es gibt allerdings einen einzigen Zahlenwert, dem kein Zeichen zugeordnet ist. Das ist der Zahlenwert 0, der in der Datenübertragung zwischen verschiedenen Computern eine bestimmte Rolle spielt. Wenn dieser Wert von der Floppy an

den Computer gesendet wird, ist er nicht in der Lage, ein Zeichen in einer der beiden Variablen T$ oder S$ zu übergeben. Die Folge davon ist, daß die entsprechende Stringvariable keinen Inhalt erhält und einen sogenannten Leerstring "" bildet.

Wenn der ASC-Befehl aber wiederum auf einen solchen Leerstring stößt, so ist die Folge ein ILLEGAL QUANTITY ERROR, der in unserem Fall in Zeile 40 auftreten würde.

Aus diesem Grund muß in einer komplexeren Anwendung, in der Nullbytes von der Floppy gelesen werden könnten, ein solcher Fehler verhindert werden. Das kann auf zwei Arten erfolgen:

Sie können einmal eine Abfrage einbauen und auf ein Nullbyte entsprechend reagieren:

```
35 IF T$="" THEN...
```

Andererseits gibt es noch eine viel elegantere Methode:

```
40 PRINT ASC(T$+CHR$(0)), ASC(S$+CHR(0))
```

Wie Sie sehen, wird jedem Zeichen, das von der Floppy kommt, das Nullbyte angehängt. Das ändert am Wert eines ASCII-Zeichens gar nichts; hat jedoch den Vorteil, daß der ASC-Befehl nie auf einen Leerstring stößt, da das CHR$(0) eindeutig einen Wert, nämlich 0 aufweist; auch, wenn dieser Wert nicht als Zeichen in einem String interpretiert werden kann.

Außer dem Problem, das ein Leerstring hervorruft, gibt es noch eine weitere Unzulänglichkeit: Alle 100 davorliegenden Bytes müssen in den Computer eingelesen werden, wenn man den Wert des 101. Bytes erhalten will (siehe auch "sequentielle Datei").

Das muß jedoch nicht unbedingt sein. Für diesen Fall stellt die Floppy einen weiteren Befehl zur Verfügung.

7.3 Der BUFFER-POINTER-Befehl (B-P)

Dieses Kommando ermöglicht den Zugriff auf jedes beliebige Byte (0 bis 255) innerhalb eines Puffers. Sie müssen lediglich die Nummer dieses Bytes angeben:

```
B-P Kanal# Byte#
```

Wollen Sie also wie geschildert auf das 101. Byte zugreifen, so geben Sie folgende Zeile zusätzlich ein:

```
PRINT#2,"B-P 2 181"
```

Die Eigenschaft des Zeigers, den Sie mit diesem Befehl verändern können, sollten Sie sich einprägen. Er wird beim Lesen eines Blocks auf das Byte 0 des Puffers gestellt und erhöht sich bei jedem Lese- oder Schreibzugriff um 1. Wird der Wert 255 ($ff überschritten, so wird automatisch wieder bei 0 angefangen. Schreibzugriffe auf den eingelesenen Sektor erfolgen genau analog zum Lesen: mit dem Positionieren auf das gewünschte Byte und einem Schreibzugriff mit dem PRINT#-Befehl. Hierbei müssen Sie wieder auf die Eigenarten des PRINT#-Befehls achten (siehe Kapitel 5).

7.4 Der BLOCK-WRITE-Befehl (B-W)

Natürlich existiert auch das Gegenstück zum Lesen eines Blocks, denn Sie wollen den geänderten Block selbstverständlich wieder abspeichern. Die Bedienung des B-W-Befehls erfolgt genau analog zum B-R-Befehl; so auch das Ersetzen dieses Befehls durch einen anderen, nämlich den U2- Befehl:

```
U2 Kanal# Drive# Spur# Sektor#
```

Ein paar *Beispiele*:

```
PRINT#1,"U2 2 0 1 0"
OPEN 3,8,15,"U2 2 0"+TRACK+SECTOR
PRINT#3, "U2";2;8;18;1
```

An dieser Stelle eine Anmerkung zu den Spur- und Sektornummern bei den eben besprochenen Befehlen. Für das Commodore-Format sind selbstverständlich die gegebenen Grenzen zu berücksichtigen; das heißt bei der 1570 eine Spurangabe von 1 bis 35 und bei der 1571 eine solche von 1 bis 70. Die Sektorzahlen richten sich dabei nach der entsprechenden Spurnummer und liegen zwischen 0 und 16, 17, 18 und 20.

Anhand der Direktzugriffs-Befehle erfahren Sie auch ganz nebenbei, wie die Floppy auf einer Diskette arbeitet. Sie sehen, daß ein Sektor nie auf der Diskette geändert wird und auch keine einzelnen Bytes geschrieben oder gelesen werden. Es wird immer ein ganzer Sektor in den jeweiligen Pufferspeicher (der für alle Anwendungen gesondert reserviert wird) gelesen, dann abgeändert und wieder auf die Diskette geschrieben.

Dazu noch ein kleines Beispiel für das Schreiben eines neuen Sektors:

```
10 OPEN 1,8,2,"U": OPEN 2,8,15
20 PRINT#1, "DAS IST EIN TEST!!!"
30 PRINT#2,"U2 2 0 1 0"
40 CLOSE1: CLOSE2
```

Hier wird die Zeichenkette "DAS IST EIN TEST!!!" in den reservierten Puffer und anschließend in Sektor 0 auf Spur 1 geschrieben. Da kein Semikolon am Ende des PRINT#-Befehls steht, wird hinter die Zeichenkette automatisch das RETURN-Zeichen 13 bzw. $0d angehängt, so daß der String mit INPUT# wieder eingelesen werden kann.

7.5 Der BLOCK-ALLOCATE-Befehl (B-A)

Wollen Sie künftig im Direktzugriff arbeiten, so darf auch dieser Befehl nicht fehlen. Sie dürfen nie vergessen, daß der Direktzugriff mit einer Datei nichts zu tun hat. Das heißt, alles, was Sie tun, geschieht ohne Einmischung des DOS. Aus diesem Grund werden natürlich die von Ihnen im Direktzugriff geschriebenen Sektoren auch nicht in die BAM eingetragen und im Directory unter einem Dateinamen abgelegt.

Um eine Katastrophe größeren Ausmaßes zu verhindern, existiert daher der B-A-Befehl. Mit ihm ist es möglich, einen Block in der BAM als belegt zu kennzeichnen, so daß zukünftige Schreiboperationen auf der Diskette diesen Sektor nicht überschreiben können. Die Syntax dieses Befehls lautet:

```
B-A Drive# Spur# Sektor#
```

Dazu wieder einige Beispiele:

```
OPEN 1,8,15,"B-A 0 1 0"
PRINT#2,"B-A 0";T;S
```

Mit diesem Befehl können Sie jeden x-beliebigen Block auf der Diskette als belegt markieren. War der Block schon belegt, so geschieht nichts in dieser Hinsicht sondern die Floppy gibt die Fehlermeldung NO BLOCK aus. Zur Erklärung sehen Sie sich bitte im Anhang die Liste der FloppyFehlermeldungen an.

Achtung: Eine Diskette, die durch Direktzugriff geschriebene und belegte Sektoren enthält, darf niemals mit dem VALlDA TE-Befehl (siehe 4.3) behandelt werden. Da die belegten Blöcke nicht im Directory eingetragen und miteinander verkettet (siehe 6.3.1.1) sind, werden sie durch den VALIDATE-Befehl wieder freigegeben und können dadurch später aus Versehen überschrieben werden.

7.6 Der BLOCK-FREE-Befehl (B-F)

Dieser Befehl ist das genaue Gegenstück zum B-A-Befehl. Er ermöglicht das Wiederfreigeben eines belegten Blocks, um ihn für weitere Schreibzugriffe auf der Diskette zur Verfügung zu stellen. Sie können natürlich auch belegte Blöcke noch im Direktzugriff lesen, abändern und schreiben. Das Belegen von Blöcken hat nur den Sinn, daß die Dateiverwaltung des DOS diese Blöcke dann nicht mehr antastet. Ein Wiederfreigeben ist nur für die spätere Anwendung von Dateien sinnvoll und wird folgendermaßen durchgeführt:

```
B-F Drive# Spur# Sektor#
```

Die Anwendung erfolgt analog zum B-A-Befehl und bedarf keiner weiteren Erläuterung.

7.7 Der MEMORY-READ-Befehl (M-R)

Die MEMORY-Befehle beziehen sich direkt auf den Speicher der Floppy (sowohl RAM, als auch ROM und I/O) und stellen keinen Diskettenzugriff dar. Aus diesem Grund ist auch das Eröffnen einer Direktzugriffs-Datei nicht erforderlich. Sämtliche Kommunikation erfolgt jetzt über den Kommandokanal.

Die Syntax des M-R-Befehl lautet:

```
M-R CHR$(ADL) CHR$(ADH) CHR$(Anzahl)
```

Dabei haben die Parameter folgende Bedeutung:

ADL	niederwertiges Byte der Speicheradresse
ADH	höherwertiges Byte der Speicheradresse
Anzahl	die Anzahl der zu lesenden Bytes (optional)

Die Anzahl der zu lesenden Bytes kann auch weggelassen werden. In diesem Fall wird nur ein Byte ausgegeben.

Einige *Beispiele*:

```
OPEN 1,8,15,"M-R"+CHR$(0)+CHR$(3)+CHR$(10)
PRINT#3,"M-R"CHR$(10)CHR$(0)
```

Da die 1570/71 einen Adreßraum von 65536 Byte besitzt, wir jedoch in einem Byte nur die Nummern 0 bis 255 unterbringen können, muß die Adreßangabe auf zwei Bytes aufgeteilt werden; das niederwertige Byte (ADL) und das höherwertige Byte (ADH). Die beiden Werte aus der Adresse A errechnen sich dabei wie folgt:

ADH = INT(A/256)

ADL = A AND 255

Diese Rechenmethode müßte Ihnen noch aus Kapitel 5.3.2 geläufig sein, wo wir unter BASIC 2.0 auf die gleiche Art die gewünschte Recordnummer zerlegt haben. Mit zwei Bytes können Sie nämlich den Bereich adressieren, den Sie benötigen: 0 bis 65535 ($0000 bis $ffff).

Nun wollen wir einmal den Inhalt der Speicherstelle 500 ($01f4) auslesen. Hierzu rechnen wir:

ADH = INT(500/256) = 1

ADL = 500 AND 255 = 244

Danach können Sie sich die Speicherstelle mit folgenden Zeilen ansehen:

```
10 OPEN 1,8,15
20 PRINT#1,"M-R"CHR$(244)CHR(1)
30 GET#1,A$
40 PRINT ASC(A$+CHR$(0))
50 CLOSE 1
```

Wie Sie sehen, werden alle aus dem Speicher gelesenen Werte direkt über den Kommandokanal an den Computer weitergegeben. Der M-R-Befehl entspricht also gewissermaßen dem PEEK-Befehl für die Floppystation.

7.8 Der MEMORY-WRITE-Befehl (M-W)

Nach dem PEEK-Befehl wünschen wir uns natürlich auch den POKE-Befehl für die Floppystation.

Mit diesem Befehl können auf einmal bis zu maximal 34 Bytes in den Speicher der Floppy geschrieben werden. Wohin Sie diese Bytes schreiben, bleibt Ihnen überlassen und führt garantiert früher oder später zum Absturz der Floppy. Sie sollten sich deshalb im nächsten Kapitel noch ein wenig intensiver mit der Bedeutung der unterschiedlichen Speicherbereiche beschäftigen. Jetzt kommen wir aber zur Syntax des M-W-Befehls:

```
M-W CHR$(ADL) CHR$(ADH) CHR$(Anzahl) CHR$(DATA1) CHR$(DATA2) ...
```

Die drei ersten Parameter dürften Ihnen dabei schon bekannt sein; neu sind die restlichen Bytes. Es handelt sich hierbei um die Datenbytes, deren Anzahl vom dritten Parameter (Anzahl) abhängt. Hier geben Sie also nicht an, wieviele Bytes Sie lesen wollen, sondern vielmehr, wieviele Bytes in den Speicher der Floppy geschrieben werden sollen. Es können maximal 34 Bytes auf einmal angegeben werden, da der Eingabepuffer der Floppy nicht mehr Zeichen aufnehmen kann.

Natürlich wollen Sie auch ein Anwendungsbeispiel für diesen Befehl sehen. Wie Sie wissen, können Sie bei der 1570/71 die Gerätenummer durch Schalter im Bereich von 8 bis 11 einstellen. Das kann aber auch softwaremäßig, also mit einem Programm geschehen. Im Fall der 1570 ersparen Sie sich so das Aufschrauben des Gehäuses. Die Nummer sei in der Variablen N zu übergeben und betrage zwischen 8 und 11:

```
OPEN 1,8,15
PRINT#1,"M-W" CHR$(119)CHR$(0)CHR$(2)CHR$(32+N)CHR$(64+N)
CLOSE 1
```

Die so eingestellte Nummer bleibt bis zu einem RESET oder dem Ausschalten der Floppy erhalten. Wie Sie sicherlich bemerken, müssen an die Floppy zwei Nummern übergeben werden (einmal in Speicherstelle 119 ($77) und einmal in 120 ($78). Das liegt daran, daß die Floppy sowohl für das Empfangen von Daten, als auch für das Senden von Daten jeweils eine eigene Nummer besitzt.

7.9 Der MEMORY-EXECUTE-Befehl (M-E)

Dieser und die noch folgenden Befehle dieses Kapitels stellen uns auf eine neue Stufe in der Programmierung der Floppystation. Es geht nun darum, analog zum SYS-Befehl im Computer, Maschinenprogramme in der Floppy ausführen zu lassen. Alle diejenigen, die sich noch nie mit Maschinensprache befaßt haben, sollten sich nicht scheuen, sich möglichst bald damit zu beschäftigen, um in den Genuß der nun folgenden, sehr leistungsfähigen, Befehle zu kommen. Wie auch der Computer, besitzt die 1570/71 einen eigenen Mikroprozessor. Es handelt sich hierbei um den Typ 6502, dessen Assemblersprache vollkommen identisch mit der des C64 oder C128 ist.

Wenn Sie die Maschinensprache noch nicht beherrschen, so müssen Sie das Buch nun nicht gleich zur Seite legen. Wir werden nämlich noch Themen

besprechen, die Ihnen zumindest schon einen kleinen Einblick in die Funktionsweise eines Prozessorsystems geben. Dadurch wird Ihnen der Einstieg vielleicht ein wenig erleichtert.

Sind Sie der Maschinensprache jedoch mächtig und haben Sie auch schon Erfahrung gesammelt, so sollten Sie sich schon jetzt einmal auf die zwangsläufig kommende Bearbeitung des dokumentierten ROM-Listings einstimmen. Beim Ausführen von Maschinenprogrammen im Floppyspeicher - und das werden Sie am Ende dieses Kapitels können - ist es jedoch unumgänglich, die internen Vorgänge der 1570/71 genau zu kennen. Sie müssen wissen, wie die Schreib- und Leseroutinen arbeiten und wie das Betriebssystem der Floppy organisiert ist. Dann und nur dann werden Sie in der Lage sein, erfolgreich eigene Maschinenprogramme in der Floppy zu starten und eventuell eigene Diskettenformate, Kopierschutzmethoden, schnelle Kopierprogramme usw. zu entwickeln.

Der M-E-Befehl ist dafür die wichtigste Voraussetzung. Er entspricht, wie schon erwähnt, dem SYS-Befehl im Computer und startet ein Maschinenprogramm im Floppyspeicher an der angegebenen Adresse. Dieses Maschinenprogramm kann auch durchaus eine ROM-Routine sein oder ROMRoutinen enthalten. Die einzige Bedingung ist ein Abschluß durch den Befehl RTS.

Die Syntax des M-E-Befehls lautet:

```
M-E CHR$(ADL) CHR$(ADH)
```

wobei ADL und ADH wieder das nieder- und höherwertige adressbyte bezeichnen.

Als Anwendungsbeispiel mag das Blinken der LED am Laufwerk im Fehlerfall dienen. Sorgen Sie dafür, daß die LED nicht brennt und geben Sie danach die folgenden Zeilen ein:

```
OPEN 1,8,15
PRINT#1,"M-E" CHR$(44)CHR$(193)
CLOSE 1
```

Sofort beginnt die LED der Floppy munter drauflos zu blinken. Mit dem Befehl haben Sie die Routine im DOS aufgerufen, die für das Einschalten des Blinkens bei auftretenden Fehlern verantwortlich ist.

7.10 Der BLOCK-EXECUTE-Befehl (B-E)

Hier haben wir noch einen BLOCK-Befehl. Obwohl diese Befehlsart eigentlich vor den MEMORY-Befehlen besprochen wurde, war es Absicht, diesen Befehl erst im Anschluß an den M-E-Befehl zu besprechen. Vielleicht fällt Ihnen schon die Ähnlichkeit der beiden Befehlsnamen auf?

Der B-E-Befehl dient dazu, einen Block von der Diskette in den Pufferspeicher der Floppy zu lesen und danach automatisch das darin enthaltene Maschinenprogramm auszuführen. Natürlich muß auch für diesen Befehl eine Direktzugriffs- Datei (siehe 7.1) eröffnet werden:

```
B-E Kanal# Drive# Spur# Sektor#
```

Hier wird nur die Sektornummer des Blocks angegeben. Der Block wird dann automatisch gelesen und das Programm ab dem Byte 0 des Inhalts ausgeführt.

Wenn Sie mit diesem Befehl arbeiten, müssen Sie darauf achten, daß Maschinenprogramme selten relokatibel (frei verschieblich) sind. In der Regel müssen sie bei der Ausführung immer im gleichen Adreßbereich stehen. In diesem Fall ist also die Angabe einer festen Puffernummer beim Eröffnen der Direktzugriffs-Datei nicht nur zweckmäßig, sondern unter Umständen sehr wichtig, um einen Absturz des Systems zu vermeiden.

Es wird zwar an dieser Stelle schon von Puffern und Adreßbereichen gesprochen; wir werden jedoch erst im nächsten Kapitel genauer darauf eingehen. Sie erfahren dann alles Wichtige über die Speicheraufteilung der 1570/71.

7.11 Die Benutzer-Befehle (USER-Befehle U3 bis U:)

Die V-Befehle führen, wie schon die vorangegangenen Befehle, Maschinenprogramme im Speicher der Floppystation aus. Es handelt sich jedoch bei diesem Befehlstyp um eine Vereinfachung beziehungsweise Abkürzung des M-E-Befehls, da einige wichtige Startadressen schon voreingestellt sind und nur noch der Befehl - ohne Parameter - übergeben werden muß (zum Beispiel: PRINT#1,"U:").

Zwei U-Befehle haben Sie in den Abschnitten 7.2 und 7.4 schon kennengelernt, nämlich U1 und U2. Der U0-Befehl übernimmt in den Floppystationen 1570 und 1571 ganz neue Aufgaben und wird von Commodore im Handbuch nicht einmal erwähnt. Wir werden diesen vielleicht wichtigsten Systembefehl der 1570/71 später kennenlernen.

Wie schon gesagt, stellen die U-Befehle gewissermaßen Abkürzungen dar, da sie festeingestellte Ansprungadressen zugeordnet bekommen haben. Tabelle 7.1 zeigt diese Adressen:

USER - Befehle	Startadresse des Maschinenprogramms
U3 oder UC	Start bei Adresse $0500 (Benutzerpuffer)
U4 oder UD	Start bei Adresse $0503 (Benutzerpuffer)
U5 oder UE	Start bei Adresse $0506 (Benutzerpuffer)
U6 oder UF	Start bei Adresse $0509 (Benutzerpuffer)
U7 oder UG	Start bei Adresse $050c (Benutzerpuffer)
U8 oder UH	Start bei Adresse $050f (Benutzerpuffer)
U9 oder UI	$ff0l; Betriebsmodus der Floppy umschalten
U: oder UJ	$eaa0; RESET ausführen

Tabelle 7.1 Die Benutzer- Befehlsadressen

Wie Sie aus der Tabelle ersehen, gibt es jeweils zwei U-Befehle für den gleichen Zweck. Wir werden uns in Zukunft an die ZahlendarsteIlung wie auch schon bei U1 und U2 halten, die jederzeit durch UA und UB ersetzt werden kann.

Da der Befehlssatz der Floppy die Ablage und Ausführung von Maschinenprogramme in der Floppy erlaubt, wäre es sehr verwunderlich, wenn es nicht auch einen Speicher bereich gäbe, in dem diese Programme abgelegt werden können, ohne mit dem DOS in Konflikt zu kommen. Dieser Speicherbereich existiert tatsächlich und befindet sich bei der Floppy an den Adressen $0500 bis $05ff. Um eventuell in diesem Bereich abgelegte Maschinenprogramme sehr leicht aufrufen zu können, stellt das DOS die Befehle U3 bis U8 zur Verfügung. Sie stellen jeweils einen Einsprungpunkt dar, wobei die einzelnen Einsprungadressen jeweils drei Bytes auseinanderliegen. Das erlaubt das Anlegen einer Sprungtabelle.

Haben Sie also ein Maschinenprogramm, das ab $0500 im Floppyspeicher abgelegt ist, so genügt zum Starten die folgende Zeile:

```
OPEN 1,8,15,"U3": CLOSE 1
```

Wie Sie vielleicht wissen, war das erste Einzellaufwerk mit einem seriellen Bus von Commodore die schon legendäre Floppystation VC1540. Diese Floppystation wurde für den VC20 Computer entwickelt. Kurze Zeit später kam der C64 auf den Markt. Dieser Computer besitzt jedoch intern eine völlig andere Struktur als der VC20 und demzufolge auch ein anderes Betriebssystem mit anderen Routinen für den seriellen Bus. Insgesamt ist der C64 etwas langsamer als der VC20, was die Entwicklung einer neuen Flop

pystation nötig machte. Es kam die 1541. Um jedoch die Produktion der 1540 ohne Marktverlust einstellen zu können, wurde die 1541 so konstruiert, daß sie sowohl an den C64 als auch an den VC20 angeschlossen werden kann. Sie wurde mit einem Befehl versehen, der eine Umschaltung zwischen VC20- und C64-Betrieb ermöglichte.

Dieser Befehl ist der U9- oder UI-Befehl. Da das DOS 3.0 der 1570/71 das "alte" DOS 2.6 der 1541 zusätzlich fast unverändert enthält, ist auch dieser Befehl noch verfügbar, der den Betrieb an einem VC20 ermöglicht. Beim Einschalten ist die Floppystation automatisch im C64/C128-Modus. Erhält sie den Befehl "U9-", so wird auf VC20-Betrieb umgeschaltet. Bei "U9+" wird wieder der Standardbetrieb für C64 und C128 gewählt.

Der letzte U-Befehl unserer Reihe ist der U:-Befehl. Er bewirkt einen Sprung zur RESET-Routine der 1570/71 und versetzt die Floppy in den Einschaltzustand, der sonst durch Aus- und wieder Einschalten erreicht werden kann.

7.12 Der &-Befehl (Utility Loader)

Dieser Befehl ist im Handbuch zur Floppystation zwar kurz erwähnt; jedoch kaum so beschrieben, daß Sie sich eine &-Datei selbst herstellen könnten, ohne das DOS- Listing gewälzt zu haben.

Wenn man mit dem &-Befehl arbeiten will, so sind einige etwas kompliziert anmutenden Regeln zu beachten. Er ist jedoch für die Programmierung der Floppy in Maschinensprache so interessant, daß wir sofort darauf eingehen.

Der &-Befehl gleicht in seiner Arbeitsweise stark dem BLOCK-EXECUTE-Befehl. Auch hier wird ein Programm von der Diskette in den Speicher der Floppy geladen und anschließend sofort ausgeführt. Es bestehen jedoch drei generelle Unterschiede zum B-E-Befehl:

1. Der &-Befehl arbeitet nicht mit einer Direktzugriffs-Datei, das heißt er benötigt keinen reservierten Pufferbereich oder eine Kanalnummer.
2. Es handelt sich bei einem Programm für den &-Befehl nicht nur um einen Block auf der Diskette, sondern um eine ganze Datei, die im weiteren Verlauf als &-Datei bezeichnet wird, ordnungsgemäß im Directory eingetragen ist und durchaus aus mehreren Blöcken bestehen kann.

3. Der &-Befehl ist nicht auf einen Pufferspeicher angewiesen. Eine &-Datei kann an einer beliebigen Stelle im Speicher der 1570/71 anfangen und an einer anderen beliebigen Stelle wieder aufhören. Es ist sogar möglich, eine &-Datei aus mehreren Maschinenprogrammen zusammenzusetzen, die an ganz verschiedenen Speicherstellen stehen. Das können die Zeropage (siehe Kapitel 8) oder der BAM-Puffer sein.

Die Verwaltung einer &-Datei geschieht dabei folgendermaßen. Sie hat generell den Dateityp USR (siehe Kapitel 5.2) und als erstes Zeichen im Dateinamen das &-Zeichen; also beispielsweise &TEST.

Soll eine solche &-Datei aufgerufen werden, so reicht es, wenn der Dateiname über den Kommandokanal geschickt wird:

```
OPEN 1,8,15,"&TEST"
```

Die Floppy sucht nun nach dem Dateinamen im Inhaltsverzeichnis und lädt das Programm, wenn es gefunden wird, an die Adresse(n), die in der Datei angegeben sind. Wenn wir uns jetzt den Aufbau einer &-Datei ansehen, so sagt das natürlich nicht, daß USR-Dateien generell so aussehen. Wie Sie aus 5.2 wissen, kann eine USR-Datei die verschiedensten Daten enthalten.

Byte	Bedeutung
000	Spurnummer des Folgeblocks, wenn vorhanden
001	Sektornummer des Folgeblocks
002/003	Startadresse des Programms (ADL/ADH) im Speicher
004	Anzahl der Bytes des Programms (maximal 255)
005-...	eigentliches Programm
x	letztes Programmbyte
x+l	Prüfsumme über den vorigen Programmteil
x+2/x+3	Start adresse des nächsten Programms im Speicher
x+4	Anzahl der Bytes des Programms (maximal 255)
x+5-...	eigentliches Programm
y	letztes Programmbyte
y+l	Prüfsumme über den vorigen Programmteil
y+2/y+3	Start adresse des nächsten Programms im Speicher
...	...

Tabelle 7.2 zeigt den Aufbau einer &- Datei

Wie Sie sehen, können Sie in einer &-Datei theoretisch beliebig viele Programme oder Programmteile aneinanderketten, wobei jedes unabhängig vom vorhergehenden und in einem völlig anderen Speicherbereich stehen kann. Die einzige Einschränkung besteht in der Längenangabe der einzelnen Programteile, die nicht länger als 256 Bytes sein können, da der Wert in einem einzigen Byte untergebracht ist.

Das aufwendigste an der Arbeit mit &-Dateien ist deren Herstellung, da diese vom DOS nicht unterstützt wird. Dabei gestaltet sich die Berechnung der Prüfsumme als etwas schwierig und ungewohnt.

Bei der Prüfsummenberechnung werden alle Programmbytes nacheinander aufaddiert und dabei der Übertrag jeder Addition ebenfalls dazugezählt. Die beiden Bytes der Startadresse müssen auch mit addiert werden. Wir betrachten die Prüfsummenberechnung an folgendem Beispiel:

Sie haben das niederwertige Byte der Adresse, zum Beispiel den Wert $90. Dazu soll jetzt das höherwertige Byte addiert werden. Nehmen wir einmal an, daß dieses höherwertige Byte $aa beträgt. Wir rechnen dann $90+$aa=$3a und das Carry-Bit, da ein Übertrag aufgetreten ist. Nach dem Schema der Prüfsummenberechnung muß der Übertrag noch zum Ergebnis addiert werden; wir erhalten also $3a+$01=$3b. So verfahren Sie mit jedem weiteren Programmbyte, bis das letzte inklusive Übertrag addiert worden ist. Das Ergebnis, also die fertige Prüfsumme, wird hinter dem letzten Programm byte abgespeichert.

Achtung! Die Prüfsumme ist in der Gesamtanzahl der Bytes eines Programmabschnitts nicht enthalten.

Wollen Sie die Prüfsumme in Maschinensprache berechnen, so ist das ganz einfach:

```
lda letztes Ergebnis  ;ist am Anfang 0
clc                   ;Übertrag löschen
adc neues Byte        ;neuen Wert addieren; Übertrag Carry
adc #$00              ;Übertrag addieren; 0 oder 1
sta neues Ergebnis    ;neuen Wert merken
rts                   ;Ende
```

Diese Routine befindet sich fast identisch im ROM der 1570/71 an der Adresse $e84b. Das DOS verwendet sie zur Kontrolle auf korrekte Prüfsummen in einer &- Datei; sie kann natürlich auch zur Herstellung der Prüfsumme verwendet werden.

Wie Sie sehen, ist das DOS sehr pedantisch, was den Aufbau einer &-Datei angeht. Haben Sie jedoch erst einmal gelernt, mit &-Dateien zu arbeiten. so werden Sie sehr schnell feststellen, daß es sich hier um eine äußerst leistungsfähige Methode zur Erweiterung des DOS-Befehlssatzes handelt. Sie werden in die Lage versetzt, ganze Diskettenbetriebssysteme zu schreiben, die, ähnlich dem CP/M Betriebssystem, eigene Systemspuren auf der Diskette aufweisen.

Am Schluß sei noch auf ein paar seltsame Fehlermeldungen der Floppystationen hingewiesen. Diese Fehlermeldungen wurden den relativen Dateien entliehen und haben folgende Bedeutung:

RECORD NOT PRESENT erscheint, wenn Sie die Prüfsumme nicht richtig ausgerechnet haben.

OVERFLOW IN RECORD erscheint, wenn die Anzahl der Programmbytes nicht mit der angegebenen übereinstimmt.

7.13 Der BOOT-Befehl (BASIC 7.0)

Dieser Befehl gehört eigentlich nicht in die Reihe der Direktzugriffs-Befehle, weil er erstens kein diskketteninterner Befehl ist und zweitens eine gesamte Datei behandelt. Er paßt jedoch an diese Stelle, da es sich auch hier um das automatische Laden und Starten eines Maschinenprogramms handelt:

Der BOOT-Befehl ist in BASIC 2.0 nicht verfügbar, das heißt er funktioniert nur auf einem Commodore 128 Computer. Hier lädt er entweder eine angegebene oder eine spezielle AUTOBOOT -Datei. Hier zunächst die Syntax des BOOT -Befehls:

```
BOOT "name",Ddrive#,Ugeräte#
```

Die Parameter name, drive# und geräte# erklären sich von selbst. Hier sind
ein paar Beispiele:

```
BOOT
BOOT "TESTPROGRAMM"
BOOT "TEST",D0,U8
```

Wird ein Programmname angegeben, so lädt der Computer automatisch das entsprechende Programm als Maschinenprogramm in den Bereich, aus dem es abgespeichert wurde (siehe auch BLOAD). Danach wird das Programm an der gegebenen Adresse gestartet (mit SYS, weshalb ein BOOT-Befehl grundsätzlich ein sinnvolles Maschinenprogramm nachladen sollte, da der Rechner sonst abstürzt).

Wir haben schon einmal festgestellt, daß der C128 beim Einschalten immer einen Diskettenzugriff versucht, wenn die Floppy vorher eingeschaltet war. Dieser Zugriff ist nichts weiter als ein BOOT-Befehl ohne Parameter, also auch ohne angegebenen Dateinamen. In diesem Fall passiert folgendes:

Die Floppy sucht auf der Diskette den Sektor 0 auf Spur 1 und liest ihn in ihren Pufferspeicher. Danach wird kontrolliert, ob dieser Sektor eine entsprechende Kennung in den Bytes 0 bis 2 enthält, die ihn als AUTOBOOTSektor ausweisen. Steht in diesen drei Bytes die Kennung "cbm", so wird automatisch das Maschinenprogramm geladen und gestartet, das im AUTOBOOT-Sektor angegeben ist. Den Aufbau dieses Sektors zeigt folgende Tabelle:

Byte	Bedeutung
000-002	CBM ($43,42,4d) Bytes für die Kennung
003-004	Startadresse eines eventuell vorhandenen Programms
005	MMU-Registerwert für Speicherkonfiguration
006	Anzahl der folgenden BOOT-Sektoren
007-...	Name für Meldung: "BOOTING ..."; Ende durch $00
xxx-...	Name des zu bootenden Programms; Ende durch $00
yyy-255	Ladeprogramm für Kassettenpuffer, falls eigene Laderoutinen erwünscht sind.

Tabelle 7.3 Inhalt eines AUTOBOOT -Sektors (Sektor 1,0)

Die Bytes für die Startadresse haben nur Gültigkeit, wenn noch einer oder mehrere weitere BOOT -Sektoren folgen. Die Anzahl dieser Sektoren ist dann in Byte 6 abgelegt; ansonsten steht dort $00. Wenn weitere Sektoren verwendet werden, so benötigt das DOS dafür keine weiteren Angaben. Es sind dies jeweils die folgenden Sektoren auf Spur 1 (die Sektoren 1,1 / 1,2 usw.).

Der Name ab Byte 7 wird beim Booten des Programms auf dem Bildschirm ausgedruckt. Er wird durch ein $00 beendet und kann deshalb auch mehrere Zeichen haben, die vielleicht eine Mitteilung an den Benutzer enthalten.

Wird die Steuerung des weiteren Ladevorgangs dem Computer überlassen, so benötigt dieser die Angabe eines Dateinamens hinter dem String für die Bildschirmanzeige, damit er weiß, welches Programm geladen werden soll. Ansonsten kann ein eigenes Programm die Steuerung übernehmen, das automatisch im Kassettenpuffer abgelegt wird und im AUTOBOOT-Sektor hinter dem, jetzt nicht angegebenen, Dateinamen steht.

Das Booten des CP/M-Betriebssystems funktioniert zum Beispiel auf die eben beschriebene Art und Weise. Liegt die CP/M-Diskette schon bei einem RESET des Computers im Laufwerk, so erfolgt das Laden automatisch; ansonsten kann das Booten durch Tippen des Befehls BOOT jederzeit erfolgen.

8 Die Hardwareorganisation der 1570/71

Die Begriffe "Pufferspeicher der Floppy" und "Programm im Speicher der Floppy" sind schon des öfteren erwähnt, aber noch nicht genauer erklärt worden. Im folgenden Kapitel werden Sie nun über alle hardwaremäßigen Vorrichtungen der Floppy informiert, die Sie bei der Programmierung beachten oder verwenden müssen.

8.1 Die Speicherorganisation der 1570/71

Zuerst befassen wir uns mit dem Speicherbereich der Floppy. Dies Kapitel ist natürlich vor allem für die Maschinensprache-Programmierer unter Ihnen wichtig, weshalb die Erläuterungen sehr ausführlich gehalten sind. Wenn Ihnen also manche Erklärung überflüssig oder gar ein wenig lächerlich vorkommt, so gehören Sie eben schon zu den erfahreneren Programmierern und sollten nachsichtig darüber hinwegsehen.

Die 1570/71 Floppystationen haben die gleichen Prozessorplatinen in ihren Gehäusen, nämlich die der 1571. Diese Platine enthält als herausragende Merkmale folgende Bausteine: einen Prozessor vom Typ 6502, zwei VIA 6522, einen CIA 6526, einen MFM-Diskcontroller WD 1770, ein statisches RAM mit 2 KByte Kapazität und ein 32 KByte ROM, sowie einen Commodore-Diskcontroller in Form eines Gate-Arrays.

Diese Bausteine wollen wir im weiteren Verlauf des Buches unter die Lupe nehmen, wobei wir mit dem Speicherbaustein, dem 2 KByte fassenden RAM, anfangen.

8.1.1 Der Bereich der Zeropage

Wie Sie vielleicht wissen, kann ein Mikroprozessor vom Typ 6502 einen Bereich von 65536 Bytes direkt adressieren. Das heißt, er kann auf jedes dieser Bytes direkt zugreifen. Die Adressierung dieser Bytes erfolgt dabei durch zwei spezielle prozessorinterne Bytes, die zusammen als ProgrammCounter (kurz PC) bezeichnet werden. Das eine dieser Bytes ist dabei das niederwertige Adreßbyte, das andere das höherwertige.

Das niederwertige Adreßbyte steuert dabei den Zugriff auf ein Byte innerhalb eines 256-Byte-Blocks, der vom höherwertigen Byte des PC bestimmt wird. Da jedes Byte einen Wert von 0 bis 255 annehmen kann, erstreckt sich der Adreßraum von 0 bis 65535.

Die ersten 256 Bytes (Adresse $0000 bis $00ff) werden als Null-Seite, beziehungsweise Zeropage bezeichnet. Dieser Bereich dient in allen 6502-Prozessorsystemen als Zwischenspeicher-Bereich für wichtige Daten und Zeiger; so auch in der Floppystation. Es kann daher fatale Folgen haben, wenn Sie achtlos Werte an wichtige Speicherstellen dieses Bereichs schreiben und damit das DOS durcheinander bringen. Im Anhang finden Sie aus diesem Grund die Belegung aller Speicherstellen im RAM der 1570/71!

8.1.2 Die Seite 1

Auch dieser Speicherbereich (Adresse $0100 bis $01ff), der sich direkt an die Zeropage anschließt, bildet einen wichtigen Bereich für Zeiger, Zähler und Daten. Er enthält außerdem den Stapelspeicher (Stack) des Prozessors, auf dem Sprungadressen und Registerwerte zwischengespeichert werden.

Auch in der Seite 1 kann ein Fehlgriff fatale Folgen haben. In der Regel ist eine dieser fatalen Folgen der "Absturz" der Floppystation.

8.1.3 Die Seite 2

Dieser Speicherbereich ($0200 bis $02ff) ist der erste etwas harmlosere Bereich, der von der Floppy für Fehlermeldungen und das Zwischenspeichern des Befehlsstrings vom Computer verwendet wird. Reicht einem erfahrenen Anwender der Pufferspeicher der Floppy für seine Zwecke nicht aus, so kann er unter Umständen einen Teil der Seite 2 für seine Zwecke mitverwenden, ohne einen "Absturz" zu verursachen. Das ist natürlich nur bei der Ausführung eines floppyinternen Maschinenprogramms möglich.

8.1.4 Der Bereich der Pufferspeicher der 1570/71

Jetzt kommt endlich der Speicherbereich, der in der Hauptsache für Aufgaben des Anwenders reserviert ist. Er erstreckt sich von $0300 bis $07ff, wobei jedoch die unterschiedlichen Aufgaben der einzelnen Pufferbereiche zu berücksichtigen sind.

Tabelle 8.1 zeigt noch einmal die Belegung des gesamten Speicherbereichs der 1570/71.

Wie Sie sehen, hat jeder der Pufferspeicher 0 bis 4 eine bestimmte Funktion. Wenn Sie jedoch als Anwender ein Maschinenprogramm im Puffer ablegen, so haben Sie gewöhnlich auch die Kontrolle über den Diskettenbetrieb. Rufen Sie also in Ihrem Maschinenprogramm keine Betriebssystemroutinen des DOS auf, die entsprechende Pufferzugriffe benötigen, so ist es Ihnen ohne weiteres möglich, Ihren Arbeitsbereich über den Puffer 2 hinaus auszudehnen und die Puffer für eigene Zwecke zu benutzen.

Byte	Bedeutung
$0000-00ff	Zeropage; Arbeitsspeicher des DOS
$0100-0145	Stackbereich des Prozessors
$0146-0lff	Pufferbereich für zweiseitigen Diskbetrieb; Ausweichpuffer für Disketten-Operationen
$0200-02ff	Pufferbereich des DOS für Fehlermeldungen, Directory und Befehlsstrings
$0300-03ff	Puffer 0: enthält immer den aktuellen Block einer Datei, das gerade bearbeitet wird.
$0400-04ff	Puffer 1: enthält beim Suchen einer Datei den Teil des Directory, der die gesuchte Datei beinhaltet.
$0500-05ff	Puffer 2: USER-Puffer. Auf diesen Bereich beziehen sich unter anderem die U-Befehle. Er wird vom DOS für den Anwender freigehalten.
$0600-06ff	Puffer 3: Directory-Puffer. Enthält immer den aktuellen Block des Directory für die weitere Bearbeitung.
$0700-07ff	Puffer 4: BAM-Puffer. Enthält immer die aktuelle BAM der eingelegten Diskette. Dieser Bereich überschneidet sich bei der 1570/71 mit dem Puffer 5, der eigentlich hardwaremäßig nicht mehr existiert.

Tabelle 8.1 Die RAM-Belegung der 1570/71

Durch die Tabelle wird Ihnen jetzt vielleicht auch die Bedeutung der Puffernummern beim Eröffnen einer Direktzugriffs-Datei klar (siehe Kapitel 7.1). Geben Sie dort zum Beispiel die Puffernummer 0 als gewünschten Pufferbereich an, so wird der Speicherbereich $0300 bis $03ff für Ihre Zwecke reserviert. Sie können dort also zum Beispiel ein Maschinenprogramm ablegen, das unter dem Adreßbereich $0300 läuft. Theoretisch können Sie bis zu 1,5 KByte Speicherbereich in der Floppy verwenden, wenn Sie geschickt programmieren.

8.1.5 Der ROM-Bereich der 1570/71

Über diesen Bereich gibt es an dieser Stelle wenig zu sagen, da er später ohnehin ausführlich dokumentiert wird. Wichtig zu wissen ist der Adreßbereich dieses Speichers, der von $8000 bis $ffff reicht.

Wenn Sie das ROM-Listing in diesem Buch durcharbeiten, werden Sie bestimmt feststellen, daß im ROM der 1570/71 eine ganze Menge Platz ist. Es ist zum Beispiel der Bereich von $aa3f bis $beff vollkommen leer (mit Sff aufgefüllt). Das sind über 5 KByte.

In diesem Bereich lassen sich eine Menge Floppyerweiterungen und eigene Systemroutinen unterbringen - und zwar auf sehr einfache Weise. Die

ROM-Bausteine der 1570/71 sind nämlich vollkommen kompatibel zur entsprechenden EPROM-Version (27256). Sie müssen das jetzige Betriebssystem lediglich in ein 27256-Eprom brennen, wobei Sie zuvor eventuell selbstentwickelte Routinen hinzugefügt haben. Dieses Eprom können Sie direkt als Ersatz für das ROM in den entsprechenden Sockel setzen (ohne Adaptersockel!).

Die Anwendungen hierfür reichen weit: vom eingebauten Treiber für ein schnelles Kopierprogramm oder einem Druckerspooler, bis zu völlig neuen Befehlen für die Floppystation (zum Beispiel RE-SCRA TCH, REPLACE ohne Fehler, Codeschloß zum Schützen von "geheimen" Disketten, ...).

8.2 Die Schnittstellenbausteine der 1570/71

Nun wollen wir uns die Verbindung des DOS mit der Außenwelt ansehen. Wie eingangs schon erwähnt, besitzt die 1570/71 eine ganze Reihe von Schnittstellenbausteinen, die nachfolgend besprochen werden.

8.2.1 Der VIA 6522 als Disk- und Buscontroller-Interface

Der VIA 6522 (versatile interface adaptor) ist ein schon etwas älterer Interface-Baustein der Firma MOS. In der 1570/71 sind jeweils zwei davon enthalten, wobei einer für die Verbindung zur Außenwelt, den seriellen Bus, und der andere für die Verbindung zur Diskette im Commodore-Format sorgt. Der VIA für den Busbetrieb beginnt bei der Adresse $1800 und der für den Diskcontroller ab Adresse $1c00.

Der VIA ist in einem 40-poligen Gehäuse untergebracht, wobei die Floppystationen 1570 und 1571 vorzugsweise mit den stromsparenden CMOSVersionen (65C22) ausgestattet wurden. Die Informationen aus den Datenblättern des VIA 6522 und aller im weiteren Verlauf besprochenen Interfacebausteine finden Sie in Tabellen zusammengestellt in Anhang B dieses Buches.

Wir werden später noch intensiv auf die Verwendung der Schnittstellenbausteine in der 1570/71 eingehen. Dazu benötigen wir jedoch noch' ein paar zusätzliche Informationen, die uns das anschließende Kapitel geben wird.

8.2.1.1 Die Schnittstelle zum seriellen Bus (VIA 1)

Der erste VIA 6522 hat die Grundadresse $1800. Wichtig für uns sind die beiden parallelen I/O-Ports dieses VIA, die den Kontakt mit der Außen

welt sicherstellen. Der VIA 1 ist für den Betrieb des seriellen Busses zuständig. Dabei steuert er im 1541-Modus den vollen Busbetrieb und ist im 1570/71-Modus nur mehr für das Timing des Busbetriebs zuständig. Ausschlaggebend für die Bedienung des seriellen Busses ist dabei Port B des VIA. Dieser liegt bei Adresse $1800. Über ihn wird der Busverkehr betrieben. Port A des Bausteins hatte in der 1541 keine Funktion und lag brach. In der 1570/71 jedoch übernimmt auch dieser Port neue Aufgaben.

Die nachfolgende Tabelle zeigt die Belegung der einzelnen Pins des Port B:

Bit/Pin	Bedeutung
0	Leitung für DATA IN (invertierender Eingang)
1	Leitung für DATA OUT (invertierender Ausgang)
2	Leitung für CLOCK IN (invertierender Eingang)
3	Leitung für CLOCK OUT (invertierender Ausgang)
4	ATN A; 1 heißt, daß jeder ATN des Computers von der Hardware der Floppy automatisch beantwortet wird.
5	Lo-Bit der Hardware-Gerätenummer (DIP-Schalter 1)
6	Hi-Bit der Hardware-Gerätenummer (DIP-Schalter 2)
7	Leitung für ATN IN (invertierend; auch über CA1/VIA 1)

Tabelle 8.2 Pinbelegung des Port B des VIA I für den seriellen Bus

Wie Sie sehen, sind den Leitungen für den seriellen Bus jeweils Inverter vorgeschaltet. Diese elektronischen Bauelemente sorgen dafür, daß ein logischer Eingangspegel immer genau in den entgegengesetzten logischen Ausgangspegel umgewandelt wird. Löschen Sie also beispielsweise das Bit 1 für den seriellen Bus, so geht die Leitung zum Computer mit der Bezeichnung DATA OUT auf den entgegengesetzten Pegel, also auf logisch I.

Wenn Sie sich das DOS-Listing ansehen, werden Sie feststellen, daß in der Dokumentation immer der Pegel angegeben ist, der von der angeschlossenen Leitung angenommen wird und nicht der Pegel, der im Interface-Baustein gesetzt wird. Das hat den Vorteil, daß Sie immer sofort feststellen können, ob ein Signal invertiert oder normal gesendet bzw. empfangen wird. Steht in der Dokumentation beispielsweise "DATA OUT auf Lo setzen" und die Routine setzt das entsprechende Bit, so können Sie sofort sagen, daß hier ein Inverter zwischengeschaltet sein muß, da die Leitung sonst auf Hi gehen müßte.

Achtung! Im Gegensatz zur Floppystation sind den Busbausteinen im Computer (C64 oder C128) keine Inverter vorgeschaltet. Hier liegen also die tatsächlichen Pegel auch als entsprechende Bitzustände an. Das muß bei der Programmierung der Busbausteine unbedingt beachtet werden!

8.2.1.2 Die Schnittstelle zur Diskette (VIA 2)

Der zweite Port des VIA 1 hat weniger mit dem seriellen Bus zu tun, als mit dem Diskettenbetrieb. Seine Belegung sei daher an dieser Stelle aufgezeigt:

Pin/Bit	Bedeutung
0	Bit=0 zeigt an, daß der Kopf auf Spur 0 steht
1	Bit=l serieller Bus auf Ausgang; Bit=0 serieller Bus auf Eingang
2	Diskettenseite für Tonkopf (1571) 0 oder 1
3	n.v.
4	n.v.
5	Systemtaktfrequenz; Bit=l heißt 2 MHz; sonst 1 MHz
6	n.v.
7	zeigt BYTE-READY an; auch über CA1/VIA 2

Tabelle 8.3 zeigt Port A von VIA 1

Der Port A von VIA 1 wird in der Regel nicht über die Grundadresse
$1801, sondern über das Portregister $180f ohne Handshake betrieben.

Nun kommen wir zur Betrachtung des zweiten VIA 6522 in der Floppy. Er verfügt über die Grundadresse $lc00, wobei wiederum beide I/O-Ports verwendet werden. Port B ist der Steuerport, das heißt die Schnittstelle zum Diskcontroller, und Port A ist die Verbindung zur Schreib-/Leseelektronik. Die Pinbelegungen im einzelnen sind:

Bit/Pin	Bedeutung
0	Ansteuerung für 1. Steppermotorspule
1	Ansteuerung für 2. Steppermotorspule
2	Steuerung für Drivemotor; Bit=l heißt Motor an
3	Steuerung für LED am Laufwerk; 1 heißt LED an
4	WRITE PROTECT; Zustand der Schreibschutz-Lichtschranke; Bit=1 heißt Diskette ungeschützt; auch CA2/VIA 1
5/6	Steuerung der Aufzeichnungsrate auf der Diskette: 00 250000 Bits/s auf Spur 31-35/66-70 01 266664 Bits/s auf Spur 25-30/60-65 10 285712 Bits/s auf Spur 18-24/53-59 11 307688 Bits/s auf Spur 01-17/36-52
7	SYNC-Signal beim Lesen von Diskette

Tabelle 8.4 zeigt die Pinbelegung von Port B des VIA 2 bei Adresse $lcOO

Pin/Bit	Bedeutung
0-7	Paralleler Datenbus zum Diskcontroller (GCR)

Tabelle 8.5 zeigt die Belegung von Port A des VIA 2 bei Adresse $lcOI

Bei der VIA 2 sind zusätzlich zu den Portregistern noch zwei Handshakeleitungen belegt worden: CA2 und CB2.

CA2 schaltet dabei das SOE-Signal (serial output enable) ein, wenn er auf Hi gesetzt wird.

CB2 schaltet die Schreib-/Lesefunktion des Tonkopfes. Wird CB2 auf Hi gesezt, so ist der Schreib- /Lesekopf auf Eingang geschaltet (Daten lesen), ansonsten befindet sich der Kopf im Schreibmodus.

Wie die jeweilige Beschaltung der Leitungen erfolgt, ist dem ROM-Listing und den Tabellen im Anhang B zu entnehmen.

Das BYTE-READY Signal, das bei Port A des VIA 1 anliegt, wurde erst bei der Entwicklung der 1570/71 dorthin verlegt. Es existiert auch noch die Leitung zum SO-Pin des Prozessors, die das Overflow-Bit in dessen Statusregister setzt, wenn ein Byte gelesen oder geschrieben wurde. Diese Leitung wurde von der 1541 übernommen und ist der Kompatibilität wegen erhalten geblieben. Es gibt also zwei Möglichkeiten das BYTE-READY-Signal abzufragen: über Port A (PA 7) von VIA 1 und über das Overflow-Flag des Prozessors.

8.2.2 Der CIA 6526 als schnelles Buscontroller-Interface

Ein weiterer Baustein in unserer Reihe ist der CIA 6526. Im Gegensatz zum VIA 6522 (versatile interface adaptor) wird der CIA (complex interface adaptor) in der 1570/71 nur zu einem ganz geringen Teil eingesetzt. Lediglich das eingebaute serielle Schieberegister wird verwendet, obwohl es dazu eigentlich keines weiteren Bausteins bedurft hätte. Auch der VIA 6522 enthält nämlich ein solches Schieberegister.

Der Rest des CIA liegt in der 1570/71 brach und kann unter Umständen für benutzereigene Zwecke verwendet werden (zum Beispiel zum Steuern von Hardwareerweiterungen).

Der CIA besitzt in der 1570/71 die Grundadresse $4000.

Eine der Besonderheiten des CIA 6526 ist seine eingebaute Echtzeituhr. die mit der Frequenz des Versorgungs-Stromnetzes getriggert wird. Da diese Frequenz sehr konstant ist, kann mit deren Hilfe eine ziemlich genaue Uhr mit Alarmzeit programmiert werden. Auch die Belegungen des CIA 6526 sind im Nachschlageteil des Buches im Anhang B beschrieben.

8.2.3 Der Diskcontroller WD 1770

Der WD 1770 von Western Digital ist ein kompletter Diskcontroller/Formatter und wird im weiteren Verlauf noch sehr ausführlich beschrieben. Er gestattet es der Floppy, Fremdformate zu lesen und zu schreiben und ist deshalb von besonderem Interesse für uns.

Der WD 1770 ist im Gegensatz zu dem "Pseudo-Controller" von Commodore ein echter Diskcontroller, der sämtliche Signale für den Diskbetrieb bereitstellt. Diese Signale werden jedoch im DOS der 1570/71 nur teilweise genutzt. So wird zum Beispiel das Positionieren des Tonkopfes in der 1570/71 immer noch softwaremäßig gesteuert, obwohl der WD 1770 diese Aufgabe vollkommen übernehmen könnte.

Im Speicherbereich der Floppy belegt der WD 1770 die Adressen $2000 bis $2003.

9 Das Arbeiten mit dem DOS

Wie Sie inzwischen wissen, belegt das Betriebssystem der Floppy, also das DOS, einen 32 KByte großen Speicherbereich von $8000 bis $ffff. Um eigene Programme in das DOS einzubinden und auszuführen wollen wir nun einmal sehen, was in diesem Bereich passiert und wie das DOS die Disketten verwaltet.

Den größten Teil dieses Buches nimmt das dokumentierte DOS-Listing ein. Dieses Listing enthält sämtliche Routinen des Floppy-Betriebssystems. Es ist unvermeidbar, daß Sie sich die einzelnen Routinen einmal ansehen um daraus zu lernen.

Das Studium des DOS ist die erste und wichtigste Voraussetzung für das erfolgreiche Arbeiten mit der 1570/71. Deshalb wurde das Betriebssystem so ausführlich wie nur möglich dokumentiert und ermöglicht damit auch dem weniger versierten Maschinensprache-Programmierer den Einstieg in die Programmierung der Floppystation.

Zu den Kapiteln, die sich mit der Arbeitsweise und der Programmierung des DOS beschäftigen, ist es leider unumgänglich, daß Sie die Maschinensprache-Programmierung beherrschen. Sind Sie ein reiner BASIC-Programmierer mit schon etwas größerer Programmiererfahrung, dann sollten Sie sich überlegen, ob Sie den Einstieg in die Maschinensprache des Prozessors 6502 nicht einmal versuchen. Sie erhalten damit neue, ungeahnte Möglichkeiten bei bestimmten Problemlösungen und kommen in den Genuß der sehr hohen Verarbeitungsgeschwindigkeit eines Mikroprozessors. Außerdem gibt es mittlerweile auf dem Markt ein so großes Angebot an wirklich guter Lektüre zu diesem Thema, die den Einstieg sehr leicht macht.

Dieses Buch wurde so aufgebaut, daß alles, was Sie zum Nachschlagen beim Programmieren benötigen, im Anhang untergebracht ist. Sie erhalten dort die Speicherbelegung der Floppy 1570/71, die Belegungen aller Interface-Bausteine (VIA 6522, CIA 6526 und WD 1770) und alle restlichen wichtigen Tabellen und Pläne. Dieses System erlaubt es Ihnen, sich das Buch einmal durchzulesen, um die einzelnen Prinzipien zu verstehen und anschließend nur noch den Nachschlageteil zu verwenden, ohne durch viel Herumblättern aus der Arbeit gerissen zu werden.

9.1 Die Funktionsweise des DOS 3.0

Das DOS 3.0 ist der direkte Nachfolger des DOS 2.6 der Floppy 1541. Dieses DOS 2.6 wiederum wurde aus den Floppystationen des Typs 4040 entnommen und auf ein Einzellaufwerk umgestellt. Beim Umschreiben des DOS für die 1541 wurden jedoch einige grundsätzliche Änderungen gegenüber der 4040 vorgenommen.

Die wichtigste Änderung ergibt sich wohl aus dem unterschiedlichen Aufbau der Hardware der 1541 zur 4040. Die 4040 besteht aus einem Computersystem mit zwei Mikroprozessoren; und zwar einem 6502 für die Verarbeitung der Anwenderbefehle und einem 6504, der die Laufwerksteuerung übernimmt, also als Diskcontroller arbeitet. Das ist bei der 1541 und auch bei der 1570/71 nicht mehr der Fall. Hier muß ein einziger 6502 die gesamte Arbeit übernehmen. Die Folge dieses "Abspeckens" ist klar: die 1541 wurde gegenüber der 4040 deutlich langsamer.

Was bei der 4040 ein zweiter Prozessor übernommen hat, wird bei der 1541 und auch bei deren Nachfolgern 1570 und 1571 durch die Interrupttechnik erledigt. Die zweite grundlegende Änderung zur 4040 ist der serielle Bus. Die Floppystation 4040 wurde über einen IEEE-488-Bus mit dem Computer verbunden, der parallel mit 8 Bit arbeitet. Bei der 1541 wurde auf einen seriellen Bus umgestiegen, der nur mehr 1 Bit "breit" ist. Das macht sich natürlich bei der Geschwindigkeit bemerkbar. So ist auch durch diese Änderung die Leistungsfähigkeit der 1541 deutlich begrenzt worden.

Bei der 1570/71 hat man dieses Manko zumindest teilweise durch einen sehr schnellen seriellen Bus wieder wettgemacht. Das Prinzip dieses Busses werden wir später noch besprechen. Jetzt interessiert uns das Prinzip der Befehlsverarbeitung in der Floppy.

9.1.1 Das Prinzip der Jobschleife

Wie oben erwähnt, besitzt die 4040 zwei Prozessoren, wobei der eine für die Befehls-Entgegennahme, Verarbeitung und grundsätzliche Steuerung der Floppystation und der andere, ein 6504, für den Diskbetrieb, also das Formatieren, Schreiben und Lesen verantwortlich war.

Ein solches System hat den Vorteil, daß mehrere Vorgänge gewissermaßen parallel verarbeitet werden können, was sich positiv auf die Geschwindigkeit auswirkt. Es hat aber den Nachteil, daß ein sehr ausgefeiltes Programm für die Synchronisation zwischen beiden Prozessoren und für den reibungslosen Ablauf aller Vorgänge sorgen muß: Stellen Sie sich vor, der Diskcontroller beginnt auf einmal, eine Diskette zu formatieren, ohne daß der Befehl dazu erteilt wurde.

Zu diesem Zweck der Synchronisation hat man die Zeropage ausgewählt, die ja bei beiden Prozessoren identisch ist. Es werden hier spezielle Speicherstellen definiert, die für die Kommunikation zwischen beiden Prozessoren verantwortlich sind, wobei das System dann folgendermaßen funktioniert:

Sie schalten die Floppy und den Computer ein, wobei natürlich passiert nichts außergewöhnliches. Der 6502, wir nennen ihn einmal Hauptprozessor , führt das RESET-Programm durch und wartet auf einen Befehl vom Computer. Der 6504, also der Coprozessor, arbeitet in einem Programmteil, der sich Jobschleife nennt. Auch er hat nichts zu tun, außer die schon angesprochenen Speicherstellen in der Zeropage dauernd zu kontrollieren.

Kommt jetzt ein Befehl, so wird dieser vom Hauptprozessor entgegengenommen und analysiert. Nehmen wir einmal an, es geht darum, einen Block im Direktzugriff von der Diskette zu laden.

Ist die Syntax des empfangenen Befehls in Ordnung, so passiert folgendes: Der Hauptprozessor speichert in den schon erwähnten Zeropage-Speicherstellen, die auch als Jobspeicher bezeichnet werden, die vom Benutzer gewünschte Spur- und Sektornummer des Blocks ab. Bis der Hauptprozessor einen Code (Jobcode) in einer bestimmten Speicherstelle des Jobspeichers ablegt geschieht weiterhin nichts. Danach geht der Hauptprozessor wieder seiner gewohnten Arbeit nach (Befehle empfangen und analysieren), wobei er jedoch die Speicherstelle, in die er den Jobcode gespeichert hat, dauernd überwacht.

Der Coprozessor findet nun in der Speicherstelle den Jobcode, der in unserem Fall "Block lesen" bedeutet. Nun wird das Laufwerk aktiviert, der Schreib-/Lesekopf positioniert, die LED am Laufwerk eingeschaltet, der entsprechende Sektor gesucht und in einen reservierten Pufferspeicher gelesen. Ist alles abgelaufen, so speichert der Coprozessor in den gleichen Jobspeicher, der auch den Jobcode enthielt, eine Rückmeldung und geht wieder in die Warteschleife zurück.

Der Hauptprozessor wiederum findet nun bei der Kontrolle des Jobspeichers die Rückmeldung und "weiß" nun, daß der Sektor in den Puffer geladen wurde. Der Block wird dann abschließend zum Computer geschickt.

Dies ist, wie Sie sehen, ein recht kompliziertes System. Der Name Jobschleife wird nun aber verständlich: der Coprozessor verharrt in einer Schleife, bis ein Job, also ein Befehlscode gefunden wird. Nach seiner Ausführung wird dann wieder in die Schleife zurückgekehrt.

Nun werden Sie sich fragen, ob man diese ganzen Vorgänge nicht auch einfacher steuern kann? Das könnte man, da der Hauptprozessor scheinbar während der Ausführung des Jobs vom Coprozessor nichts zu tun hat. Das stimmt jedoch nicht ganz. Dieses ganze System läuft nämlich auf einem Doppellaufwerk (4040), und es ist daher durchaus möglich, daß der Hauptprozessor schon wieder einen Befehl empfängt und analysiert, noch während der Coprozessor mit dem Lesen beschäftigt ist. Auf diese Weise kann die Floppy sehr schnell hintereinander Befehle empfangen und analysieren, ohne sich um die Steuerung des Laufwerks kümmern zu müssen.

9.1.2 Die Jobschleife der 1570/71

Die 1570/71 besitzt, wie schon erwähnt, keinen zweiten Prozessor, der den Hauptprozessor entlastet. Die Jobschleife funktioniert bei dieser Floppy dennoch im Prinzip ganz ähnlich. Es wird hier lediglich von der Interrupttechnik des 6502-Prozessors Gebrauch gemacht. Wir reden jetzt nicht mehr von einem Hauptprozessor und einem Co prozessor, sondern von einem Hauptprogramm und einem Diskcontrollerprogramm, beziehungsweise einer Jobschleife.

Der Ablauf sieht vollkommen analog aus. Das Hauptprogramm holt und analysiert einen Befehl. Dann wird der Jobcode inklusive aller notwendigen Parameter (Spurnummer, Sektornummer, Diskettenseite,...) in den Jobspeicher geschrieben und auf die Rückmeldung gewartet. Das DiskcontrollerProgramm (künftig nur noch als Jobschleife bezeichnet) besteht aus einem Interruptprogramm, das durch einen Timer regelmäßig aufgerufen wird (etwa alle 8 ms). Es findet den Jobcode und führt den Job aus. Danach schreibt es eine Rückmeldung in den Jobspeicher, die dem Hauptprogramm signalisiert, daß der Job ausgeführt worden ist.

Das Prinzip ist also das gleiche geblieben, es hat lediglich seinen Sinn und Zweck verloren. Während der Hauptprozessor in der 4040 weitergearbeitet hat, ist es dem Hauptprogramm in der 1570/71 unmöglich, während eines aufgetretenen Interrupts weiterzulaufen. Der Interrupt hat ja schließlich den Zweck, das Hauptprogramm zu unterbrechen, wie der Name schon sagt.

Sie werden im DOS 3.0 der 1570/71 einige Relikte finden, die den parallelen Betrieb von zwei Prozessoren unterstützen und in der 1570/71 ihren Sinn verloren haben. Diese Anpassung des Multiprozessor-Programms an ein Singleprozessorsystem hat neben dem Geschwindigkeitsverlust auch noch einige Fehler entstehen lassen, die schon in der 1541 vorhanden waren und auch bei der 1570/71 noch nicht ausgemerzt wurden.

Der Geschwindigkeitsverlust ist natürlich auf die, für ein SingleprozessorSystem aufwendige, Programmierung zurückzuführen. Sämtliche Vorteile, die dieser Programmierstil bei einem Zweiprozessorsystem gebracht hat, sind in Nachteile für die 1570/71 umgeschlagen. Schon aus diesem Grund ist es empfehlenswert, die Interna der Floppy kennenzulernen, um durch eigene Programmierung einige Mängel zu beheben.

9.1.3 Die Besonderheiten des DOS 3.0 gegenüber dem DOS 2.6

Es ist Ihnen vielleicht bekannt, daß die 1570/71 das DOS 2.6 der 1541 nahezu unverändert enthält. Bei der DOS-Version 3.0 handelt es sich lediglich um eine Erweiterung gegenüber der 1541, wobei auf größtmögliche Kompatibilität geachtet wurde.

Prinzipiell sind nur zwei Änderungen zu verzeichnen. Erstens der schnellere serielle Bus in Verbindung mit einem Cl28 und zweitens die Möglichkeit, Fremdformate im MFM-Standard zu verarbeiten. Bei der 1571 kommt natürlich auch noch die andere Mechanik mit dem zweiseitigen Diskettenbetrieb hinzu.

Das Original-DOS der 1541 steht ab Adresse $c100 bis $ffff im Speicher. Es wurden lediglich ein paar Sprünge zur Erweiterung eingefügt, die die neuen Möglichkeiten der 1570/71 unterstützen. Das neue Betriebssystem steht im Bereich von $8000 bis $bfff und enthält sämtliche Routinen für MFM-Verarbeitung, zweiseitigen Diskettenbetrieb und schnelle Busroutinen.

Eine weitere Besonderheit ist sicherlich die Taktgeschwindigkeit der 1570/71 gegenüber früheren Commodore-Floppies. Diese beiden Floppystationen arbeiten nämlich unter anderem mit 2 MHz Taktfrequenz im Gegensatz zur 1541 mit 1 MHz. Äußerlich wird das vor allem am sehr schnellen Blinken der LED bei Fehlern deutlich, wenn die Floppy am C128 angeschlossen ist. Am C64 oder C128 im 64er-Modus simuliert die Floppy eine 1541 und schaltet auf 1 MHz-Betrieb um. Hier blinkt die LED mit der gleichen Geschwindigkeit, wie bie der 1541.

Wir werden später noch neue Befehle für die 1570/71 kennenlernen, die die 1541 nicht enthält. Diese Befehle sind vor allem für den schnellen seriellen Bus zuständig; sie erlauben aber auch das Umschalten des Betriebsmodus der Floppy zwischen 1541- und 1570/71-Modus. Auch C64-Besitzer kommen dann in den Genuß der neuen schnellen Formatierroutine dieser Floppies.

9.2 Programme im Pufferspeicher der Floppy

Wie Sie Programme in den Pufferspeicher der Floppy schreiben und dort starten können, wurde schon in Kapitel 7 besprochen. Jetzt interessiert uns, welche Programme man in der Floppy ablegen kann und wie diese aussehen müssen.

Vorzugsweise nimmt man als Speicherbereich in der Floppy den Puffer 2 ($0500 bis $05ff), der als Benutzerpuffer definiert ist. Er kann durch die Benutzer-Befehle (USER, siehe 7.11) sehr bequem programmiert werden.

Bevor Sie nun jedoch "drauflos programmieren" sind hier noch ein paar Tips und Warnungen. Wenn Sie mit Maschinenprogrammen in der Floppy arbeiten, befinden Sie sich in der Regel hinter den Sicherheitsbarrieren des DOS (Schreibschutzabfrage, Fehlererkennung, usw.), die die Disketten weitestgehend vor Pannen schützen sollen - es sei denn, diese Sicherheit wird in die eigenen Maschinenprogramme eingebaut. Sie können sich daher bei Fehlern ganze Disketten unwiderruflich überschreiben oder löschen und sollten deshalb stets mit Übungsdisketten arbeiten. Die Floppy ist außerdem von der Programmierung her nicht so benutzerfreundlich konzipiert wie der Computer und es erfordert wesentlich mehr Mühe, in der Floppy ein Programm zum Laufen zu bringen. Stürzt die Floppy einmal ab, und das kann bei einem Programmierfehler sehr schnell passieren, so hilft hier nur der Resetknopf und das Programm im Pufferspeicher ist verloren.

Ein sehr wichtiger Punkt ist die Mechanik der Floppy. Im Gegensatz zum Computer werden in der Floppy mechanische Teile durch Motoren bewegt. Alle mechanischen Teile sind jedoch dem Verschleiß und einer gewissen Bruchgefahr unterworfen. Denken Sie also immer daran, daß Sie sorgfältig programmieren, denn die kleinste Panne ist schon ein verstellter Tonkopf. Überlasten Sie die Mechanik niemals durch "Wahnsinnsprogramme" und verwenden Sie bei Unkenntnis der technischen Details lieber die Systemroutinen zur Ansteuerung, die auf die Mechanik der Floppy abgestimmt sind.

Ein weiterer Punkt betrifft die Fehlerbehandlung. Wenn Sie als Benutzer eine Diskette in das Laufwerk schieben und sich das Inhaltsverzeichnis ansehen, dann erledigt die Floppy alles vollautomatisch, inklusive Initialisieren der Diskette. Arbeiten Sie nun auf der Ebene der Jobschleife, so sind alle automatischen Vorrichtungen natürlich inaktiv, wenn sie nicht ausdrücklich angesprungen werden. Sie müssen eine Diskette vor einem Zugriff also "von Hand" initialisieren, bevor Sie die Jobschleife aufrufen. Denken Sie bitte immer an diese wenigen Ratschläge. Sie ersparen sich damit unter Umständen zerstörte Disketten und Laufwerke und stundenlange Fehlersuche.

9.2.1 Die Nutzung der Jobschleife

Nach diesen Ratschlägen wollen wir mit der Programmierung beginnen. Das nächstliegende Problem bei einem Programm im Pufferspeicher der Floppy ist natürlich der Zugriff auf die Diskette. Werden keine illegalen Zugriffe verlangt und soll nur mit herkömmlichen Sektoren gearbeitet werden, so fällt Ihnen dazu der Abschnitt über die Jobschleife ein. Wie schon erwähnt, steuert diese Schleife den gesamten Diskzugriff und müßte demnach z.B. zum Lesen eines Sektors aufgerufen werden.

Zu diesem Zweck sollten Sie einmal die RAM-Belegung der Floppy betrachten. In der Zeropage fallen dann nämlich sofort die Speicherstellen $0000 bis $0011 auf. Es handelt sich hierbei um die schon erwähnten Jobspeicher, wobei für jeden Pufferspeicher der Floppy ein Jobspeicher existiert.

Wichtig ist an dieser Stelle natürlich, daß Sie für einen Job immer einen anderen Puffer anwählen müssen, als den, in dem Ihr Programm steht. Die Jobschleife arbeitet nämlich immer im angewählten Pufferbereich.

Generell ist die Bedienung der Jobschleife sehr einfach. Nachfolgend sind alle Befehle an die Jobschleife aufgeführt:

$80	Lesen eines Sektors
$88	Lesen eines Sektors auf der gleichen Spur
$90	Schreiben eines Sektors
$a0	Verify eines Sektors
$b0	Suchen eines Sektors/einer Spur
$c0	Zurückfahren des Kopfes auf Spur 0
$d0	Starten eines Programms im Floppypuffer
$e0	Einbinden eines Programms im Floppypuffer in die Jobschleife
$f0	Formatieren einer Diskette

Wichtig bei den Diskettenzugriffen ist, daß die Diskette vorher initialisiert worden ist. Wir wollen einmal sehen, wie man einen Sektor (zum Beispiel Sektor 18,0) mit Hilfe der Jobschleife in den Pufferspeicher der Floppy liest. Unser Programm soll in Puffer 2 stehen und der Sektor soll in Puffer 0 ($0300 bis $03ff) geladen werden.

Schreiben Sie folgendes Maschinenprogramm in den Pufferspeicher der Floppy und legen Sie es bei $0500 ab:

```
0500 lda #18          ; Spurnummer 18
0502 sta $06          ; Spurnummer für Job in Puffer B
0504 lda #0           ;    Sektornummer 0
0506 sta $07          ; Sektornu__er für Job in Puffer B
```

```
0508 lda #$80          ; Jobcode für "Sektor lesen"
050a sta $00           ; an Jobschleife für Puffer 8 übergeben
050c lda $00           ; Jobspeicher lesen und
050e bmi $050C         ; warten, bis Job zu Ende ausgeführt
0518 cmp #2            ; auf Fehler prüfen
0512 bcs error         ; verzweige, wenn Fehler (Rückmeldung >1)
0514 ...               ; weitere Behandlung; kein Fehler
error ...              ; zur Fehlerbehandlung
```

Vergessen Sie nicht, die Diskette vor dem Start zu initialisieren. Wie Sie sehen, ist der ganze Vorgang recht kurz und einfach zu programmieren. Hier die Erklärung der wichtigsten Schritte:

Zuerst werden die Spur- und Sektornummer für Puffer 0 gesetzt. Dann wird der Jobcode für Lesen übergeben. Wie Ihnen vielleicht aufgefallen ist, bestehen alle Jobcodes aus Werten größer als $7F (127), also Werten, bei denen Bit 7 gesetzt ist. Diese Einteilung erlaubt eine einfache Abfrage auf das Ende des Jobs, da die Rückmeldungen generell kleiner als 128 sind und somit die Bedingung bei $050e nicht mehr erfüllen. Wichtig bei der Ausführung von Jobs ist natürlich, daß kein SEI-Befehl verwendet wurde, da die Jobschleife über den IRQ-Befehl aufgerufen wird. Der NMI hat in der 1570/71 keine Funktion.

Die Fehlerbehandlung nach Ausführung eines Job ist ebenfalls denkbar einfach. Wird eine Rückmeldung größer als 1 übergeben, so ist ein Fehler aufgetreten. Die Fehlermeldungen der Jobschleife im einzelnen:

$01	kein Fehler; alles ok
$02	Blockheader nicht gefunden
$03	SYNC nicht gefunden
$04	Datenblock nicht gefunden
$05	Datenprüfsumme falsch
$06	Fehler beim Formatieren
$07	Fehler bei Verify
$08	Diskette schreibgeschützt
$09	Headerprüfsumme falsch
$0a	Datenblock zu lang
$0b	falsche ID im Blockheader/ Diskette nicht initialisiert
$0d	Indexloch nicht gefunden
$0e	Syntaxfehler bei MFM
$0f	keine Diskette im Laufwerk
$10	Fehler bei Dekodierung; illegale GCR-Bytes

Wie Sie sehen, existieren also alle Fehlermeldungen der Floppy, die den Diskettenbetrieb betreffen, bereits auf der Ebene der Jobschleife. Sie brauchen lediglich in den Klartext umgewandelt zu werden. Wenn Sie die Fehlertabelle im Anhang betrachten, können Sie erkennen, welche JobRückmeldung zu welchem Klartext gehört.

In der Jobschleife wird auch noch auf Schreibschutz geprüft, wie Sie an der Meldung $08 sehen können. Gehen Sie jedoch noch weiter hinab auf die Ebene unter der Jobschleife, in der Sie ein Programm schreiben können, das als Teil der Jobschleife betrachtet wird, so ist Vorsicht geboten.

Die übrigen Kommandos werden analog zum Kommando für "Block lesen" verwendet. Eine Ausnahme bilden nur die beiden Jobcodes $d0 und $e0, die jeweils ein Programm starten.

9.2.2 Programme in die Jobschleife einbauen

Wichtig ist vor allem der Jobcode $e0. Er erlaubt Programme zu schreiben, die als Interruptprogramme direkt in der Jobschleife ausgeführt werden. Die Programme, die wir mit dem M-E aufrufen, müssen durch den RTS-Befehl beendet werden, da die Floppy die Kontrolle dann wieder an das Hauptprogramm übergibt. Mit Jobschleifen-Programmen ist das jedoch anders. Diese müssen nur irgendwie zu einem definierten Abschluß der System-Interruptroutine führen und können so auch Teile der übrigen Jobschleife überspringen.

Verwenden Sie das Kommando $d0, so wird ein Maschinenprogramm am Beginn des gewählten Puffers gestartet und in die Jobschleife eingebaut. Es handelt sich hierbei also um ein echtes Interruptprogramm.

Verwenden Sie den Jobcode $e0, so geht das DOS davon aus, daß ein Programm im Puffer steht, das einen Diskettenzugriff durchführt. Bei $e0 müssen nämlich zusätzlich vorher die Spur- und Sektornummer im Jobspeicher abgelegt worden sein. Wird dieser Jobcode dann ausgeführt, so fährt die Jobschleife das Laufwerk hoch und positioniert den Schreib-/Lesekopf auf der angegebenen Spur. Danach wird das Maschinenprogramm am Anfang des entsprechenden Puffers gestartet. Übergeben Sie den Jobcode für Puffer 0, so aktiviert die Jobschleife das Laufwerk und springt anschließend zur Adresse $0300.

Wie Sie sehen, kann an dieser Adresse dann ein Programm stehen, das zum Beispiel in der Lage ist, fehlerhafte Sektoren einzulesen, um eine zerstörte Diskette zu retten oder einen Kopierschutz zu entwickeln.

Am Ende der eigenen Routine muß dann der Akku mit dem Wert der gewünschten Rückmeldung geladen und zurück in die Systemjobschleife gesprungen werden. Sie können also auch beliebige Rückmeldungen setzen und an Ihr Hauptprogramm übergeben.

Verwenden Sie den Jobcode $d0, um ein Programm in die Jobschleife einzubinden, so unterbleibt das Starten des Laufwerks. Hier ist es also zum Beispiel möglich, eine eigene Jobschleife mit anderen Jobcodes zu entwickeln und in die gegebene Jobschleife einzufügen. Der Kreativität sind mit diesem Befehlen keine Grenzen gesetzt.

Aus dem letzten Kapitel kennen Sie die Belegungen der VIAs 6522. Sie können also diese VIAs direkt bedienen und entweder auf den seriellen Bus oder die Diskette zugreifen, sofern Sie im Commodore-Format arbeiten.

9.2.3 Wichtige Systemroutinen für Programmierer

An dieser Stelle werden ein paar Routinen aus dem Betriebssystem herausgestellt. Sie werden entweder sehr oft aufgerufen und/oder ins RAM der Floppy kopiert, um ausgeführt zu werden, oder sind wichtig und zum Ansehen empfohlen.

$eaa0	System RESET -Routine
$f2b0	Jobschleife für 1541-Modus
$92ba	Jobschleife für 1570/71 Modus
$f969	Jobschleife beenden; Rückmeldung setzen (1541)
$99b5	Jobschleife beenden; Rückmeldung setzen (1570/71)
$fe00	Kopf auf Lesen umschalten
$8764	Drivemotor einschalten
$8770	Drivemotor ausschalten
$e60a	Rückmeldung der Jobschleife im Klartext ausgeben; gegebenenfalls Fehlerblinken aktivieren
$f556	auf SYNC-Signal warten (1541)
$9754	auf SYNC-Signal warten (1571)
$fac7	Jobroutine zum Formatieren einer 1541- Diskette
$f300...	Routinen zum Ansehen empfohlen

Wichtig sind vor allem die vier letzten Routinen oder Adressen, da es sich hier um die Paraderoutinen zum Schreiben oder Lesen einer Diskette im 1541-Format handelt. Diese Routinen sollten Sie sich auf jeden Fall ansehen. Sie lernen dadurch, wie man einen Diskettenzugriff programmiert. Sie werden ohnehin bei der Analyse anderer Programme im Pufferspeicher feststellen, daß diese häufig nur geringfügig abgeänderte Kopien aus dem DOS der Floppy verwenden.

10 Die Aufzeichnung auf eine Diskette

Nachdem Sie nun in der Lage sind, direkt in die Interna des DOS einzugreifen und auch Daten auf die Diskette zu schreiben beziehungsweise von der Diskette zu lesen, ist es nur logisch, daß Sie auch wissen wollen, wie das funktioniert.

Theoretisch ist es möglich, die meisten nötigen Informationen aus dem dokumentierten DOS-Listing zu entnehmen (Routinen zum Bearbeiten wurden bereits empfohlen). Sie werden jedoch sehr schnell auf Begriffe stoßen, die Sie noch nicht kennen und die dennoch zum Teil schon in manchen Kapiteln erwähnt wurden.

Nach diesem Kapitel haben Sie jedoch die größten Schwierigkeiten hinter sich und können sofort in die Diskettenprogrammierung einsteigen. Im Augenblick steht Ihnen jedoch noch ein wenig Theorie bevor.

10.1 Die Aufzeichnung von Daten auf Magnetträger

Wir wollen uns nun ansehen, wie zum Beispiel ein Byte auf einer Diskette aussieht. Was ist überhaupt eine Diskette und nach welchem Prinzip kann man darauf überhaupt Daten speichern?

Wie Sie sicherlich wissen, besteht eine Diskette aus einer dünnen Kunststoffscheibe mit einer Beschichtung aus magnetischem Material. Diese Schicht läßt sich ähnlich einem Eisenstück magnetisieren. Der Vorteil der Diskettenbeschichtung ist jedoch, daß diese Magnetisierung dauerhaft erhalten bleibt und deshalb zur Datenspeicherung verwendet werden kann.

Diese Magnetisierung erfolgt in unserem Fall durch einen Schreibkopf, der in seinem Inneren einen winzigen Elektromagneten enthält. Bild 10.1 zeigt das Schema eines solchen Schreib-/Lesekopfes.

Wenn nun Strom durch die Wicklung dieses Schreib-/Lesekopfes fließt, so wird der Eisenkern magnetisiert, und zwar in der Richtung abhängig von der Polung. Diese Magnetisierung überträgt sich auf die empfindliche Schicht der Diskette, die vor dem Tonkopf vorbeigedreht wird und speichert demzufolge Daten darauf.

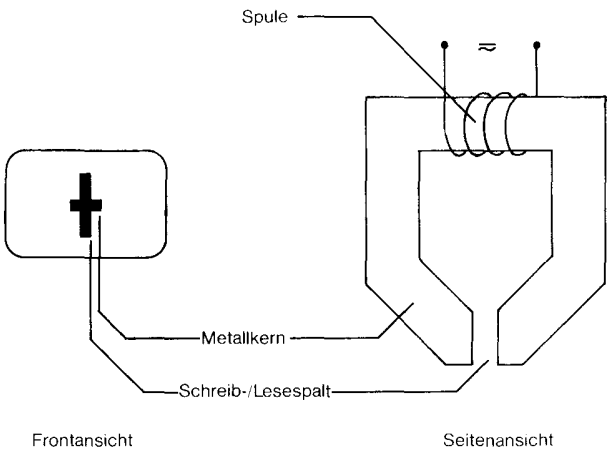


Bild 10.1: Darstellung eines Schreib-/Lesekopfes

Die Datenspeicherung geht nach einem sehr einfachen Prinzip vor sich: eine Elektronik zerlegt jedes Byte in seine 8 Bits. Die elektrischen Signale dieser Bits erscheinen folgendermaßen am Tonkopf: gleichbleibende Stromrichtung entspricht einem 0-Bit, wechselnde Stromrichtung entspricht einem 1-Bit. Die Ausgabe auf die Diskette ist also letztendlich ein unregelmäßiger Wechselstrom im Tonkopf. Jedesmal, wenn ein 1-Bit kommt, wird die Stromrichtung umgepolt und damit auch die resultierende Magnetisierung auf der Diskette. Bei einem 0-Bit bleibt der Strom im Tonkopf wie er ist. Bild 10.2 zeigt schematisch einen solchen Schreibvorgang:

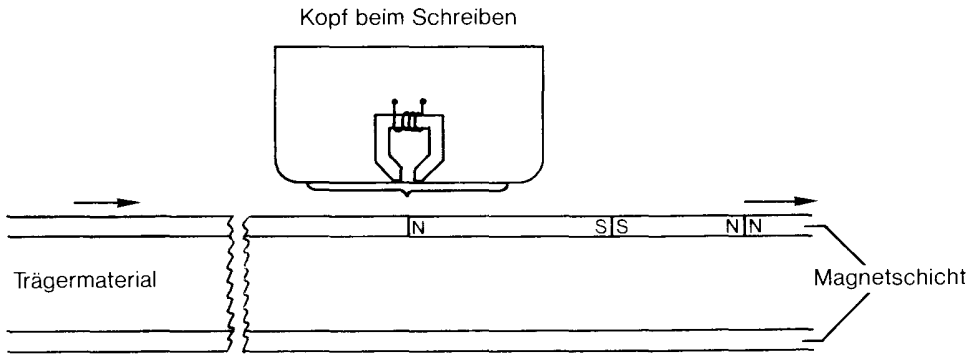


Bild 10.2: Tonkopf beim Schreiben auf eine Diskette

Sollen die Daten wieder gelesen werden, so kehrt man den ganzen Vorgang des Schreibens einfach um. Dreht sich nun die Diskette am Tonkopf vorbei, so wird entsprechend der Magnetisierungsrichtung auf der Diskette wieder ein Strom im Tonkopf induziert. Dieser Strom wird verstärkt und in ein digitales Ausgangssignal zurückverwandelt. Dabei entspricht jeder auftretende Magnetisierungswechsel einer Spannungsumpolung im Tonkopf und damit einem 1-Bit; geschieht am Tonkopf nichts, so handelt es sich um ein 0-Bit. Bild 10.3 zeigt noch einmal, wie das gemeint ist.

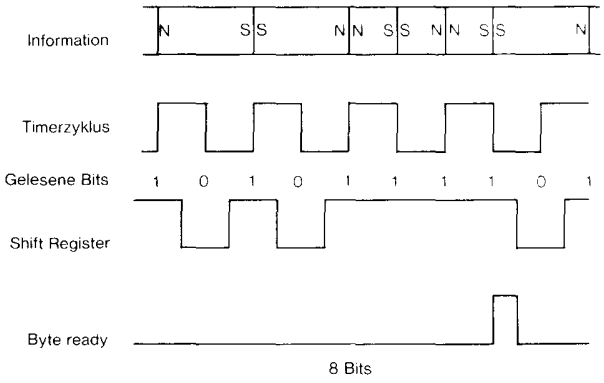


Bild 10.3: Das Lesen von einer Diskette

Diese Aufzeichnungstechnik erfordert natürlich ein gewisses Timing beim Lesen der Daten, da ein 0-Bit schließlich nichts anderes ist, als "garnichts". Bei gleichbleibender Magnetisierung wird nämlich kein Strom induziert. Die Floppy wartet also beim Lesen immer eine gewisse Zeit, bis das entsprechende Datenbit eingelesen sein muß. Danach wird festgestellt, ob während dieser Zeit eine Spannungsänderung aufgetreten ist oder nicht. Beim Schreiben passiert das gleiche. Hier wird einfach eine gewisse Zeit lang "nichts getan", wenn ein 0-Bit aufgezeichnet werden soll.

10.2 Das Timingproblem beim Lesen und Schreiben

Die Zustandsänderung in einer bestimmten Zeiteinheit entspricht also jeweils dem Wert des Bits, das gelesen oder geschrieben werden soll. Nun darf man aber auch nicht vergessen, daß selbst der beste Laufwerkmotor gewissen Drehzahlschwankungen unterliegt. Diese Drehzahlschwankungen können zu einer Menge Fehlinformationen führen.

Sie sehen jedoch in der Praxis, daß eine Floppystation äußerst zuverlässig arbeitet. Sie bringt sehr selten Schreib- oder Lese- Fehlermeldungen - und wenn, dann aus anderen Gründen.

Prinzipiell kann man sich natürlich eine Menge einfallen lassen, um solche Drehzahlschwankungen auszugleichen. In der Floppy 1570/71 sind zumindest die zwei am weitesten verbreiteten Methoden implementiert. Zwei Methoden deshalb, weil die 1570/71 sowohl das GCR-Format als auch das MFM-Format verarbeiten kann. Damit haben wir auch schon die Namen der beiden Systeme: MFM und GCR.

10.2.1 Das GCR-Format

Wenden wir uns zuerst dem schon von der 1541 her bekannten Commodore-Format mit der GCR-Aufzeichnungsmethode zu. GCR steht dabei für "group code recording" und bezeichnet ein V erfahren, das die Daten verschlüsselt, bevor sie auf die Diskette kommen, und das funktioniert folgendermaßen.

10.2.1.1 Das Ausgleichen von Laufwerksschwankungen unter GCR

Bei Laufwerksschwankungen kann man davon ausgehen, daß sie sich in einem Bereich ereignen, der zumindest mehrere Datenbits umfaßt, bevor das interne Timing nicht mehr mit der Position der Diskettenscheibe übereinstimmt.

Wie wir wissen, besitzt die Schreib- /Leseelektronik der Floppy einen Timer, der immer eine gewisse Anzahl von Systemzyklen verstreichen läßt, bevor ein Bit als eingelesen gilt. Danach wird festgestellt, ob in dieser Zeit ein Magnetisierungswechsel stattgefunden hat oder nicht und entsprechend ein Datenbit gesetzt oder nicht gesetzt.

Nehmen wir einmal an, die Diskette dreht sich kurzzeitig schneller. Dann kann es passieren, daß in einem Lesezyklus mehr als ein Bit von der Diskette am Tonkopf vorbeigedreht wurde und somit zum Beispiel zwei Magnetisierungswechsel stattgefunden haben.

Der Timingzyklus der Leseelektronik muß also der Geschwindigkeit der Diskette angepaßt werden. Dreht sich die Diskette schneller, so muß auch das Timing verkürzt werden, um kein Bit zu verschlucken oder mehrere auf einmal einzulesen. Dreht sich die Diskette wieder langsamer, so muß ein Lesezyklus länger dauern, um dem nächsten Bit die entsprechende "Herandrehzeit" zu lassen.

Commodore geht nun folgenden Weg: Es werden einfach immer alle ankommenden Magnetisierungswechsel dazu verwendet, den Timer neu zu triggern, das heißt, der Timer wird bei jedem 1-Bit neu gesetzt und hat somit immer einen Startwert, der sehr genau der Diskettendrehzahl entspricht; vorausgesetzt, es kommen oft genug hintereinander 1-Bits vom Tonkopf. Hätten wir nämlich zum Beispiel 20 $00-Bytes hintereinander auf der Diskette, so könnte der Timer für die Länge von 80 Lesezyklen nicht getriggert werden. Als praktisch notwendig hat sich erwiesen, den Timer spätestens alle 3 Lesezyklen, also alle 3 Bits neu zu triggern, um auch die kleinsten Laufwerksschwankungen sicher herauszufiltern. Die Leseelektronik der Floppy muß also spätestens nach zwei ankommenden 0-Bits wieder ein 1-Bit erhalten, um sicher funktionieren zu können.

Damit eine solche Bitfolge, also höchstens zwei 0-Bits hintereinander, gewährleistet ist, müssen die Daten codiert werden. Dazu kommen wir im übernächsten Abschnitt.

10.2.1.2 Markierungen auf der Diskette

Im Gegensatz zum MFM-Format verwendet Commodore keine hardwaremäßigen Markierungen (zum Beispiel das Indexloch der Diskette), um sich auf einer Spur zurechtzufinden. Will man jedoch einen Sektor einer Spur einlesen, so ist es unumgänglich, daß man sich auch auf der Spur zurechtfindet und seine Position bestimmen kann. Zusätzlich muß natürlich auch noch ein Sektor gefunden werden. Zuerst interessiert uns aber, wie sich die Floppy auf einer Spur orientiert.

Sie haben vielleicht schon etwas von SYNC-Signalen gehört oder gelesen. Diese Signale erlauben es der Floppy, die Position des Schreib- /Lesekopfes auf einer Spur festzustellen.

Wie Sie aus dem vorigen Abschnitt wissen, dürfen auf einer Diskette keine Datenbytes stehen, die mehr als zwei 0-Bits hintereinander enthalten. Es gibt jedoch noch eine weitere Einschränkung: Verboten sind ebenfalls Datenbytes, die mehr als acht 1-Bits hintereinander enthalten, also zum Be ispiel 2 Datenbytes mit jeweils dem Wert $ff hintereinander auf der Diskette.

Dieses Verbot hat seinen Grund in der Notwendigkeit von Markierungen auf jeder Spur einer Diskette. Mehr als acht 1-Bits hintereinander lösen bei der Leseelektronik nämlich ein Signal aus, das als SYNC-Signal bezeichnet wird. Dieses Signal entspricht keinem Byte- oder Bitwert, sondern wird auf einer getrennten Leitung über Bit 7 der Adresse $lc00 (VIA 2/Port B) an das Computersystem geschickt. Die Datenbytes hingegen kommen bei Port A des VIA 2 ($1c01) an.

Eine Markierung auf der Diskette mit einfachen Datenbytes ist natürlich deshalb nicht möglich, weil es keine Bitkombination (außer eben einer speziell behandelten) gibt, die nicht auch in einem Datenfeld zu finden sein könnte und dann die Leseelektronik durcheinander brächte.

Wie Sie aus den letzten beiden Abschnitten ersehen können, gibt es also im Commodore-Format zwei Gründe die Daten zu codieren, die auf die Diskette geschrieben werden sollen:
1. damit niemals mehr als zwei O-Bits folgen, um Timingprobleme auszuschließen und
2. damit niemals mehr als acht I-Bits folgen, um die Verwechslung von Daten und SYNC-Markierungen auszuschließen.

10.2.1.3 Die GCR-Codierung

Bei der nun folgenden Methode handelt es sich um die gesuchte Codierung von Datenbytes. Wir müssen bei der Floppystation also künftig zwischen zwei verschiedenen Schreibarten unterschieden:
- das Schreiben von Datenbytes
- das Schreiben von SYNC-Markierungen

Da SYNC-Signale lediglich durch mehr als acht direkt hintereinander stehende I-Bits ausgelöst werden, können Sie sehr einfach auf die Diskette gebracht werden, nämlich durch direktes Schreiben. Bei der 1570/71 verwendet man fünf $ff-Bytes. Eine SYNC- Markierung besteht also aus 40 1-Bits, die direkt auf die Diskette geschrieben werden.

Beim Schreiben von Datenbytes werden diese erst einmal in die entsprechenden GCR-Werte umgewandelt und danach auf die Diskette geschrieben. Die Umwandlung sämtlicher Datenbytes geschieht dabei folgendermaßen:

Wie Sie wissen, besteht ein Byte aus 8 Bit, wobei man diese 8 Bits wiederum in 2 mal 4 Bits aufteilt; ein sogenanntes Hi-Nibble (Bits 4-7) und ein Lo-Nibble (Bits 0-3). Dazu ein Beispiel:

Wert	binär	Hi-Nibble	Lo- Nibble
$45	01000101	0100----	----0101

Diese Zerlegung eines Bytes in sein Lo- und sein Hi-Nibble ist die Voraussetzung für die Umwandlung in das GCR-Format. Bei diesem Format wird jedes mögliche 4-Bit-Nibble eines Bytes in ein entsprechendes 5-BitGCR-Nibble umgewandelt. Die folgende Tabelle zeigt, welche Binärnibbles in welche GCR-Nibbles umgewandelt werden:

Hexadezimal	Dezimal	Binär	GCR - Äquivalent
$0	(0)	0000	01010
$1	(1)	0001	01011
$2	(2)	0010	10010
$3	(3)	0011	10011
$4	(4)	0100	01110
$5	(5)	0101	01111
$6	(6)	0110	10110
$7	(7)	0111	10111
$8	(8)	1000	01001
$9	(9)	1001	11001
$a	(10)	1010	11010
$b	(11)	1011	11011
$c	(12)	1100	01101
$d	(13)	1101	11101
$e	(14)	1110	11110
$f	(15)	1111	10101

Tabelle 10.1: Umwandlungstabelle Binär in GCR Diese Umwandlung sei an einem Beispiel dargestellt:

Nehmen wir einmal an, wir hätten das Byte $45 und das Byte $e2. Diese beiden Werte werden in ihre Binärwerte umgewandelt und sehen dann so aus:

$45 = %01000101 $e2 = %11100010

Wir zerlegen diese Werte nun in ihre Nibbles und wandeln diese in die entsprechenden GCR-Äquivalente um:

Binär:	0100	0101	1110	0010
GCR:	01110	01111	11110	10010

Da die Schnittstelle zum Diskcontroller immer 8 Bit, also ein komplettes Byte zum Schreiben auf die Diskette erwartet, werden die GCR-Nibbles der Reihe nach wieder zu Bytes zusammengefaßt:

```
01110 01111     11110 10010     werden zu
 0111 0011   1111 1101  0010    werden zu
   $73          $fd       $2-
```

Wie Sie erkennen können, sind aus unseren ehemals zwei Bytes nun zweieinhalb GCR-Bytes geworden. Eine solche Kombination ist natürlich ungeeignet, da ein halbes Byte nicht auf die Diskette geschrieben werden kann. In der Praxis verwendet die Floppy 1570/71 deshalb Routinen, die immer 4 Binärbytes gleichzeitig in 5 GCR-Bytes umwandeln, damit vollständige Bytes erhalten werden. An einem weiteren Beispiel sei das verdeutlicht:

```
Hex:              30       12            29              5a
Bin:     0011 0000     0001 0010     0010 1001     0101 1010
GCR:   10011 01010  01011 10010  10010 11001  01111 11010
ergibt 1001 1010 1001 0111 0010 1001     0110 0101  1111 1010
=      9a           97              29     65           fa
```

Die letztendlich entstehenden 5 GCR-Bytes können nun direkt zum Diskcontroller geschickt und auf die Diskette geschrieben werden.

10.2.2 Das MFM-Format

Das MFM-Format ist eine der am meisten verbreiteten Methoden zur Datenaufzeichnung. Fast alle Personal Computer verwenden heute dieses Format, und die 1570/71 ist dank ihres eingebauten WD 1770 in der Lage, alle diese Formate einwandfrei zu lesen und sogar zu schreiben.

Natürlich hat auch das MFM-Format die Probleme mit den Laufwerkschwankungen der Floppystation. Aber man geht hier einen anderen Weg, als beim GCR-Format, um Lesefehler auszuschließen.

10.2.2.1 Das Ausgleichen von Laufwerksschwankungen unter MFM

Das Prinzip des Einlesens von Daten entspricht genau dem des GCR-Formats. Auch bei MFM werden die 1-Bits durch Magnetisierungsänderungen auf der Diskette angezeigt, und auch hier dienen diese Magnetisierungsänderungen zum Triggern des Timers für den Lesevorgang.

Anders als beim GCR-Format codiert man die Datenbytes nun aber hardwaremäßig auf eine Art, die je nach Byte unterschiedlich ist. Es werden dabei fehlende 1- Bits eines Bytes durch entsprechende Taktbits ersetzt.

Der Diskcontroller kennt also zwei verschiedene Bits auf der Diskette; nämlich die Taktbits und die Datenbits, wobei die Taktbits immer am Anfang eines Timingzyklus auftreten und den Timer triggern, während die Datenbits im Timingzyklus weiter nach hinten verschoben sind.

Obwohl das MFM-Format das Indexloch beim Beschreiben von Disketten berücksichtigt, benötigt es natürlich auch Markierungen, die für das Suchen von Sektoren von entscheidender Bedeutung sind. Diese Markierungen, werden dadurch gekennzeichnet, daß sie aus einem Bytewert bestehen, bei dem zum Beispiel ein ganz bestimmtes Taktbit fehlt, obwohl es von der Bytestruktur her an die jeweilige Stelle gehört hätte. Durch diese Methode kann man sich das softwaremäßige Codieren der Datenbytes sparen, bei dem sehr viel Zeit verbraucht wird. Andererseits erlaubt das MFM-Format nicht so hohe Schreibdichten wie das GCR-Format. Das verhält sich bei der 1570/71 jedoch anders, denn hier paßt je nach Format mehr auf eine MFM-Diskette, als auf eine Diskette im 1541-GCR-Format.

10.2.2.2 Die Codierung unter MFM

Wir wollen uns nun ansehen, wie ein MFM-Controller zwischen Datenbytes und Markierungsbytes unterscheidet. Betrachten Sie dazu bitte Tabelle 10.2:

HEX - Wert	Verwendung oder Codierung unter MFM
$00-$f4	Nur Datenbytesj werden normal unter MFM geschrieben
$f5	Datenbyte: normales Schreiben unter MFM Markierung: ergibt $a1; kein Taktbit bei Bit 4 und 5; außerdem Vorbesetzung für CRC-Byte
$f6	Datenbyte: normales Schreiben unter MFM-Markierung: ergibt $cl; kein Taktbit bei Bit 3 und 4
$f7	normales Schreiben unter MFM; wird im MFM-Sektor auch als Platzhalter für CRC-Byte verwendet (siehe 10.3.2)
$f8-$ff	Nur Datenbytes; werden normal unter MFM geschrieben

Tabelle 10.2: Codierung von Bytes unter MFM

Wie Sie aus der Tabelle ersehen, werden nicht einfach sämtliche Bytes codiert, wie unter GCR, sondern es wird unterschieden, an welcher Stelle die Bytes gelesen oder geschrieben werden, beziehungsweise, welche Funktion das entsprechende Byte haben muß.

Die räumliche Unterscheidung ist unter MFM möglich, da der Controller anhand der Indexlochposition "weiß", ob er sich gerade an einer Markie

rung oder innerhalb eines Datenblocks befindet. Bei einem Zugriff auf eine Spur wird immer auf das Indexloch gewartet und dann bis zum gewünschten Sektor vorgetastet. Aus diesem Grund ist es unmöglich, unter MFM auf das Indexloch zu verzichten, da der Controller sonst keinen Anhaltspunkt hat, ob er gerade ein Datenbit oder ein Taktbit einliest. Diese kann er nur durch das Timing beim Lesen unterscheiden, da die Taktbits immer einen kurzen Augenblick vor einem Datenbit kommen.

10.3 Aufbau von Spuren und Sektoren einer Diskette

Wie Bits prinzipiell auf eine Diskette geschrieben werden, wissen Sie jetzt. Sie wissen nun auch, wie Bytes auf der Diskette aussehen müssen, damit keine Pannen durch Laufwerkschwankungen passieren. Jetzt werden Sie erfahren, wie die nächsthöheren Strukturen auf der Diskette aussehen, die Sektoren.

Wie schon erwähnt, besitzt die Floppy zwei Arten von Bytes, die Datenbytes und die Markierungsbytes. Das gilt sowohl für das MFM-Format, bei dem die Markierungsbytes durch fehlende Taktbits gekennzeichnet sind, als auch für das GCR-Format, bei dem es nur eine einzige Art von Markierungen, nämlich die SYNC-Markierungen, gibt.

10.3.1 Der Aufbau eines Sektors im GCR-Format

Zuerst betrachten wir den Aufbau eines Sektors im GCR-Format. Wie Sie wissen, enthält ein solcher Sektor immer 256 Datenbytes. Die Anzahl der Sektoren einer Spur schwankt von 21 (ganz außen) bis 17 (ganz innen).

Ein einzelner Sektor hat dabei folgenden Gesamtaufbau:

SYNC	$08+Blockheader	SYNC	$07+Datenbytes+Prüfsumme

→ Leserichtung

Bild 10.4: Darstellung eines Sektors im GCR-Format

Wie Sie sehen, beginnt der Sektor mit einer SYNC-Markierung. Wenn die 1570/71 also anfängt, einen Sektor zu suchen, so wird zuerst nach einer SYNC-Markierung Ausschau gehalten. Wird diese Markierung gefunden und überlesen, so stellt sich natürlich die Frage, ob es sich auch um den richtigen, den gesuchten Sektor handelt. Zu diesem Zweck geht dem eigentlichen Datenblock noch ein sogenannter Sektorkopf oder Blockheader voraus. In diesem Reader befinden sich alle Informationen über den folgenden Datenblock. Bild 10.5 zeigt einen solchen Blockheader:

SYNC	$08	Prüfsumme	Sektor	Spur	ID2	ID1	$0f	$0f	Lücke 1

→ Leserichtung

Bild 10.5: Aufbau eines Blockheaders

Da jedoch auch dem Datenblock eine SYNC-Markierung vorausgeht, wie in Bild 10.4 gezeigt, und der Kopf beim Zugriff auf eine neue Spur an einer ganz beliebigen Stelle stehen kann, muß er wissen, ob es sich bei der gefundenen SYNC-Markierung um die eines Datenblocks oder eines Blockheaders handelt.

Zu diesem Zweck steht direkt hinter der SYNC-Markierung ein Kennzeichen. Dieses Kennzeichen beträgt $08 bei einem Blockheader und $07 bei einem Datenblock.

Achtung: Alle dargestellten Bytes liegen natürlich im GCR-Format vor und müssen deshalb zuerst decodiert werden, wenn man sie von der Diskette liest.

Findet der Diskcontroller (in unserem Fall die Jobschleife) also ein $07 nach der SYNC-Markierung, so wird sofort auf die nächste Markierung gewartet, da es sich bei der gefundenen, um die eines Datenblocks handelt.

Nehmen wir jedoch einmal an, es handelt sich um das Kennzeichen $08, so werden anschließend die folgenden fünf Bytes von der Diskette gelesen und decodiert. Danach erfolgt die Kontrolle auf die richtige Prüfsumme, um Lesefehler auszuschließen. Stimmt die Prüfsumme nicht, so erfolgt die Ausgabe eines "22, READ ERROR".

Stimmt die Summe jedoch mit der eingelesenen überein, so wird auf die richtige ID geprüft. Tritt hier ein Fehler auf, so ist ein "29, DISK ID MISMATCH" die Folge.

Sind diese ganzen Hürden erfolgreich überwunden, so wird jetzt endlich nachgesehen, ob auch der richtige Blockheader eingelesen wurde. Das wird anhand der Spur- und Sektornummer aus dem Header geprüft. Stimmt diese Angabe nicht, so wird der nächste Blockheader gesucht, bis der richtige gefunden wird.

Nach dem Blockheader stehen jeweils noch zwei $0f-Bytes und eine folgende, 8 Byte große Lücke auf der Diskette. Diese 10 Bytes enthalten keinen Inhalt, sondern dienen lediglich dem Zweck, dem Diskcontroller bei Schreibvorgängen Zeit zum Umschalten zu lassen. Beim Schreiben eines Sektors wird nämlich nur nach dem betreffenden Blockheader gesucht und

dann auf "Schreiben" umgeschaltet. Die nachfolgende SYNC-Markierung mit den Datenbytes wird vollständig neu geschrieben. Die 8 Lückenbytes liegen natürlich auch codiert vor, das heißt, auf der Diskette ist diese Lücke in Wirklichkeit 10 GCR-Bytes groß und besteht aus einer Folge von 10 $55-Bytes, die das Äquivalent zum binären $00-Byte bilden.

Nun aber zum Datenblock. Sein Aufbau ist noch einmal in Bild 10.6 genauer dargestellt:

Bild 10.6: Aufbau eines Datenblocks

Nach dem Kennzeichen von $07 folgen also jetzt die eigentlichen Datenbytes. Diese 256 Datenbytes werden bei der GCR-Codierung zu 320 GCRBytes. Mit der Prüfsumme und dem Datenblockkennzeichen werden insgesamt 326 Bytes auf die Diskette hinter die SYNC-Markierung geschrieben. Der Blockheader besteht übrigens inklusive Lücke aus 20 GCR-Bytes.

Nach dem Datenblock und der Prüfsumme, die bei Nichtübereinstimmung einen "23, READ ERROR" auslöst, kommt noch einmal eine Lücke. Diese Lücke hat je nach Spur eine variable Größe und dient als Sicherheitsabstand zur nachfolgenden SYNC-Markierung des nächsten Sektors. Außerdem sorgt diese Lücke für eine gleichmäßige Verteilung aller Sektoren auf einer Spur, die dem Diskcontroller Zeit gibt, nach einem Schreibvorgang vor Beginn des nächsten Sektorheaders wieder auf Lesen umzuschalten.

Sie sehen also, wie sämtliche Sektoren einer Diskette im GCR-Format aussehen. Diese Form der Formatierung erlaubt es Commodore zum Beispiel, auf das Indexloch als Markierung zu verzichten, da die SYNC-Markierungen auf der Diskette sehr zuverlässig arbeiten.

10.3.2 Der Aufbau eines Sektors im MFM-Format

Das MFM-Format geht hier einen anderen Weg. Es ist auf das Vorhandensein der Indexlochkennung angewiesen und hat deshalb den folgenden Aufbau eines Sektors und einer Spur. "Einer Spur" deshalb, weil im MFMFormat, im Gegensatz zum GCR-Format ein Sektor keine unabhängige Einheit darstellt.

Im GCR-Format ist es theoretisch möglich, eine gesamte Spur zu löschen und darauf nur einen einzigen Sektor unterzubringen. Dieser Sektor kann

anhand seiner SYNC-Markierung sicher eingelesen und beschrieben werden.

Im MFM-Format sieht die ganze Sache anders aus. Hier ist eine Spur nach einem ganz bestimmten Prinzip aufgebaut, das das Vorhandensein des Indexlochs der Diskette und eines Sektors als Einheit erfordert. Es kann hier also kein einzelner Sektor nach Belieben irgendwo untergebracht werden, um beispielsweise einen Kopierschutz zu konstruieren.

Das funktioniert nicht, da das MFM-Format von einem WD 1770 Diskcontroller verwaltet wird. Dieser Diskcontroller ist im Gegensatz zum Commodore "Diskcontroller" ein vollwertiger Baustein, der fast alle Lesevorgänge hardwaremäßig und automatisch abwickelt, ohne daß dabei viel Software notwendig ist. Das Lesen eines Sektors ist hier also eine controllerinterne Angelegenheit. Wenn nur eine winzige Abweichung in wichtigen Parametern vom Standardformat festgestellt wird, so bricht der WD 1770 den Vorgang automatisch mit einer Fehlermeldung ab. Das Suchen eines Sektors und die Kontrolle auf Lesefehler erfolgen beim WD 1770 nämlich vollautomatisch, ohne zusätzlichen Programmaufwand und ohne Eingriffsmöglichkeit des Benutzers.

Nun aber zurück zum Thema. Der wichtigste Unterschied eines solchen MFM-Sektors zum Commodore-Format ist die Tatsache, daß unter MFM eine Sektorgröße gewählt werden kann. Der Benutzer kann beim Formatieren einer Diskette zwischen Sektoren mit 128, 256, 512 und 1024 Byte auswählen. Die meisten MFM-Formate (IBM PC, Kaypro, Commodore PC 10, Vortex, usw.) arbeiten mit 512-Byte Sektoren. Sie lehnen sich dabei an die Spezifikationen des CP/M 2.2 und des MS- DOS an.

Uns steht es frei, welches Format wir wählen. Der Gesamtaufbau aller Sektoren ist identisch und unterscheidet sich nur in der Größenbezeichnung des Sektors und seiner daraus resultierenden Anzahl von Datenbytes.

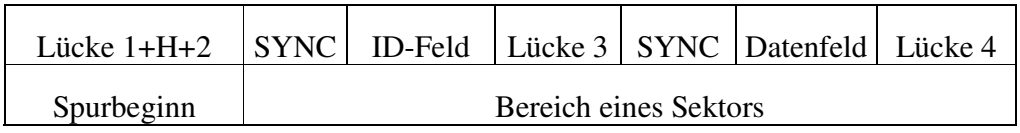


Bild 10.7: Aufbau eines Sektors unter MFM mit Index-Adreßmarke

Wie Sie aus Bild 10.7 ersehen können, existieren auch im MFM-Format sogenannte SYNC-Markierungen. Sie entsprechen jedoch in keinster Weise denen des GCR-Formats und haben auch eine etwas andere Wirkung auf

den Controller. In der Funktion sind jedoch beide Typen gleich. Sie sollen den Diskcontroller synchronisieren und ihn für den Anfang von Bytewerten auf der Diskette bereitmachen.

Wenn wir nun im folgenden den Aufbau eines Sektors unter die Lupe nehmen, so sollten Sie sich noch einmal die Tabelle 10.2 ins Gedächtnis zurückrufen. Alle Bytes in den nachfolgenden Bildern, die den Markierungsbytes entsprechen, sind mit einen T gekennzeichnet. Treffen Sie also auf ein $f5T, so wissen Sie, daß an dieser Stelle der Wert $a1 mit dem fehlenden Taktbit zwischen Bit 4 und 5 auf die Diskette geschrieben wird.

Zuerst interessiert der Bereich, der mit den eigentlichen Sektoren wenig zu tun hat. Er ist in Bild 10.7 als "Lücke 1+H+2" bezeichnet und trägt den Namen "index address mark". Es handelt sich bei der Index-Adreßmarke um eine Art "Spurheader", wobei beim Formatieren einer Spur einmal auf das Indexloch gewartet wird. Nach dem Indexloch wird dann sofort diese Index-Adreßmarke geschrieben, deren Aufbau in Bild 10.8 zu sehen ist:

80*$4e	12*$00	3*$f6T	$fc	50*$4e	60*$4e	...
Lücke 1	SYNC	Index-Adreßmarks			Lücke 2	...

→ Leserichtung

Bild 10.8: Aufbau der Index-Adreßmarke

Wie Sie sehen, steht nach dem Indexloch erst einmal eine Lücke als Verzögerungszone, die dem Controller Zeit läßt, seine Parameter für das Lesen oder Schreiben bereitzustellen. Danach kommt eine SYNC-Markierung, die aus zwölf $00-Bytes besteht. Es folgt der Code 3 mal $f6, der dem Schreiben von drei $c2-Bytes mit fehlendem Taktbit zwischen Bit 4 und Bit 5 des Codes entspricht. Danach kommt eine ID. Diese ID hat für die Index-Adreßmarke den Wert $fc. Es gibt noch zwei andere Typen von ID-Bytes, die dann jeweils den Sektorkopf und den Datenblock ankündigen. Nach der ID folgt eine Lücke, die aus 110 $4e-Bytes besteht. Diese Lücke ist in der Darstellung in zwei Blöcke aufgetrennt, da die 1570/71 in der Lage ist, eine Spur auch mit fehlender Index-Adreßmarke zu formatieren. Bei fehlender Index-Adreßmarke wird nach dem Erkennen des Indexlochs direkt mit den 60 $4e-Bytes begonnen. Der Rest einer Spur ist der gleiche.

Wir wollen jetzt den Bereich eines Sektors betrachten, der dem Blockheader unter GCR entspricht. Sehen Sie sich dazu einmal Bild 10.9 an.

12*$00	3*$f5T	$fe	Spur	Seite	Sektor	Größe	CRC	22*$4e
SYNC	ID-Feld des Sektors							Lücke 3

→ Leserichtung

Bild 10.9: Aufbau des MFM-ID-Feldes eines Sektors

Wie Sie sehen, besteht eine SYNC-Markierung unter MFM generell aus 12 Bytes mit dem Wert $00 wie auch schon bei der Index-Adreßmarke. Diese Zone läßt den Controller quasi zur Ruhe kommen und signalisiert ihm, daß die Bits, die er als erstes nach der SYNC-Markierung wieder einliest, Taktbits sind.

Nach der SYNC-Markierung kommt eine Markierung, die aus drei $a1-Taktbytes besteht. Nach dieser Markierung, die dem Controller den Beginn des ID-Feldes mitteilt, folgt die ID des Blockheaders. Diese ID besteht aus einem $fe-Byte und identifiziert das ID-Feld des Sektors.

Anschließend folgen Spurnummer, Seitenangabe der Diskette, Sektornummer und Größe des Sektors. Die Spur- und Sektornummer bedürfen dabei keiner weiteren Erläuterung. Die Seitenangabe enthält einen Wert zwischen 0 und 1 und gibt die Diskettenseite an, auf der geschrieben oder gelesen wird. Die Sektorgrößenangabe enthält einen Wert zwischen 0 und 3, wobei

0 für einen 128- Byte-Sektor,
1 für einen 256- Byte-Sektor,
2 für einen 512- Byte-Sektor,
3 für einen 1024- Byte-Sektor steht.

Der Sektorgröße folgt ein sogenanntes CRC-Byte. Dieses Byte ist eine Art Prüfsumme über das ID-Feld und entspricht in seiner Funktion der Prüfsumme des Blockheaders im GCR-Format. Stimmt dieses Byte beim Lesen nicht mit dem errechneten Wert überein, so gibt der Diskcontroller den Fehlercode für DATA CRC ERROR aus. Beim Schreiben wird vom Computer für das CRC-Byte übrigens der Wert $f7 eingesetzt. Dieses Byte teilt dem Diskcontroller mit, daß es sich hierbei um einen Platzhalter für das CRC-Byte handelt, das vom Controller "eigenhändig" eingesetzt wird.

Nach dem CRC-Byte folgt eine Lücke aus $4e-Bytes, die das ID-Feld vom nachfolgenden Datenfeld trennt.

Das Datenfeld, das nun die eigentlichen Datenbytes enthält entspricht dem Datenblock im GCR-Format und wird ebenfalls durch eine SYNC-Markierung aus 12 $OO-Bytes eingeleitet. Bild 10.10 zeigt den genauen Aufbau:

12*$00	3*$f5T	$fb	Datenbytes	CRC	x*$4e	...
SYNC	Datenfeld des Sektors				Lücke 4	...

→ Leserichtung

Bild 10.10: Au/bau des Daten/eides im MFM-Format

Nach der SYNC-Markierung folgen wiederum die drei Markierungs-Taktbytes, die die folgende ID ankündigen. Diese ID besteht aus dem Wert $fb, der dem Controller mitteilt, daß er sich vor einem Datenblock beziehungsweise einem Datenfeld befindet.

Das Datenfeld selbst besteht nun aus der gegebenen Anzahl von Bytes, also 128, 256, 512 oder 1024 Datenbytes. Diese Datenbytes werden wiederum durch ein CRC-Byte abgeschlossen, wie das schon im ID-Feld der Fall war. Das CRC-Byte wird dabei vom Prozessor wieder mit $f7 angegeben und vom Controller automatisch durch den errechneten Wert ersetzt.

Die Lücke 4 am Ende eines Sektors ist von den übrigen Parametern der Formatierung abhängig. Ihre Größe ist deshalb unterschiedlich, wie der Parameter x andeuten soll. Sie hängt sowohl von der Anzahl der Sektoren als auch von deren Größe ab.

Nach der Lücke 4 beginnt, falls die Spur noch nicht vollständig ist, ein neuer Sektor, der durch die SYNC-Markierung seines ID-Feldes angekündigt wird.

Ist hingegen schon der letzte Sektor der Spur erreicht, also der Sektor mit der höchsten Nummer, so dauert dessen Lücke 4 bis zum Erkennen des Indexlochs, womit die Spur wieder von vorne beginnt.

11 Die Programmierung der 1570/71-Diskcontroller

Sie wissen nun, wie eine Diskette aufgebaut ist, wie Daten darauf geschrieben und wieder heruntergelesen werden und wie Sie Maschinenprogramme in der Floppy zum Laufen bekommen. Was nun noch fehlt, sind Bedienungsanleitungen zur Programmierung der beiden Diskcontroller der 1570/71. Diese Anleitungen sollen in diesem Kapitel gegeben werden. Dabei fangen wir mit dem GCR-Format an.

11.1 Der Commodore-Diskcontroller

Wie Sie schon erfahren haben, handelt es sich bei diesem Baustein eigentlich gar nicht um einen Controller im üblichen Sinn. Wir haben es hier vielmehr nur mit einer Schreib- und Leselogik zu tun, die den Diskettenbetrieb steuert. Wollen wir Daten auf die Diskette bringen, so muß das softwaremäßig geschehen, was sowohl Vor- als auch Nachteile hat. Zu den Nachteilen zählt sicher der große Aufwand und die geringere Geschwindigkeit gegenüber anderen Systemen. Der große Vorteil ist allerdings, daßman beim Schreiben auf Disketten nicht an gegebene Formatanweisungen gebunden ist. Sie können sich also Ihre eigenen Diskettenformate zusammenstellen und auch wirksame Kopierschutzmethoden entwickeln.

An dieser Stelle sollen nun ein paar Standardroutinen vorgestellt werden, mit denen das Schreiben und Lesen von Daten erfolgen kann. Es sei dabei auch wieder auf das DOS- Listing verwiesen, dessen Studium sich gerade für diesen Zweck lohnt. Die Belegung der einzelnen Portregister der Schnittstellenbausteine können Sie in Kapitel 8 und Anhang B noch einmal nachlesen.

11.1.1 Das Umschalten von Lesen auf Schreiben und umgekehrt

Dieser Vorgang ist der wohl wichtigste, bevor ein Diskettenzugriff erfolgen kann. Sie sollten sich an dieser Stelle einprägen, daß der Hauptmodus der Floppy immer das Lesen ist. Wenn also Daten geschrieben werden sollen, so wird auf Schreiben umgeschaltet, dann werden die Daten geschrieben und danach muß sofort wieder auf Lesen umgeschaltet werden. Halten Sie sich nicht an dieses Schema, so kann es passieren, daß die Floppy im Schreibmodus bleibt und Ihnen solange weitere Daten oder ganze Spuren überschreibt, bis die Umschaltung erfolgt. Seien Sie also mit dem Schreiben

immer sehr vorsichtig und vergewissern Sie sich, daß nach dem Schreiben wieder auf Lesen umgeschaltet wird.

Nun kommen wir aber zu den einzelnen Routinen, zuerst zum Umschalten von Lesen auf Schreiben:

```
lda #$ff   ; alle Bits setzen und damit
sta $1c03  ; Port A (Tonkopf) auf Ausgang schalten
lda $1c0c  ; Elektronik durch
and #$1f   ; schalten des CB2 auf
ora #$c0   ; Hi-Pegel
sta $1c0c  ; auf Schreibbetrieb schalten
...        ; Schreiben von Daten sofort (!!!) beginnen
```

Nach der Ausführung dieser Befehle befindet sich die Floppy im Schreibbetrieb. Alle Daten, die jetzt an Port A des Diskcontrollers VIA 2 anliegen, werden direkt auf die Diskette geschrieben.

Da sich die Diskette mit ziemlich hoher Geschwindigkeit dreht, erfolgen auch Schreib- und Lesevorgänge entsprechend rasch. Sie haben mit einer Übertragungsrate von 31250 bis 38461 Byte pro Sekunde zu rechnen. Beachten Sie also immer, daß Ihnen bei einer System taktfrequenz von 1 MHz maximal zwischen 26 und 32 Prozessor- Taktzyklen für die Datenverarbeitung zur Verfügung stehen, bevor das nächste Byte auf die Diskette geschrieben oder von der Diskette gelesen worden ist!

Sie müssen sich unter anderem auch damit abfinden, daß während einer Umschaltung zwischen Lesen und Schreiben ein paar Bits auf der Diskette einen undefinierten Zustand erhalten, denn schließlich dreht sich die Diskette während des Umschaltvorgangs weiter.

Wenn Sie sich für die Ansteuerung der Interface-Bausteine, die jetzt besprochen werden, genauer interessieren, so schlagen Sie bitte im Anhang B nach. Dort sind von allen Bausteinen die wichtigsten Belegungen aus den Datenblättern zusammengefaßt.

Nun aber zum Umschalten auf das Lesen, das jedesmal direkt nach dem Schreiben des letzten Bytes erfolgen sollte:

```
lda $1c0c  ; Elektronik durch Umschalten des CB2
ora #$e0   ; auf Lo-Pegel auf Lesebetrieb
sta $1c0c  ; umschalten
lda #$00   ; alle Bits löschen und den Tonkopf
sta $1c03  ; auf Eingang schalten
...        ; weitere Bearbeitung...
```

Diese Routine finden Sie im DOS übrigens in identischer Form bei Adresse $fe00. Eine Routine zum Umschalten auf Schreiben existiert aus Geschwindigkeitsgründen nicht extra, sondern beinhaltet in der Regel gleich die nötigen Schreibvorgänge. (Siehe zum Beispiel ab Adresse $fe0e, wo eine Spur mit $55-Bytes vollgeschrieben, also gelöscht wird)

11.1.2 Das Schreiben eines Bytes

Nachdem Sie auf Schreiben umgeschaltet haben, wollen Sie direkt im Anschluß daran ein Byte auf die Diskette schreiben. Das kann folgendermaßen aussehen:

```
      lda #$Byte ; GCR-Byte holen
      sta $1c01  ; und zum Schreibkopf
      clv        ; Flag für BYTE READY löschen
loop  bvc loop   ; auf BYTE READY warten
      ...        ; weiter...
```

Es gibt jedoch in der 1570/71 noch eine zweite Möglichkeit:

```
      lda #$Byte ; GCR-Byte holen
      sta $lc0l  ; und zum Schreibkopf
loop  bit $180f  ; Flag für BYTE READY abfragen
      bmi  loop  ; auf BYTE READY warten
      ...        ;    weiter...
```

Diese zweite Möglichkeit wurde im Gegensatz zur 1541 bei der 1570/71 ergänzt. Wenn ein Schreib- oder Lesevorgang stattfindet, so muß der Prozessor natürlich warten, bis dieser Vorgang beendet ist, bevor er zum Beispiel ein zweites Byte schreibt oder liest. Zu diesem Zweck dient die BYTE-READY-Leitung des Diskcontrollers. Sie signalisiert das Ende eines Schreib- oder Lesevorgangs. Diese Leitung ist zum einen über den SO-Pin mit dem Prozessor verbunden. Das Overflow-Flag in dessen Statusregister wird demnach gesetzt, wenn die Leitung auf Hi geht. Andererseits wurde auch eine Verbindung zu CA1 des VIA 1 gelegt, um den Zustand der Leitung im Port A anzuzeigen.

Wichtig ist bei der Overflow-Abfrage ein Löschen des Flags nach dem Schreib- oder Lesevorgang. Bei der Portabfrage muß das Löschen nur vor dem ersten Zugriff erfolgen, um einen definierten Zustand zu erreichen. ansonsten erfolgt das Zurücksetzen des Flags automatisch durch den Diskcontroller.

Das Starten des Schreibvorgangs erfolgt jeweils automatisch mit dem Schreiben des Bytes in Port A des Diskcontrollers.

11.1.3 Das Lesen eines Bytes

Das Lesen eines Bytes geschieht analog zum Schreiben, wobei es wiederum zwei verschiedene Möglichkeiten gibt:

```
     clv        ; Flag für BYTE READY löschen
loop bvc loop   ; auf BYTE READY warten
     lda $1c01  ; Byte von Controller holen
     ...        ; zur weiteren Bearbeitung...
```

oder:

```
loop bit $180f  ; Flag für BYTE READY holen
     bmi loop   ; auf BYTE READY warten
     lda $1c01  ; Byte von Controller holen
     ...        ; zur weiteren Bearbeitung...
```

Wenn Sie also Bytes schreiben und lesen, so denken Sie bitte an die knappe Zeit von 26 bis 32 Taktzyklen. Eine größere Bearbeitung eines Bytes ist daher nicht sofort möglich, sondern Sie müssen zuerst alle gewünschten Bytes einlesen, in einen Puffer schreiben und anschließend bearbeiten.

11.1.4 Das Schreiben einer SYNC-Markierung

Der folgende Vorgang gehört natürlich mit zu den wichtigsten beim Diskettenbetrieb. Ohne vorangehende SYNC-Markierung fällt es in der Regel schwer, geschriebene Daten auf einer Spur wiederzufinden. Wenn Sie eine SYNC-Markierung schreiben wollen, dann schalten Sie vorher auf Schreiben um und führen direkt danach folgende Befehle aus:

```
     ldx #$05   ; 5 Bytes für eine SYNC-Markierung
start lda #$ff  ; Byte %11111111 für SYNC-Markierung
     sta $1c01  ; Byte zum Diskcontroller
     clv        ; Flag für BYTE READY löschen
loop bvc loop   ; Schreib organg abwarten
     dex        ; nächstes Byte
     bne start  ; und weitermachen, bis SYNC fertig
     ...        ; weiter...
```

Das Idx #$05 sollten Sie sogar noch möglichst vor dem Umschalten auf Schreiben ausführen. Nach dem Umschalten auf Schreiben (und auch auf Lesen) ist es unbedingt notwendig, so schnell wie möglich das Byte auf die Diskette zu schreiben, um "undefinierte" Zonen, die das Timing des Diskcontrollers durcheinander bringen können, so klein wie möglich zu halten (vielleicht 1 bis 3 Bits maximal).

Natürlich können Sie auch diesmal wieder die zweite Methode zum Schreiben der SYNC-Markierung verwenden:

```
      ldx #$05   ; 5 Bytes für eine SYNC-Markierung
start lda #$ff   ; Byte %11111111 für SYNC
      sta $1c01  ; zum Diskcontroller schicken
loop  bit $180f  ; BYTE READY abfragen
      bmi loop   ; auf BYTE READY warten
      dex        ; nächstes Byte
      bne start  ; und weitermachen
      ...        ; weiter...
```

11.1.5 Das Suchen einer SYNC-Markierung

Hier ist das Gegenstück zum Schreiben der SYNC-Markierung, das Suchen. Von Lesen zu sprechen ist in diesem Fall nicht angebracht, da eine SYNC-Markierung vom Diskcontroller nicht mehr so gelesen werden kann, wie das bei einem GCR - Byte der Fall ist.

Eine auftretende SYNC-Markierung löst beim Diskcontroller das hardwaremäßig festgelegte SYNC-Signal aus. Während dieses Signal auftritt, wird der Schreib-/Leseport des Diskcontrollers gesperrt, das heißt, nicht mehr mit Daten versorgt - und auch kein BYTE READY mehr ausgelöst -, bis die SYNC-Zone auf der Diskette vorbei ist. Aus diesem Grund enthält der VIA 2 in seinem Port zum Schreib-/Lesekopf nach dem Auftreten des SYNC-Signals einen undefinierten Wert. Dieser entspricht der Bitfolge, die bis zur Feststellung des SYNC-Bereichs durch den Controller eingelesen wurde.

Zu diesem Sperren während eines SYNC-Signals eine kurze Bemerkung: Vielleicht haben Sie schon einmal etwas von "Killertracks" gehört? Das sind Spuren, bei denen sich die Floppy scheinbar "aufhängt", wenn Sie versucht, etwas davon einzulesen. Diese Spuren sind nichts weiter als eine SYNC-Markierung, die über die gesamte Spur geschrieben wurde. Versucht der Controller nun einen Lesezugriff, so stellt er fest, daß hier gerade eine SYNC-Markierung vorhanden ist. Er sperrt also das Lesen von Daten und wartet, bis diese Markierung vorüber ist. Bei einer durchgehenden Spur gibt es aber kein Ende, und so dauert dieses Warten ewig! Die Software zum Lesen hängt dabei natürlich in der Leseschleife zum Abwarten des BYTE-READY-Signals und da dieses Signal nie kommt, kommt auch das Programm nicht weiter: die Floppy scheint "abgestürzt" zu sein. Bei einem solchen Problem hilft in der Regel das Öffnen des Laufwerkschachtes. Der Schreib-/Lesekopf empfängt dann keine Daten mehr und die Floppy meldet sich mit einer Fehlermeldung zurück.

Nun aber zu einer Routine, die das Abwarten einer SYNC-Markierung vornimmt. In ihrer einfachsten Form siht sie folgendermaßen aus:

```
loop  bit $1c00  ; Port A lesen und Bit 7 (SYNC) testen
      bmi loop   ; SYNC (Lo-aktiv) abwarten
      lda $1c01  ; Port B für Daten wieder freinachen
      ...        ; weiter...
```

Diese einfache Form verwendet man jedoch im DOS nicht. Es kann ja passieren, daß eine unformatierte Diskette im Laufwerk liegt, die keine SYNC-Markierungen enthält. Die Floppy würde sich dann "aufhängen", da kein SYNC-Signal erscheint. In der Praxis setzt man daher einen Zähler oder einen Timer ein. In der 1570/71 sind beide Versionen vorhanden. Im DOS der 1541 verwendet man den Timer 1 von VIA 1 als Zähler, und im ergänzten DOS der 1570/71 verwendet man das X- und das Y- Register. Nachfolgend sind beide Versionen abgedruckt, zuerst die Version der 1570/71:

```
      ldx #$0f   ; Zähler Hi setzen
      ldy #$00   ; Zähler Lo setzen; 3848 Versuche
loop  bit $1c00  ; SYNC aufgetreten?
      bpl end    ; verzweige, wenn ja
      dey        ; sonst Zähler Lo vermindern
      bne loop   ; und weitersuchen
      dex        ; Zähler Hi vermindern
      bne loop   ; und weitersuchen
      jmp error  ; "21, READ ERROR"; SYNC not found
end   lda $1c01  ; Port wieder freimachen
      ...        ; weiter; alles ok...
```

jetzt die Version im 1541-Modus:

```
      lda #$d0   ; Tinerwert für ca. 53 ns
      sta $1805  ; Timer Hi setzen und starten
      lda #$03   ; Vergleichswert für Ende
loop  bit $1805  ; Timer abgelaufen?
      bpl error  ; ja; Fehler
      bit $1c00  ; SYNC gefunden?
      bmi loop   ; weitermachen, wenn nein
      lda $1c01  ; Port wieder freinachen
      ...        ; weiter; alles ok
error jmp error  ; zur Fehlerbehandlung: "21, READ ERROR"
```

Die erste Version können Sie theoretisch immer verwenden. Bei der zweiten müssen Sie vorsichtig sein, wenn Sie im 1570/71-Modus arbeiten, da die Systemtaktfrequenz hier 2 MHz beträgt und der Timer dementsprechend zu schnell herunterzählt. Es ist übrigens nicht mehr möglich, den Timer 1 des VIA 1 mit einem entsprechend doppelt so hohen Wert zu laden, da der Maximalwert, der als Timerwert Hi abgespeichert werden kann, $ff beträgt.

Mit dieser Bemerkung haben Sie jetzt praktisch alles, was Sie für die Programmierung des GCR-Controllers benötigen. Der Rest ist praktische Erfahrung und Studium des DOS-Listings. In kürzester Zeit werden Sie dann in der Lage sein, schnelle Kopierprogramme, Kopierschutzmethoden, Befehlserweiterungen oder Spezialformate zu entwickeln - und das alles auf einer Floppy.

11.2 Der WD 1770 Diskcontroller von Western Digital

Ohne Sie gleich in Panik versetzen zu wollen, muß gesagt werden, daß die Programmierung dieses Diskcontrollers nicht ganz so einfach ist wie die des Commodore-Diskcontrollers. Das hat folgenden Grund: Der WD 1770 stellt eine Gesamteinheit dar, die in der Lage ist, eine gesamte Diskettensteuerung vollkommen selbständig durchzuführen. Zum Betrieb dieses Controllers ist aus diesem Grund nur wenig Software, dafür aber fundiertes Wissen über seine Funktionsweise Voraussetzung.

Im Prinzip werden Kommandos an den WD 1770 wie vom Hauptprogramm an die Jobschleife der 1570/71 übergeben, wobei jedoch ein paar Dinge zu berücksichtigen sind, die wir jetzt gleich besprechen werden.

11.2.1 Die Kommandos des WD 1770

Im Anhang B finden Sie die wichtigen Daten über den WD 1770 in einer Tabellensammlung zusammengefaßt, die Ihnen später das nötige Nachschlagen erleichtern soll. In diesem Kapitel werden ein paar dieser Daten jedoch noch einmal aufgeführt.

Zuerst befassen wir uns mit den Registern des WD 1770. Er besitzt im großen und ganzen nur vier, die nachfolgend in einer kleinen Tabelle dargestellt sind:

Register	Adresse	Schreibfunktion	Lesefunktion
0 $0	$2000	Kommandoregister	Statusregister
1 $I	$2001	Spurregister	Spurregister
2 $2	$2002	Sektorregister	Sektorregister
3 $3	$2003	Daten zum Kopf	Daten vom Kopf

Uns interessiert das Register 0 zuerst. Wie Sie feststellen können, liegt dieses Register im Adreßbereich der Floppy bei Adresse $2000 (Grundadresse des WD 1770) und hat beim Schreiben eine andere Funktion als beim Lesen. Wir beschäftigen uns zuerst mit der Schreibfunktion.

Das Register 0 empfängt die Befehle an den WD 1770, wobei der Controller vier verschiedene Befehlstypen unterscheidet. Diese Befehlstypen und ihre Steuercodes sind in der nachfolgenden Tabelle aufgeführt:

Typ	Name des Kommandos	Bitbelegung des Kommandoregisters							
		7	6	5	4	3	2	1	0
1	Restore	0	0	0	0	h	V	r1	r0
1	Seek	0	0	0	1	h	V	r1	r0
1	Step	0	0	1	u	h	V	r1	r0
1	Step-in	0	1	0	u	h	V	r1	r0
1	Step-out	0	1	1	u	h	V	r1	r0
2	Read Sector	1	0	0	m	h	E	0	0
2	Write Sector	1	0	1	m	h	E	P	a0
3	Read Address	1	1	0	0	h	E	0	0
3	Read Track	1	1	1	0	h	E	0	0
3	Write Track	1	1	1	1	h	E	P	0
4	Force Interrupt	1	1	0	1	i3	i2	i1	i0

Dabei haben die Bitkennzeichnungen folgende Bedeutungen:

Zeichen	Bedeutung des entsprechenden Bits
h	Motor On Flag: 0: Motor bekommt keine Hochlaufzeit 1: Motor bekommt 6 Umdrehungen Zeit zum Anlaufen
V	Verify Flag: 0: kein Verify durchführen 1: Verify auf gegebener Spur durchführen
r1, r0	Stepping rate: Gibt die Zeit an, in der zwei Impulse für den Steppermotor aufeinanderfolgen; Steppertaktrate: 00: alle 6 ms 01: alle 12 ms 10: alle 20 ms 11: alle 30 ms Bits haben in der 1570/71 keine Bedeutung, da der Steppermotor softwaremäßig gesteuert wird!
u	Update flag: 0: keine Änderung des Spurregisters 1: Spurregister wird bei Stepperbewegung automatisch auf den richtigen Stand gebracht

Zeichen	Bedeutung des entsprechenden Bits
m	Multiple Sector Flag: 0: nur ein Sektor soll behandelt werden 1: mehrere Sektoren sollen behandelt werden
a0	Data Address Mark: 0: Data Address Mark für gültigen Sektor setzen 1: Data Address Mark für gelöschten Sektor setzen
E	30ms Settling Delay: 0: keine Verzögerung 1: wartet nach Kopfpositionierung 30 ms ab
P	Write Precompensation: 0: keine Write Precompensation 1: Write Precompensation an
i3-i0	Interrupt Condition: Bits setzen die Interruptbedingung bei einem Force Interrupt Kommando (Typ 4-) fest: i0: 1= keine Bedeutung i1: 1= keine Bedeutung i2: 1= Interrupt bei Erkennung des Indexlochs i3: 1= sofortiger Interrupt alle 0: kein Interrupt

Tabelle 11.1: Der Befehlssatz des WD 1770

11.2.1.1 Die Kommandos vom Typ 1

Wir wollen nun alle Kommandos des WD 1770 ein wenig genauer unter die Lupe nehmen und fangen dabei mit den Kommandos des Typs 1 an.

Das Restore- Kommando:

Dies Kommando fährt den Schreib- jLesekopf auf Spur 0 zurück, bis die Lichtschranke für den Anschlag unterbrochen wird. Ist dann das Verify F1ag gesetzt, so wird zusätzlich geprüft, ob Spur 0 ordnungsgemäß formatiert ist. Bei der 1570/71 hat dieses Kommando keine Bedeutung, da der WD 1770 keine Möglichkeit der Steppermotorsteuerung hat. Die Leitungen für die Steppermotorsteuerung wurden von Commodore erstaunlicherweise nicht verwendet und liegen deshalb brach. Erstaunlicherweise deshalb, weil der WD 1770 eine vollständige Logik für die Steppersteuerung enthält und alles vollautomatisch übernommen hätte. Commodore hat es indessen vorgezogen, diese Steuerung sehr umständlich vom Betriebssystem vornehmen zu lassen.

Merken Sie sich also bei der Programmierung des WD 1770, daß sich der Schreib- /Lesekopf der Floppy schon in der richtigen Position befinden muß, bevor der Controller aufgerufen wird, da dieser keinen Einfluß auf die Kopfbewegungen nehmen kann.

Das Seek- Kommando:

Dies Kommando erwartet im Spurregister die gewünschte Spurnummer und positioniert den Schreib-/Lesekopf dann entsprechend. Ist das Verify F1ag gesetzt, so wird die Spur zusätzlich auf korrekte Formatierung untersucht.

Das Step-Kommando:

Dies Kommando bewegt den Kopf um eine Spur in die Richtung, in die auch die letzte Bewegung erfolgt ist. r0 und r1 geben dabei die Steppergeschwindigkeit an. Das Verify Flag sorgt gegebenenfalls für eine Prüfung der gewünschten Spur.

Das Step-in-Kommando:

Dies Kommando positioniert den Kopf auf die nächsthöhere Spurnummer. Wenn das Update Flag gesetzt ist, wird die Spurnummer im Spurregister entsprechend erhöht.

Das Step-out-Kommando:

Dies Kommando wirkt wie ein Step-in, nur wird der Kopf um eine Spur nach außen (Richtung Spur 0) bewegt.

Wie Sie sehen, haben alle Kommandos vom Typ 1 etwas mit der Steppersteuerung zu tun: Spurnummer ins Spurregister, Seek-Kommando an den DiskcontroIler, und schon würde der WD 1770 den Kopf auf die richtige Spur positionieren, wenn die Leitungen dafür angeschlossen wären ...

Vielleicht ist der eine oder der andere Bastler unter Ihnen, der sich gerne einmal etwas näher mit diesem Problem auseinandersetzt? Für einen Hardware-Spezialisten dürfte es kein Problem sein, die 1570/71 in dieser Richtung ein wenig aufzurüsten, und dem WD 1770 die Steppersteuerung zu ermöglichen.

11.2.1.2 Die Kommandos vom Typ 2

Die Kommandos vom Typ 2 dienen dem Lesen und Schreiben von Sektoren und werden vom DOS 3.0 der 1570/71 am häufigsten eingesetzt.

Die Kommandos des Typs 2 erfordern jeweils die gewünschte Spur- und Sektornummer im Spur- und Sektorregister des WD 1770. Sobald der WD 1770 ein Kommando vom Typ 2 empfängt, wird das Busy-Flag gesetzt (dazu später mehr). Ist das E-Bit im Kommandobyte gesetzt, so wartet der Controller 30 ms, bevor er sich an die Ausführung des Kommandos macht.

Startet das Kommando, so sucht sich der Controller zuerst einmal den Reader des gewünschten Sektors. Es wird auf die richtige Spur- und Sektornummer im Vorspann des Sektors und auf die Prüfsumme, das CRC-Byte, geachtet. Sind diese Bedingungen nicht erfüllt, so meldet der Controller einen "Sector not found"-Status und bricht das Kommando ab. Andernfalls kommt eines der folgenden Kommandos zur Ausführung.

Das Read - Sector- Kommando:

Nachdem das korrekte CRC-Byte gelesen wurde, erwartet der Controller nun innerhalb von 43 Bytes den Beginn des folgenden Datenblocks (Data Address Mark). Kommt der nicht, so wird das Kommando mit dem Status "Sector not found" abgebrochen. Andernfalls werden die Datenbytes eingelesen und im Datenregister ($2003) bereitgestellt. Danach wird im Statusregister das DRQ- Bit gesetzt, das das bereitstehende Byte signalisiert. Der Computer hat nun bis zum Eintreffen des nächsten Bytes Zeit, das jetzige Datenbyte auszulesen und abzuspeichern; ansonsten schreibt der WD 1770 das folgende Byte ins Datenregister und setzt im Statusregister das Lost Data Flag, das ein verlorengegangenes Byte anzeigt. Auf diese Weise wird der gesamte Sektor eingelesen und dem Prozessor übergeben. Danach kontrolliert der Diskcontroller noch das abschließende CRC-Byte für den Datenblock und setzt gegebenenfalls einen "CRC Error"-Status, wenn ein Fehler aufgetreten ist. Anschließend wird noch der Typ der gelesenen Data Address Mark ins Statusregister geschrieben, der anzeigt, ob der gelesene Sektor gültig oder gelöscht war.

Das Write-Sector- Kommando:

Nach dem Lesen des CRC-Bytes, wartet der Controller noch 22 Bytes ab, bevor er auf Schreiben umschaltet. Ist das Datenregister ($2003) bis dahin vom Computer nicht mit dem ersten Byte zum Schreiben gefüllt worden, so wird der Lost Data Status angezeigt und das Kommando abgebrochen. Ansonsten werden die 12 Bytes der SYNC-Markierung geschrieben. Anschließend folgt das Schreiben der Data Address Mark, die den Datenblockbeginn ankündigt. Soll der Datenblock als gelöscht oder ungültig deklariert werden, so muß im Kommandobyte das a0-Bit (Bit 0) gesetzt sein. Sonst wird in die Data Address Mark der Status eines gültigen Sektors eingetragen. Der WD 1770 schreibt nun den Datenblock auf die Diskette und

erwartet dazu vom Prozessor die Bereitstellung der Daten im Datenregister. Erfolgt das nicht, so wird das Kommando nicht abgebrochen, sondern der Controller schreibt den Rest des Datenblocks mit $00 voll und bildet das korrekte CRC-Byte, das ebenfalls auf die Diskette geschrieben wird. 24 Mikrosekunden danach wird der Befehl für beendet erklärt.

11.2.1.3 Die Kommandos vom Typ 3

Diese Kommandos befassen sich mit einer ganzen Spur einer Diskette und behandeln auch das Formatieren. Bei allen Kommandos vom Typ 3 wird das Busy-Flag direkt nach dem Empfang des Kommandos gesetzt und das Kommando ausgeführt.

Das Read-Address-Kommando:

Wenn der Controller dieses Kommando erhält, wird sofort das nächste verfügbare ID-Feld einer Spur eingelesen, wobei dem Prozessor die folgenden sechs Bytes übergeben werden:

Byte 1	Spurnummer
Byte 2	Seitennummer der Diskette
Byte 3	Sektornummer
Byte 4	Sektorgröße
Byte 5	1. CRC- Byte
Byte 6	2. CRC-Byte.

Obwohl der Prozessor diese Werte erhält, werden sie vom Controller geprüft und gegebenenfalls der "CRC Error"-Status gesetzt.

Das Read-Track-Kommando:

Mit diesem Kommando werden sämtliche Bytes einer gesamten Spur von der Diskette gelesen und ohne Fehlerprüfung direkt in das Datenregister ($2003) geschrieben. Der Prozessor kann sie dort abholen und zum Beispiel eine Diagnose der Spur erstellen. Da sämtliche Bytes gelesen werden, kann es passieren, daß einige Fehlbytes an den Stellen eingelesen werden, die der Controller jedes mal zum Umschalten zwischen Schreiben und Lesen verwendet. Dabei handelt es sich jedoch immer um eine der 4 Lückentypen, so daß der Fehler nicht ins Gewicht fällt. Nach einer Diskettenumdrehung ist das Kommando beendet.

Das Write-Track-Kommando:

Dieses Kommandobyte dient zum Formatieren einer Spur auf der Diskette. Die Ausführung dieses Kommandos beginnt mit dem Erkennen des Index

lochs, das den Anfang einer Spur kennzeichnet, und ist mit dem erneuten Auftreten des Indexloch-Impulses beendet, da dann eine volle Diskettenumdrehung stattgefunden hat. Die Bytefolge, die dabei auf die Diskette geschrieben wird, können Sie sowohl dem DOS-Listing (ab Adresse $8a86), als auch dem Aufbau einer Spur beziehunsweise eines Sektors in Kapitel 10 entnehmen.

11.2.1.4 Die Kommandos vom Typ 4

Diese Kommandos bestehen eigentlich aus einem einzigen Kommando mit unterschiedlichen Möglichkeiten. Es handelt sich hier darum, den WD 1770 durch einen Interrupt zum Abbrechen eines laufenden Kommandos zu bewegen. Die Voraussetzung ist dabei, daß das Abbruchkommando mindestens 16 Mikrosekunden im Kommandoregister anliegen muß, sonst wird die Interruptanforderung ignoriert. Die Bits 0 bis 3 bestimmen dabei im Kommandobyte die Interruptbedingung (siehe Tabelle 11.1):

Bit 0	sofortiger Interrupt
Bit 1	Interrupt nach dem nächsten Erkennen des Indexlochs
Bit 2	keine Bedeutung
Bit 3	keine Bedeutung

Die Bits müssen für die entsprechende Bedingung gesetzt werden. Das Kommando wird beim Überschreiben des Kommandobytes im Kommandoregister ignoriert, wenn die 16 Mikrosekunden noch nicht abgelaufen sind.

11.2.2 Die Rückmeldungen des WD 1770

Bisher haben wir uns mit den Befehlen an den WD 1770 beschäftigt. Es muß aber auch möglich sein, die Rückmeldungen dieses Controllers empfangen zu können, damit das DOS weiß, wann Fehler aufgetreten sind oder das Kommando ordnungsgemäß ausgeführt wurde.

Zu diesem Zweck gibt es im WD 1770 das Statusregister, das genau die gleiche Adresse wie das Kommandoregister ($2000) besitzt, und sich bei Lesevorgängen an dieser Adresse einblendet. Die Belegungen dieses Registers sind folgende (siehe auch Anhang B):

Bit	Bedeutung
0	Busy Flag: 0: Controller bereit für Befehlsempfang 1: Kommandos der Typen 2 und 3 werden ausgeführt
1	Data Request/Index Flag: 0: zeigt an, das das Datenregister aktuell ist 1: zeigt an, wenn ein Byte gelesen oder geschrieben wurde Bei Typ 1: zeigt Zustand der Indexlochlichtschranke an 1= nicht unterbrochen
2	Lost Data/Track0 Flag: 0: ok 1: Daten beim Lesen oder Schreiben verlorengegangen Bei Typ 1: zeigt Spur 0 Postion des Kopfes an 1= Kopf nicht auf Spur 0
3	CRC Error Flag: 0: ok 1: CRC Error aufgetreten
4	Record Not Found Flag: 0: ok 1: Record not found Meldung yon Controller
5	Record Type/Spin Up Flag: 0: Data Address Mark für normalen Sektor setzen 1: Data Address Mark für ungültigen Sektor setzen Bei Typ 1: Zeigt an, daß Motor 6 Umdrehungen beendet hat 1= Hochlaufphase beendet
6	Write Protect Flag: 0: Ok 1: Diskette schreibgeschützt (nur bei Schreiboperationen)
7	Motor On Flag: 0: Motor ausgeschaltet 1: Motor eingeschaltet

Tabelle 11.2: Die Belegung des Statusregisters des WD 1770

12 Floppy 1570/71 Spezial

Im nun folgende Kapitel geht es um die Eigenschaften der 1570/71, die im Handbuch nicht erwähnt werden, obwohl es die herausragenden Merkmale dieser neuen Floppies schlechthin sind. Gemeint ist die Arbeit mit der 1570/71 im schnellen Busmodus, der als Burst-Modus bezeichnet wird.

Wie die 1541, so besitzt auch die 1570/71 einen seriellen Bus. Dieser Bus kann jedoch auf das Vielfache seiner ursprünglichen Geschwindigkeit (der Geschwindigkeit im 1541-Modus) gebracht werden, wenn man die Regeln dafür kennt.

Für diesen sehr schnellen Busbetrieb gibt es in der 1570/71 einen kompletten zusätzlichen Befehlssatz, der wie schon gesagt, im Handbuch mit keinem Wort erwähnt wird. Dieser Befehlssatz ist unter dem Namen U0-Befehle zusammengefaßt und soll nun unter die Lupe genommen werden.

12.1 Die USER-0-Befehle (U0)

Wenn Sie sich noch an Kapitel 7.11 erinnern, so werden Sie wissen, daß die Benutzer-Befehle (USER-Befehle) dort schon einmal erklärt wurden. Es handelte sich dabei um die Befehle U1 bis U:, die dort ausführlich erläutert wurden. Ein einziger Benutzer-Befehl wurde jedoch weggelassen: der U0-Befehl. Diesem Befehl sind nämlich völlig neue und vielseitige Funktionen zugewiesen worden, weshalb wir ihn jetzt betrachten wollen.

Sie finden übrigens in diesem Buch einen hervorragenden Diskmonitor zum Abtippen. Dieser BURSTMON 3.0 arbeitet fast ausschließlich mit den U0-Befehlen und erreicht in der Verarbeitung Geschwindigkeiten, die Sie bei einem vergleichbaren Programm kaum finden werden (siehe Kapitel 13).

Es sei an dieser Stelle vor der Besprechung der einzelnen Burst-Kommandos noch erwähnt, daß Sie für den Betrieb der meisten dieser Befehle. spezielle Empfangs- und Senderoutinen im Computer benötigen. Diese Routinen arbeiten außerordentlich schnell und sind deshalb natürlich in Maschinensprache geschrieben. Im Anschluß an die Befehle werden diese Routinen ebenfalls aufgeführt.

12.1.1 Das BURST-READ-Kommando

Wenn Sie dieses Kommando an die Floppy senden, so werden eine oder mehrerer Sektoren sehr schnell von der Diskette gelesen, wobei diese über den schnellen Busbetrieb sofort zum Computer übertragen werden können. Mit den herkömmlichen Floppyroutinen und den nachträglich vorgestellten Busroutinen für den Computer können Lesegeschwindigkeiten unter einer Sekunde für eine komplette Spur erreicht werden, wobei im GCR-Format 21 Sektoren zum Computer übertragen werden.

Die nachfolgende Tabelle zeigt die Belegung und Bedeutung der einzelnen Bits. Der Befehl wird ganz normal in einem Befehlsstring über den Kommandokanal an die Floppy gesendet.

Byte Bedeutung

00 entspricht dem Zeichen U mit dem Wert $55

01 entspricht dem Zeichen 0 mit dem Wert $30

02 Befehlsbyte, die Bits haben folgende Bedeutung:
- Bit 0: Drivenummer (immer 0)
- Bit 1: immer 0
- Bit 2: immer 0
- Bit 3: immer 0
- Bit 4: Wahl der Diskettenseite unter MFM (0 oder 1)
- Bit 5: nur Pufferübertragung; bei 1 wird dem Computer nur der Pufferinhalt, in dem die gelesenen Sektoren normalerweise stehen, übertragen
- Bit 6: bei 1 werden eventuelle Lesefehler ignoriert
- Bit 7: bei 1 wird ein Sektor nur in den Puffer gelesen und nicht zum Computer übertragen

03 Gewünschte Spurnummer zum Lesen

04 Gewünschte Sektornummer zum Lesen

05 Anzahl der Sektoren, die gelesen werden soIlen

06 Spurnummer, auf die der Kopf danach positioniert werden soll.
Diese Angabe ist optional

Busverkehr: Senden des Kommandos über den Befehlskanal an die Floppy. Die Floppy sendet darauf jeweils ein Statusbyte und nachfolgend den Sektorinhalt zum Computer.

Tabelle 12.1: Die Struktur des BURST -READ-Kommandos

Wie Sie sehen, geht dem eigentlichen Kommandobyte immer die Zeichenfolge U0 voran, und das wird auch bei allen nun folgenden Befehlen der Fall sein. Wo die einzelnen Befehle im DOS stehen, können Sie ab Adresse $8030 nachlesen. Die erste Routine des DOS 3.0 ist nämlich die Zusatzbehandlung für die erweiterten Funktionen des U0-Befehls.

Die Funktionen der einzelnen Bits erklären sich eigentlich von selbst, bis auf Bit 5, 6 und 7.

Wenn im Kommandobyte das Bit 5 gesetzt ist, so wird kein Sektor von der Diskette eingelesen, sondern es wird lediglich der Pufferinhalt zum Computer übertragen, in dem die Sektoren üblicherweise stehen. Im Burst-Modus verwendet die Floppy für Schreib- und Leseoperationen immer den Adreßbereich ab $0300. Das Ende des Bereiches hängt von der Größe des zu behandelnden Sektors ab. Bei GCR-Sektoren ist das Ende immer $03ff; bei MFM-Sektoren der Länge 1024 Bytes liegt das Ende bei $06ff.

Das gesetzte Bit 6 ermöglicht das Ignorieren von eventuell auftretenden Lesefehlern. Sie werden hier also in die Lage versetzt, auch defekte Sektoren ordnungsgemäß einzulesen, wenn das DOS noch in der Lage ist, diese von der Diskette zu laden.

Das gesetzte Bit 7 erlaubt das Lesen eines Sektors, ohne daß dieser zum Computer übertragen wird. Zusammen mit der Funktion von Bit 5 können Sie einen Sektor erst einmal von der Diskette lesen und ihn zu einem späteren Zeitpunkt in den Computer übernehmen.

Im Normalbetrieb sind die Bits 5, 6 und 7 auf 0. Es wird dann ein Sektor von der Diskette gelesen und direkt danach zum Computer übertragen.

Die folgenden Parameterbytes sind eindeutig, bis auf die Bytes 5 und 6.

Durch Byte 5 kann die Anzahl der Sektoren angegeben werden, die im Burst-Modus direkt hintereinander gelesen werden sollen. Zu beachten ist dabei, daß die Sektornummer der folgenden Sektoren vom gesetzten Sektorabstand (sector interleave) abhängig ist. Das Setzen dieses Abstands ist auch ein BURST-Kommando. Steht der Interleave- Wert also auf 1, so wird immer der physikalisch nächste Sektor von der Diskette gelesen (z.B. nach Sektor 0 kommt 1, 2, 3, usw.). Ist der Interleave-Wert auf 5 und Sie lesen als ersten Sektor zum Beispiel Sektor 18, 0, so wird als nächster Sektor 18, 5 und danach gegebenenfalls Sektor 18, 10 und so weiter gelesen.

Byte 6 gibt die Spurnummer an, auf die der Kopf nach der Ausführung des BURST-READ positioniert werden soll. Das erlaubt sehr schnelle Lesevorgänge auf unterschiedlichsten Spuren.

Wenn die Diskette gewechselt wurde, so ist vor dem Lesevorgang ein INQUIRE-DISK-Kommando durchzuführen. Das entspricht dem Initialisieren von GCR-Disketten und ist sowohl auf das GCR-Format als auch auf das MFM-Format anwendbar. Wird dieses Kommando nicht gegeben, so meldet der Controller den Fehlerstatus "disk id mismatch or disk change". Das INQUIRE-DISK-Kommando werden wir in diesem Kapitel noch behandeln.

Noch eine Bemerkung zum Busverkehr: Wurde das BURST-READ-Kommando an die Floppy gesendet, so muß sich der Computer für den Empfang von Burst-Daten bereitmachen. Das sind Daten, die über den schnellen seriellen Bus gesendet werden. Die Floppy sendet bei BURST-READ in jedem Fall ein Statusbyte, dem dann auf Wunsch der Sektor folgt. Werden mehrere Sektoren gelesen, so sendet die Floppy immer nach Lesen eines Sektors das entsprechende Statusbyte und danach die Sektordaten.

12.1.2 Das BURST-WRITE-Kommando

Hier handelt es sich um das Gegenstück zum BURST-READ-Kommando. Die Syntax dieses Befehls ist der des Lesekommandos sehr ähnlich und lautet folgendermaßen:

Byte Bedeutung

00 entspricht dem Zeichen U mit dem Wert $55

01 entspricht dem Zeichen 0 mit dem Wert $30

02 Befehlsbyte, die Bits haben folgende Bedeutung:
- Bit 0: Drivenummer (immer 0)
- Bit 1: immer 1
- Bit 2: immer 0
- Bit 3: immer 0
- Bit 4: Seitenwahl bei MFM-Format (Seite 0 oder 1)
- Bit 5: 1 bedeutet, Computer schreibt nur Daten in den Puffer ab $0300; kein Diskettenzugriff
- Bit 6: 1 bedeutet, Schreib- und Lesefehler ignorieren
- Bit 7: 1 bedeutet, daß nur der Sektor geschrieben wird. Die Floppy erwartet die Daten schon im Puffer.

03 Gewünschte Spurnummer für Schreiben

04 Gewünschte Sektornummer für Schreiben

05 Anzahl der Sektoren zum Schreiben

07 Nächste Spur für Kopfpositionierung nach Ende des Schreibens.

Busverkehr: Der Computer muß nach dem Senden des Kommandos den Busverkehr auf Burst-Betrieb umstellen und gegebenenfalls die Daten für den Sektor an die Floppy schicken. Nach dem Schreiben eines jeden Sektors gibt die Floppy ein Statusbyte aus.

Tabelle 12.2: Die Struktur des BURST - WRITE- Kommandos

Die Bedienung des BURST-WRITE-Kommandos erfolgt analog zu der des BURST-READ-Kommandos. Auch die einzelnen Kommandobits haben die gleiche Bedeutung. Lediglich Bit 5 und Bit 7 beziehen sich jetzt auf das Schreiben von Daten und nicht auf das Lesen. Das heißt, ist Bit 5 gesetzt, so wird der schon vorhandene Pufferinhalt auf die Diskette geschrieben, ohne daß zuvor Daten vom Computer geholt werden. Ist Bit 7 gesetzt, so holt sich die Floppy lediglich einen vollständigen Pufferinhalt vom Computer, wobei dieser nicht auf die Diskette geschrieben wird.

Mit den Flags für das "nur in den Puffer schreiben" oder "nur den Puffer auslesen", ist es für Sie also auch ganz bequem möglich - vor allem mit hoher Geschwindigkeit -, Maschinenprogramme in den Puffer der Floppy zu schreiben oder Daten wieder herauszulesen.

12.1.3 Das INQUIRE-DISK-Kommando

Dieses Kommando oder das QUERY-DISK-Kommando (siehe 12.1.7), muß nach jedem Diskettenwechsel und nach dem Formatieren gegeben werden, um die Diskette zu initialisieren (log in the diskette), wenn anschließend ein Burst-Kommando an die Floppystation geschickt wird, das einen Diskettenzugriff vornimmt. Wird dieses Initialisieren unterlassen, so erfolgt die Statusmeldung "disk id mismatch or disk changed" vom Controller.

An dieser Stelle sei darauf hingewiesen, daß diese Initialisierung nichts mit dem Floppykommando I zu tun hat. Das I-Kommando dient der Initialisierung einer GCR-formatierten Diskette, wobei deren BAM gelesen wird. Das INQUIRE-DISK-Kommando hingegen fährt den Kopf auf Spur 0 (beziehungsweise 1) und liest dann die wichtigen Diskettenparameter aus dem nächstbesten Blockheader. Hierbei muß natürlich zwischen GCR- und MFM-Format unterschieden werden. Im GCR-Format wird der Jobcode zum Suchen eines Sektors ausgeführt und dabei die ID geladen; wohingegen im MFM-Format die Sektorparameter gesetzt werden, als da sind: Anzahl der Sektoren pro Spur, Größe der Sektoren, Nummer des kleinsten und größten Sektors einer Spur.

Byte	Bedeutung
00	entspricht dem Zeichen U mit dem Wert $55
01	entspricht dem Zeichen 0 mit dem Wert $30
02	Kommandobyte, wobei die Bits folgendermaßen zu setzen sind: Bit 0: Drivenummer (immer 0) Bit 1: immer 0 Bit 2: immer 1 Bit 3: immer 0 Bit 4: Seitennummer bei MFM-Diskette Bit 5: egal Bit 6: egal Bit 7: egal
Busverkehr: Computer muß nach dem Senden des Kommandos auf den Burst-Modus umschalten. Die Floppy gibt nach der Ausführung des Befehls ein Statusbyte aus.	

Tabelle 12.3: Die Struktur des INQUIRE-DISK-Kommandos

Wie fast alle Burst-Befehle, so gibt auch das INQUIRE-DISK-Kommando nach seiner Ausführung ein Statusbyte an den Computer aus. Dieses Statusbyte ist für die U0-Befehle typisch.

12.1.4 Das FORMAT-Kommando

Mit diesem Burst-Kommando ist es möglich Disketten sowohl im GCR, als auch im MFM-Format zu formatieren. Es handelt sich hierbei jedoch um ein reines Formatieren, das heißt, es wird weder im GCR noch im MFM-Format ein Directory auf die Diskette gebracht. Hier ist zuerst das Formatieren im MFM-Format. Die Kommandofolge sieht folgendermaßen aus:

Byte	Bedeutung
00	entspricht dem Zeichen U mit dem Wert $55
01	entspricht dem Zeichen 0 mit dem Wert $30
02	Kommandobyte, wobei die Bits folgende Bedeutung haben: Bit 0: Drivenummer (immer 0) Bit 1: immer 1 Bit 2: immer 1 Bit 3: immer 0 Bit 4: Seitenummer der Diskette 0 oder 1 Bit 5: Flag für Formatieren; 1=beidseitig; 0=einseitig Bit 6: Flag für Index Address Mark; 1= schreiben; 0= nicht schreiben Bit 7: immer 0 bei MFM
03	Steuerbyte für Formatierung: Bits 0 bis 5 enthalten den logischen Startsektor Bit 6: Anzeige der Sektortabelle: 1= Sektortabelle wird vom Computer übergeben 0= Sektortabelle wird von der Floppy selbst erstellt. Bit 7: Flag für Formatierung; bei MFM-Format immer 1
04	Sektorabstand für Hardware (sector interleave) (Angabe nicht notwendig); bei fehlender Angabe 0
05	Größe der Sektoren (0-3): (Angabe nicht notwendig) 0 - 128 Bytes pro Sektor 1 - 256 Bytes pro Sektor (bei fehlender Angabe) 2 - 512 Bytes pro Sektor 3 - 1024 Bytes pro Sektor
06	Letzte Spurnummer beim Formatieren; Angabe nicht notwendig; sie wird dann 39 gesetzt
07	Anzahl der Sektoren pro Spur (Angabe nicht notwendig). Wird der Parameter angegeben, so richtet sich die Maximalzahl der Sektoren nach Byte 05 und zwar: 26 bei Größe 0 16 bei Größe 1 9 bei Größe 2 5 bei Größe 3
08	Nummer der Spur für Start der Formatierung (Angabe nicht notwendig; Floppy setzt dann 0)
09	Physikalischer Beginn der ersten Spur: (Angabe nicht notwendig; Floppy setzt dann 0)

10	Füllbyte; Wert, mit dem die Sektoren beim Formatieren gefüllt werden sollen; (Angabe nicht notwendig; die Floppy setzt dann $e5 als Defaultwert)
11-...	Bytes der Sektortabelle, falls in Byte 03 des Befehls das Bit 6 gesetzt war.

Busverkehr: Nach dem Senden des Befehls und der Daten vom Computer über den Kommandokanal kein Busverkehr mehr. Die Floppy sendet beim Formatieren kein Statusbyte!

Tabelle 12.4: Die Struktur des FORMAT-Kommandos im MFM-Modus

Die vielen Parameter des FORMAT-Kommandos bedürfen ein paar Erläuterungen.

Zuerst einmal zur Sektortabelle. Der Benutzer kann entweder eine eigene Sektorreihenfolge auf jeder Spur der Diskette bestimmen, oder er kann die Reihenfolge von der Floppy setzen lassen. Diese nimmt dann die Standardwerte, mit dem Beginn bei niederen Sektornummern und dem Ende bei den höchsten Sektornummern.

Die physikalische Sektorreihenfolge hängt von der Angabe des Interleave-Wertes in Byte 04 ab. Steht dieser Wert auf Null, so schreibt die Floppy nach dem Sektor 0 den Sektor 1 und so weiter. Ansonsten erfolgt die Steigerung schrittweise nach Angabe des Interleave-Wertes. Bei der Angabe einer Sektortabelle können Sie vollkommen eigene Reihenfolgen setzen. Die Floppy schreibt die Sektoren dann in der Reihenfolge auf die Diskette, in der sie durch die Tabelle bestimmt sind.

Alle Angaben beim Formatieren ab dem Byte 03 sind optional, das heißt, sie müssen nicht angegeben werden. Die Floppy setzt dann jeweils den Standardwert. Will man jedoch einen Parameter angeben, der beispielsweise in Byte 08 zu finden ist, so müssen alle anderen Parameter bis dahin ebenfalls angegeben werden.

Erklärung der einzelnen Parameter:

Der logische Startsektor beim Formatieren ist die Sektornummer, die auf einer Spur jeweils den Sektor mit der kleinsten Nummer bildet. Das ist in der Regel Sektor 0.

Die Floppy kennt insgesamt drei verschiedene Interleave-Werte. Erstens den Wert, den Sie bei der Formatierung unter MFM angeben können. Dieser Wert ist ein sogenannter Hardware-Interleave-Wert, und er bestimmt die physikalische Reihenfolge der Sektoren auf jeder Spur. Der zweite Interleave-Wert kann durch den Burst-Befehl SECTOR INTERLEAVE angegeben werden. Hier handelt es sich um einen Software-Interleave-Faktor. der beim BURST-READ und beim BURST-WRITE-Kommando zu berücksichtigen ist, wenn mehrere Sektoren hintereinander gelesen oder ge-

schrieben werden. Der dritte Interleave-Wert ist nur bei GCR-Disketten von Bedeutung. Er wird mit dem U0>S-Befehl gesetzt und beschreibt die Reihenfolge, die beim Schreiben von Dateien vom DOS für die Sektoren gewählt wird (siehe auch 6.3.1.2)

Der Wert der Sektorgröße spricht für sich. Hier wird die Datenmenge angegeben, die in einem MFM-Sektor untergebracht werden kann.

Die Nummer der letzten Spur beim Formatieren beschreibt die Spur, die als letztes formatiert werden soll. So können auf einer Diskette zum Beispiel nur die Spuren 0 bis 10 formatiert werden.

Die Anzahl der Sektoren einer Spur hängt von der Größe eines Sektors auf der Diskette ab (siehe Tabelle 12.4). Sie kann maximal den in der Tabelle dargestellten Wert erreichen oder aber auch darunter liegen. Sie müssen eine Spur also nicht vollständig mit Sektoren füllen und können theoretisch auf jeder Spur nur einen einzigen Sektor unterbringen.

Die Nummer des logischen Starttracks entspricht der des Startsektors. Hier wird entschieden, mit welcher Spurnummer die Formatierung beginnen soll. Das ist unabhängig von der physikalischen Position der entsprechenden Spur.

Das Byte 09 hingegen beschreibt die physikalische Spur, mit dem die Formatierung beginnen soll. Sie können also theoretisch mit Spur 2 beginnen (Byte 08) und diese auf Position von Spur 0 auf der Diskette (Byte 09) legen.

Das Füllbyte bedarf eigentlich keiner weiteren Erläuterung. Es beschreibt den Leerinhalt der Sektoren, also das Datenbyte, das ein Sektor enthält, wenn er noch nie beschrieben worden ist.

Die Sektortabelle muß alle Sektoren einer Diskette enthalten und wird vom DOS auf Korrektheit überprüft. Sie enthält lediglich die Sektornummern sämtlicher Sektoren der Diskette; also bei 1024-Byte Sektoren zum Beispiel 0,1,2,3,4,0,1,2,3,4,... Die Reihenfolge der Sektoren einer Spur kann dabei vollkommen unterschiedlich sein und erlaubt das Erstellen eigener, individueller Formate.

Wichtig beim Formatieren ist die Tatsache, daß kein Statusbyte von der Floppy gesendet wird, wie es bei den meisten anderen Burst-Kommandos der Fall ist. Der Computer würde sich also "aufhängen", wenn er ein solches erwarten würde.

Nun aber zum Formatieren einer GCR-Diskette. Hier sind weit weniger Parameter zu beachten und deshalb auch nicht so viele Variationen möglich, wie im MFM-Format:

Byte	Bedeutung	
00	entspricht dem Zeichen U mit dem Wert $55	
01	entspricht dem Zeichen 0 mit dem Wert $30	
02	Kommandobyte, wobei die Bits folgende Bedeutung haben:	
	Bit 0:	Drivenummer (immer 0)
	Bit 1:	immer 1
	Bit 2:	immer 1
	Bit 3:	immer 0
	Bit 4:	egal
	Bit 5:	egal
	Bit 6:	egal
	Bit 7:	Flag für Formatierung: Hier kann gewählt werden ob eine Spur vor dem Formatieren ausgemessen wird oder nicht. Das Ausmessen sorgt für eine gleichmäßige Verteilung aller Sektoren und entsprechend gleichgroßen Lücken zwischen den einzelnen Sektoren. 1 bedeutet Spur ausmessen 0 bedeutet Spur nicht ausmessen
03	Steuerbyte für Formatierung: bei GCR immer 0	
04	erstes Zeichen der ID	
05	zweites Zeichen der ID	

Busverkehr: Auch hier gilt wieder, daß die Floppy kein Burst-Statusbyte ausgibt, nachdem die Diskette formatiert wurde.

Tabelle 12.5: Struktur des FORMAT-Kommandos unter GCR-Format

Die Parameter beim GCR-Format sprechen eigentlich für sich. Wichtig ist an dieser Stelle nur noch, und das betrifft alle Formate, daß die Diskette durch das Formatieren nicht automatisch initialisiert wird. Bei Zugriffen muß also auch nach der Formatierung zuerst ein INQUIRE-DISK-Kommando folgen, bevor ein Schreib- oder Lesezugriff stattfinden kann.

12.1.5 Das SECTOR-INTERLEAVE-Kommando

Dieses Kommando behandelt den Sektorabstand beim Burst-Modus der 1570/71. Es wird dadurch festgelegt, in welchem Abstand Sektoren einer Spur geschrieben oder gelesen werden.

Das SECTOR-INTERLEAVE-Kommando hat dabei zwei Betriebsarten. Einmal das Lesen des augenblicklich eingestellten Interleave Wertes und zum anderen das Setzen eines neuen Wertes durch den Benutzer.

Byte	Bedeutung
00	entspricht dem Zeichen U mit dem Wert $55
01	entspricht dem Zeichen 0 mit dem Wert $30
02	Kommandobytej wobei die Bits folgende Bedeutung haben:
	Bit 0: Drivenummer (immer 0)
	Bit 1: immer 0
	Bit 2: immer 0
	Bit 3: immer 1
	Bit 4: immer 0
	Bit 5: egal
	Bit 6: egal
	Bit 7: Flag für Schreiben oder Lesen des Interleave:
	0= Interleave wir neu gesetzt
	1= Interleave wird von der Floppy ausgegeben
03	Interleave-Wert (nur, wenn 02/Bit 7 gleich 0)

Busverkehr: Wird Interleave neu gesetzt, so muß der Computer das Byte hinter den Kommandobytes an die Floppy senden. Die Floppy gibt daraufhin kein Byte aus. Wird der Interleave-Wert vom Computer gelesen, so sendet die Floppy nach Erhalt des Kommandos ein Byte mit dem Interleave-Wert.

Tabelle 12.6: Struktur des SECTOR-INTERLEAVE-Kommandos

12.1.6 Das QUERY-DISK-FORMAT-Kommando

Dieser U0-Befehl ist vor allem für Diagnoseprogramme interessant. Er erlaubt das Analysieren eines Diskettenformates auf einer bestimmten Spur, wobei die Spur vom Benutzer beliebig gewählt werden kann. Dieses Kommando beinhaltet das INQVIRE-DISK-Kommando und kann deshalb gleichzeitig zum Initialisieren einer Diskette verwendet werden.

Byte	Bedeutung
00	entspricht dem Zeichen U mit dem Wert $55
01	entspricht dem Zeichen 0 mit dem Wert $30
02	Kommandobytej die Bits haben folgenden Bedeutung:
	Bit 0: Drivenummer (immer 0)
	Bit 1: immer 1
	Bit 2: immer 0
	Bit 3: immer 1
	Bit 4: Diskettenseite im MFM-Format
	Bit 5: egal
	Bit 6: egal
	Bit 7: Flag für folgende Spurnummer: bei 1 folgt noch Byte 03 mit Spurnummer
03	Spurnummer (angeben, wenn 02/Bit 7 gleich 1 ist)

Busverkehr: Nach Beenden des Kommandos gibt die Floppy ein Burst-Statusbyte aus. Wurde ein MFM-Format festgestellt, so folgen zusätzlich zum Statusbyte folgende Informationen:

2. Byte:	Anzahl der Sektoren der entsprechenden Spur
3. Byte:	Spurnummer, die im Sektorheader gefunden wurde
4. Byte:	minimale Sektornummer der analysierten Spur
5. Byte:	maximale Sektornummer der analysierten Spur
6. Byte:	Hardware-Interleave, der beim Formatieren der Diskette festgelegt worden ist.

Tabelle 12.7: Die Struktur des QUERY-DISK-FORMAT-Kommandos

Wie Sie aus der Tabelle ersehen können, werden im Falle eines identifizierten MFM-Formats eine ganze Menge an Informationen an den Computer ausgegeben, die eine genaue Analyse eines gefundenen Diskettenformats möglich machen. Bei GCR-formatierten Disketten erfolgt außer dem Statusbyte keine Rückmeldung, da das GCR-Format immer gleich ist.

12.1.7 Das INQUIRE-STATUS-Kommando

Mit diesem Kommando ist es möglich, die Ausgabe des Burst-Statusbytes der Floppy zu erzwingen oder aber dieses Statusbyte nach eigenem Belieben neu zu setzen.

Byte	Bedeutung	
00	entspricht dem Zeichen U mit dem Wert $55	
01	entspricht dem Zeichen 0 mit dem Wert $30	
02	Kommandobyte; wobei die Bits folgende Bedeutung haben:	
	Bit 0:	Drivenummer (immer 0)
	Bit 1:	immer 0
	Bit 2:	immer 1
	Bit 3:	immer 1
	Bit 4:	immer 0
	Bit 5:	egal
	Bit 7,6:	00 - Burst-Statusbyte schreiben (Byte 03)
		01 - Burst-Status schreiben und Diskettenwechsel-Flag löschen
		10 - Burst-Statusbyte lesen
		11 - Burst-Status lesen und Diskettenwechsel darin anzeigen
03	Neues Statusbyte, wenn 02/Bit 7 gleich 0 war	

Busverkehr: Die Floppy erwartet vom Computer entweder ein Statusbyte, oder Sie gibt den aktuellen Status nach Erhalt des Kommandos aus.

Tabelle 12.8: Die Struktur des INQUIRE-STATUS- Kommandos

Zum Kommandobyte dieses Befehls eine Erläuterung. Die Bits 6 und 7 bestimmen den Modus des Befehls. Ist Bit 7 gleich 0, so erwartet die Floppy vom Computer ein neues Statusbyte; ansonsten gibt sie den aktuellen Status aus. Der Benutzer kann dabei mit Bit 6 wählen, inwiefern er einen möglicherweise soeben stattgefundenen Diskettenwechsel berücksichtigen will.

Die Floppy stellt einen Diskettenwechsel durch die Änderung des Zustands an der Schreibschutz-Lichtschranke fest. Ist eine solche Änderung erfolgt,

so wird intern das Flag für Diskettenwechsel gesetzt. Wenn nun beim INQUIRE-STATUS-Kommando das Bit 6 gesetzt ist, so wird beim Lesen des Statusbytes angezeigt, ob ein Diskettenwechsel stattgefunden hat (der Status lautet dann: disk id mismatch or disk change), oder es wird beim Schreiben des Status das Flag für den Diskettenwechsel gelöscht. Die Floppy "vergißt" dann einen eben erfolgten Diskettenwechsel, der noch nicht durch INQUIRE-DISK aufgehoben worden war.

12.1.8 Das FASTLOAD-Kommando

Dieses Kommando wird vom C128 wohl am meisten benötigt, wenn Sie mit dem Computer arbeiten. Bei jedem Laden eines Programms wird dieser Befehl nämlich vom Computer in der Floppy gestartet, wodurch ein sehr schnelles Laden von Programmen gegenüber der 1541 mit einem C64 oder der 1570/71 mit einem C64 erfolgt.

Byte	Bedeutung
00	entspricht dem Zeichen U mit dem Wert $55
01	entspricht dem Zeichen 0 mit dem Wert $30
02	Kommandobytej wobei die Bits folgende Bedeutung haben:
	Bit 0: immer 1
	Bit 1: immer 1
	Bit 2: immer 1
	Bit 3: immer 1
	Bit 4: immer 1
	Bit 5: egal
	Bit 6: egal
	Bit 7: Flag für Programmdatei; ist dieses Bit gesetzt, so muß die angegebene Datei keine Programmdatei sein; es kann auch eine sequentielle Datei sein. Ist das Bit=0, so wird nur nach einer Programmdatei mit dem gegebenen Namen gesucht!
03-...	Name der gesuchten Datei

Busverkehr: Nach Erhalt des Befehls gibt die Floppy jeden Sektor der Datei mit dem Burst-Statusbyte davor aus.

Tabelle 12.9: Die Struktur des FASTLOAD-Kommandos

Das FASTLOAD-Kommando kann natürlich, wie alle anderen Burst-Kommandos auch, nur mit den entsprechenden Burst-Routinen im Computer funktionieren. Diese Burst-Routinen für den Computer werden wir gleich behandeln, nachdem wir zuvor die restlichen U0-Befehle etwas genauer betrachtet haben.

12.1.9 Die UTILITY-Kommandos

Die nun folgenden U0-Kommandos benötigen keine Burst-Routinen für die Datenübertragung, da es sich hierbei um Kommandos für die Einstel-

lung der verschiedenen Arbeitsmodi der 1570/71 handelt. Diese Einstellung kann sowohl von einem C64 als auch von einem Cl28 aus in gleicher Weise erfolgen. Wie schon vorher, so werden auch hier die Befehle über den Kommandokanal an die Floppy geschickt.

12.1.9.1 Das Setzen des GCR-Interleave-Faktors (U0>S)

Dieser Befehl verändert die voreingestellten Interleave-Werte für die Dateibearbeitung im GCR-Format. Der Wert steht normalerweise auf 6 und kann vom Benutzer beliebig geändert werden. Dabei wird einfach der Befehl U0>S mit dem folgenden Interleave-Wert an die Floppy geschickt. Dazu zwei Beispiele:

```
OPEN 1,8,15, "U0>S"+CHR$(6)
PRINT#15,"U0>S";CHR$(10)
```

12.1.9.2 Das Setzen der Versuche bei Lesefehlern (U0>R)

Wenn bei einem Lesevorgang Fehler auftreten, so versucht das DOS in der Regel mehrere Male, den Lesevorgang doch durchzuführen. Die Anzahl der Versuche beträgt dabei im Normalfall 5. Wünscht der Benutzer jedoch eine Änderung dieses Wertes, so kann das durch den Befehl U0>R geschehen. Hier wird diese Anzahl auf einen, vom Benutzer wählbaren, Wert neu gesetzt. Dazu zwei Beispiele:

```
OPEN 1,8,15, "U0>R"+CHR$(5)
PRINT#15,"U0>R";CHR$(9)
```

12.1.9.3 Der ROM-Test der 1570/71 (U0>T)

Mit dem folgenden Befehl werden Sie in die Lage versetzt, jederzeit auf eine korrekte ROM-Prüfsumme zu testen, wie Sie in $8000/8001 abgespeichert ist. Bei der Ausführung dieses Befehls ist darauf zu achten, dass er durch die langwierige Berechnung der Prüfsumme geraume Zeit benötigt (ca. 10 Sekunden). Die Floppy ist daher während dieser Zeit nicht ansprechbar und scheint "abgestürzt". Bei Programmen, die diesen Befehl verwenden, kann es deshalb nie schaden, ein "Einen Augenblick Geduld. bitte!" einzufügen, um den Anwender nicht im unklaren über die Vorgänge zu lassen. Der Befehl lautet U0> T und wird ohne Parameter direkt über den Kommandokanal zur Floppy geschickt.

12.1.9.4 Umschalten der Betriebsart (U0>M)

Dieser Befehl dient dazu, die 1570/71 entweder in den 1541-Modus oder in den schnelleren 1570/71-Modus zu schalten. Im 1541-Modus läuft die Floppy nur mehr mit einer Taktfrequenz von 1 MHz (sonst 2 MHz) und simuliert eine 1541 sehr originalgetreu (sogar das Formatieren dauert wieder 90 Sekunden). Die USER-0-Befehle (außer das Umschalten des Betriebsmodus und die Kopfanwahl, siehe 12.1.9.5) funktionieren dann nicht mehr und der schnelle Busbetrieb ist ebenfalls nicht mehr möglich. Dieser Modus ist beim Einschalten voreingestellt und muß erst durch den C128 auf die 1570/71-Betriebsart umgeschaltet werden.

Die Syntax des Befehls lautet U0>M0 für das Umschalten auf 1541-Betrieb und U0>M1 für 1570/71-Betrieb. Der 1570/71-Betriebsmodus kann übrigens auch beim Betrieb der Floppy an einem C64 verwendet werden. Man kommt dann in den Genuß des schnelleren Formatierens und bei der 1571 zusätzlich in den Genuß der doppelten Diskettenkapazität.

12.1.9.5 Das Anwählen eines Schreib-/Lesekopfes (U0>H)

Dieser Befehl ist nur auf der 1571 verfügbar und erlaubt dort das Umschalten zwischen beiden Diskettenseiten, wenn sich die Floppy im 1541-Modus befindet. Die Syntax lautet dann U0>H0 für den Kopf auf der Diskettenseite 0 und U0>H1 für den Kopf auf der Diskettenseite 1.

Achtung! Eine Diskette, die auf einer 1541 durch Umdrehen auf beiden Seiten beschrieben wurde, kann auf diese Art natürlich nicht vollständig gelesen werden, da die zweite Seite durch das Umdrehen der Diskette in verkehrter Richtung beschrieben wurde und deshalb vom Kopf 1 der 1571 nicht mehr gelesen werden kann. Das Umschalten nützt also nur dann etwas, wenn die Diskette generell auf der 1571 im 1541-Modus beidseitig beschrieben wurde.

12.1.9.6 Das Einstellen der Gerätenummer (U0>geräte#)

In Kapitel 1.3.7 haben Sie schon erfahren, wie Sie die Gerätenummer Ihrer Floppystation hardwaremäßig ändern können; nämlich durch Umlegen der entsprechenden DIP-Schalter auf der Platine. Soll das Umschalten jedoch nur kurzfristig erfolgen, um beispielsweise zwei Laufwerke zum Kopieren zusammenzuschließen, so kann des Ändern der Gerätenummer auch per Software mit dem Befehl U0>geräte# erfolgen. Dieser Befehl wird behandelt wie alle vorangegangenen UTILITY-Kommandos auch, wobei für geräte# eine Gerätenummer von 4 bis 30 eingestellt werden kann. Wie Sie sehen, können Sie auf diese Art sogar Gerätenummern einstellen, die über die DIP-Schalter nicht möglich sind.

12.2 Das Burst-Statusbyte der 1570/71

Bei fast allen Burst-Kommandos der 1570/71 wird nach Beendigung des Kommandos von der Floppy ein Statusbyte an den Computer übergeben. Dieses Statusbyte ersetzt bei den USER-0-Befehlen die Klartext-Fehlermeldung, wie Sie über den Kommandokanal kommt und wird in der Regel vor der Übertragung eines jeden Sektors oder nach dem Schreiben eines Sektors an den Computer gesendet.

Arbeiten Sie also im Burst-Modus, das heißt mit schnellen Busroutinen und den U0-Kommandos, so sollten Sie das Statusbyte der Floppy berücksichtigen. Die Klartext-Fehlermeldung ist in der Regel zweitrangig und gibt nicht den Status wieder, der während einen Burst-Kommandos auftritt, sondern den Status, der dem Kommandokanal bei Ende des Kommandos zur Verfügung gestellt wird.

Die folgende Tabelle zeigt die Bedeutung der einzelnen Bits im Statusbyte des Diskcontrollers:

Bit	Bedeutung
0-3	Controller-Status der Floppystation nach einem Diskettenzugriff. Der Status hat dabei drei verschiedene Meldungsarten, je nach Befehl und Diskettenformat. Statusbelegung bei einer Diskette im GCR-Format: Bits 3,2,1,0: 0000: ok 0001: ok 0010: sector not found 0011: no sync 0100: data block not found 0101: data block checksum error 0110: format error 0111: verify error 1000: write protect error 1001: header block checksum error 1010: data extends into next block 1011: disk id mismatch/disk change 1100: n.v. 1101: n.v. 1110: syntax error 1111: no drive present Statusbelegung bei einer Diskette im MFM-Format: Bits 3,2,1,0: 0000: ok 0001: ok 0010: sector not found 0011: no address mark 0100: n.v.

	0101: data crc error
	0110: format error
	0111: verify error
	1000: write protect error
	1001: header block checksum error
	1010: n.v.
	1011: disk change
	1100: n.v.
	1101: n.v.
	1110: syntax error
	1111: no drive present
	Statusbelegung bei FASTLOAD-Kommando (gesamtes Byte):
	Bits 7,6,5,4,3,2,1,0:
	00000000 ok
	00000001 ok
	00000010 file not found
	00011111 EOI
4-5	Sektorgröße:
	Bits 5,4:
	00: 128 Bytes Sektoren
	01: 256 Bytes Sektoren
	10: 612 Bytes Sektoren
	11: 1024 Bytes Sektoren
6	Drivenummer; bei der 1570/71 immer 0
7	Format-Modus: 0=GCR; 1=MFM
	Dieses Bit bestimmt die Funktionen der restlichen Bits des Statusbytes, wobei
	im GCR-Modus die Sektorgröße immer auf 256-Byte-Sektoren stehen muß!

Tabelle 12.10: Das Burst-Statusbyte und dessen Belegung

Wie Sie sehen, erhalten Sie mit dem Status byte des Controllers immer einen recht umfassenden Bericht über den Modus, in dem sich die Floppy gerade befindet.

12.3 Die Burst-Routinen für den Computer

Im folgenden Abschnitt wollen wir wieder einmal praktisch arbeiten. In diesem Kapitel sind Burst-Routinen für den Computer abgedruckt, die originalgetreu im Diskmonitor des nächsten Kapitels vorhanden sind. Sie stellen Standardlösungen für Burst-Datenübertragungs-Routinen dar und sind deshalb vollständig dokumentiert worden.

Zuerst wollen wir das betreffende Burst-Kommando an die Floppy überhaupt erst einmal übertragen. Diese Übertragung erfolgt in Maschinensprache, da der Computer direkt im Anschluß daran in den Burst-Modus umschalten muß, um keines der sofort folgenden Bytes der Floppystation zu verpassen. Die folgende Routine ist, wie alle anderen, aus dem BURSTMON entnommen:

Der Kommandokanal muß mit OPEN 1,8,15 schon geöffnet worden sein. Das Y-Register enthält die Länge des Befehlsstrings:

```
        lda $0a1c     ; Flags für Diskbetrieb laden
        and #$bf      ; Flag für schnellen Busbetrieb löschen
        sta $0a1c     ; und wieder abspeichern
        ldx #$01      ; Dateinummer (in diesem Fall 1)
        jsr $ffc9     ; Ausgabegerät setzen
        ldx #$00      ; Index-Startwert laden; der Monitor legt den
loop    lda $1430,x   ; Befehlsstring ab $1430 ab
        jsr $ffd2     ; Zeichen an die Floppy ausgeben
        inx           ; Zeiger erhöhen
        dey           ; Anzahl der Bytes minus 1
        bne loop      ; weiter, wenn noch Zeichen auszugeben sind
        jsr $ffcc     ; Ausgabe zur Floppy zurücksetzen
        bit $0a1c     ; Flag für schnellen Busmodus testen
        bvc error     ; Ende, wenn Floppy nicht im Fast-Modus
        ...           ; ...
```

Jetzt ist der Befehlsstring also zur Floppy geschickt worden und diese beginnt nach dem Kommando für "Ausgabe zurücksetzen" sofort mit der Ausführung des Befehls. Die nun folgende Routine dient dem Einlesen entweder nur eines Statusbytes oder eines gesamten Sektors, wobei die Größe des Sektors in 256-Byte-Blöcken in $b0 und die Größe eines Blocks ($80 oder $00) in $bl abgelegt sein muß. Diese Angaben sind notwendig. da ein gelesener Sektor je nach Format eine unterschiedliche Anzahl von Bytes aufweisen kann. Die Adresse, ab der ein eventuell eingelesener Sektor abgelegt wird, muß in den Speicherstellen $fb und $fc (L/H) stehen. Wichtig ist ferner, daß beim Cl28 der I/O-Bereich zwischen $d000 und $dfff eingeschaltet ist (etwa durch Beschreiben des Konfigurations-Registers $ff00 mit 0).

```
        sei             ; Interrupts unterbinden
        bit $dc0d       ; Kontrollregister löschen
        ldx $bO         ; Anzahl der Blöcke holen
        lda $dd00       ; Register für seriellen Bus lesen
        eor #$10        ; CLOCK-Leitung invertieren
        sta $dd00       ; und setzen
        lda #$08        ; Flag für Schieberegister eingelesen
l1      bit $dc0d       ; testen; Byte erhalten?
        beq l1          ; warten, bis Byte eingelesen
        lda $dd00       ; Register für seriellen Bus lesen
        eor #$10        ; CLOCK-Leitung invertieren
        sta $dd00       ; und wieder setzen
        lda $dc0c       ; Byte aus Schieberegister holen
        sta $fa         ; und als Statusbyte merken
        and #$0f        ; Statusbits isolieren
        cmp #$02        ; Fehlermeldung?
        bcs error       ; verzweige, wenn ja
        ldy #$00        ; Index setzen
loop    lda #$08        ; Flag für Schieberegister eingelesen
l2      bit $dc0d       ; testen; Byte erhalten?
        beq l2          ; warten, bis Byte eingelesen
        lda $dd00       ; Register für seriellen Bus lesen
        eor #$10        ; CLOCK-Leitung invertieren
        sta $dd00       ; und setzen
        lda $dc0c       ; Byte aus Schieberegister holen
        sta ($fb),y     ; und in Puffer schreiben
        iny             ; nächstes Byte
        cpy $b1         ; schon ein Block gelesen?
        bne loop        ; weitermachen, wenn nein
        inc $fc         ; Pufferadresse Hi erhöhen
        dex             ; Anzahl der zu lesenden Blöcke vermindern
        bne loop        ; und weiterlesen, bis alles übertragen
        clc             ; Flag für Fehler löschen
        .byte $24       ; nächsten Befehl überspringen
error   sec             ; Flag für Fehler setzen
        cli             ; Interrupts wieder zulassen
        rts             ; Ende
```

Tabelle 12.11: Routine für das BURST-READ-Kommando

Wie Sie aus dieser Routine sehen können, ist die Übertragung im schnellen Busmodus von der Floppy zum Computer sehr einfach und aus diesem Grund auch sehr schnell. Der Benutzer muß nur noch warten, bis die Übertragung beendet ist.

Das Holen eines einzelnen Statusbytes ist in dieser Routine natürlich schon enthalten. In diesem Fall muß einfach der Rest für das Lesen des Sektors übersprungen werden oder entfallen.

Wie das Lesen eines Sektors, so ist auch das Schreiben eines Sektors sehr einfach. Die folgende Routine ist dafür zuständig, wobei ein zusätzliches Flag, nämlich $fd, benötigt wird. Hier wird der jeweils gültige Zustand der

CLOCK-Leitung für die nächste Übertragung abgelegt. Die restlichen Parameter entsprechen denen von BURST -READ:

```
        sei             ; Interrupts unterbinden
        lda #$40        ; Flag für CLOCK-Leitung
        sta $fd         ; setzen
        ldx $bO         ; Anzahl der zu schreibenden Blöcke
        ldy #$00        ; Index setzen
        lda $d505       ; MMU Modus-Konfigurationsregister lesen
        ora #$08        ; seriellen Bus auf Ausgabe schalten
        sta $d505       ; und neuen Modus setzen
        lda #$7f        ; serielles Schieberegister
        sta $dc0d       ; auf Ausgabe schalten und Flags löschen
        lda #$00        ; Wert für Timer Hi
        sta $dc05       ; setzen
        lda #$03        ; Wert für Timer Lo; Übertragunsgsrate 6 us
        sta $dc04       ; setzen; schnellst mögliche Übertragungsrate
        lda $dc0e       ; Kontrollregister für Timer lesen
        and #$80        ; Schieberegister auf Ausgang und
        ora #$55        ; Timer auf "free running mode"
        sta $dc0e       ; setzen
        bit $dc0d       ; Flags für Schieberegister löschen
loop    lda $dd00       ; seriellen Bus lesen
        cmp $dd00       ; konstanter Wert?
        bne loop        ; nein, konstanten Wert abwarten
        eor $fd         ; ClOCK-Bit invertieren
        and #$40        ; und prüfen
        beq loop        ; warten, bis Floppy Daten anfordert
        lda $fd         ; CLOCK-Bit
        eor #$40        ; invertieren für nächsten Zustand
        sta $fd         ; und als Flag wieder setzen
        lda ($fb),y     ; Byte aus Puffer holen
        sta $dc0c       ; und ins Schieberegister schreiben
        lda #$08        ; Bit für Schieberegister
l2      bit $dc0d       ; testen; Byte schon ausgegeben?
        beq l2          ; warten, bis Byte ausgegeben
        iny             ; Index auf nächstes Zeichen
        cpy $b1         ; schon ein Block ausgegeben?
        bne loop        ; weitermachen, wenn nein
        inc $fc         ; Pufferadresse Hi erhöhen
        dex             ; Anzahl der auszugebenden Blöcke vermindern
        bne loop        ; weitermachen, bis alles ausgegeben
        lda #$08        ; Flag für Timer nach Herunterzählen anhalten
        sta $dc0e       ; setzen
        lda $d505       ; Modus-Konfigurationsregister der MMU
        and #$f7        ; seriellen Bus auf Eingang
        sta $d505       ; neuen Modus setzen
        bit $dc0d       ; Flag für Schieberegister löschen
        lda $dd00       ; seriellen Bus lesen
        eor #$10        ; CLOCK-Leitung umschalten
        sta $dd00       ; und wieder setzen
        lda #$08        ; Flag für Schieberegister
l3      bit $dc0d       ; testen; Byte eingelesen?
        beq l3          ; warten, bis Byte eingelesen
        lda $dc0c       ; Byte aus Schieberegister holen
        sta $fa         ; als Statusbyte merken
        lda $dd00       ; seriellen Bus lesen
```

```
        and #$ef        ; CLOCK-Leitung löschen
        sta $dd00       ; und setzen
        lda $fa         ; Statusbyte zurückholen
        and #$0f        ; Statusbits isolieren
        cmp #$02        ; Fehler aufgetreten?
        cli             ; Interrupts wieder zulassen
        bcs error       ; verzweige zur Fehlerroutine bei Fehler
        rts             ; sonst Ende; alles ok
error   ...             ; Fehlerbehandlung...
```

Tabelle 12.12: Routine für das BURST-WRITE-Kommando

Wie Sie aus dem Programm sehen können, werden die Bits mit einer Geschwindigkeit von I Bit pro 6 Mikrosekunden übertragen. Das entspricht einer Übertragungsrate von weit über 20 000 Bytes pro Sekunde bei einer Taktfrequenz von I MHz. Schalten Sie jetzt auch noch den Cl28 in die 2MHz-Betriebsart (Befehl: FAST), so ergeben sich für eine serielle Schnittstelle wahrhaft enorme Übertragungsraten, die sogar das Senden einer gesamten Spur der Diskette in einer Diskettenumdrehung zulassen.

Die oben angegebenen Routinen dürften wohl für alle Anwendungszwecke ausreichend sein. Es steht Ihnen nun nichts mehr im Weg, Teile dieser Routinen in eigenen Programmen zu verwenden und dadurch in den Genuß der schnellen Übertragung zu kommen. Wie wäre es denn zum Beispiel mit Programmen wie schnelles Abspeichern auf Diskette, schnelles Kopieren von Disketten, Spuranalyse einer eingelegten Diskette, ...

12.4 Burst-Routinen auch für den C64

Die C 64-Besitzer unter Ihnen werden vielleicht teilweise mit neidischen Gedanken an die glücklichen C128-Anwender gedacht haben, die in den Genuß der schnellen Busroutinen kommen.

Doch halt, die schnellen Busroutinen werden durch einen Interface-Baustein vom Typ CIA 6526 ermöglicht, und dieser befindet sich in zweifacher Ausführung auch im C64. Wie Sie die Floppy jederzeit, auch bei einem angeschlossenen C64 in den 1570/71-Betriebsmodus versetzen können, das haben Sie bereits in Kapitel 12.1.9.4 erfahren. Nun werden Sie erfahren, welche Möglichkeiten noch im C64 mit der 1570/71 stecken.

Der einzige Unterschied im seriellen Bus des C128 zum C64 ist die Verkabelung von zwei Leitungen. Beim C128 ist die SRQ-Leitung zusätzlich mit dem CNT-Pin des CIA 1 im Computer und die DATA-Leitung zusätzlich mit dem SP-Pin des CIA 1 verbunden. Das CNT-Pin entspricht dabei einer Taktleitung, die die Geschwindigkeit der Übertragung des Schieberegisters regelt (siehe auch Anhang B: CIA 6526). Das SP-Pin entspricht dem Ausgang des seriellen Schieberegisters.

Wenn Sie sich die Belegung des USER-Ports des C64 einmal genauer betrachten, so werden Sie sicherlich bei den Pins 4 bis 7 stutzig werden. Diese Pins tragen die Namen CNT1, SP1, CNT2 und SP2. Es handelt sich hierbei um die schon beschriebenen Pins der beiden CIA 6526 und zwar jeweils die Takt- und die Übertragungsleitung für das serielle Schieberegister. In unserem Fall brauchen Sie also nur die beiden Pins mit den entsprechenden Leitungen des seriellen Busses zu verbinden. Wollen Sie den gleichen CIA 6526 wie im C128 verwenden, auf dem auch die oben beschriebenen Burst-Routinen laufen, so müssen Sie den Pin 4 (CNT 1) vom USER-Port mit der SRQ-(Service Request)-Leitung des seriellen Bus und Pin 5 (SP 1) mit dessen DATA-Leitung verbinden. Sie erhalten so einen originalen C128-Bus.

Die oben beschriebenen Burst-Routinen laufen dann fast ohne Änderung auch auf dem C64. Lediglich die Behandlung des MMU-Modusregisters und die Flagabfrage, ob sich die angeschlossene Floppy im FAST-Modus befindet, sind zu entfernen, da der C64 keine Betriebssystemroutinen zur Feststellung des Floppymodus besitzt.

13 Der BURSTMON 3.1

Dieses Kapitel ist kein "Lehrkapitel" wie die vorangegangenen. Es stellt vielmehr eine Bedienungsanleitung dar. Unter Listing 13.1 finden Sie nämlich einen teils in BASIC und teils in Maschinensprache geschriebenen Diskmonitor für den Commodore 128, der Ihnen ein äußerst leistungsfähiges Werkzeug in die Hand gibt. Durch die optimale Mischung aus BASIC und Maschinensprache ist dieser Monitor überraschend schnell in der Bearbeitung sämtlicher Funktionen und erlaubt eine komfortable Bedienung aller Diskettenfunktionen der 1570/71.

Wie Sie aus dem Namen des Monitors schon ersehen können, handelt es sich hierbei um ein Programm, das ausschließlich mit den USER-O-Befehlen arbeitet. Das hat den Vorteil, daß es keine Rolle spielt, welches Format die eingelegte Diskette hat. Der Monitor bearbeitet MFM-Disketten genausogut wie GCR-Disketten.

Da der Monitor über eine sehr komfortable Bildschirmmaske verfügt, wurde er für einen 80-Zeichen-Bildschirm des C128 konzipiert. Er erlaubt so, dank der Fenstertechnik des Commodore 128, eine vielseitige Parameteranzeige, so daß der Benutzer immer alles im Blickfeld hat.

Wenn Sie den Monitor abgetippt und auf eine Diskette gespeichert haben, dann starten Sie bitte mit RUN.

Es wird nun die Bildschirmmaske des Monitors aufgebaut, die während der gesamten Bearbeitung immer erhalten bleibt. dabei können Sie drei verschiedene Fenster erkennen:

oben	Fenster mit der Anzeige des Controller- Burst-Status
mitte	Arbeitsfenster (jetzt leer)
unten	Kommandofenster für die Befehlseingabe.

Im oberen Fenster können Sie schon einmal die Maske des Controller-Status erkennen. Es erfolgt hier immer eine Klartext-Auswertung des Statusbytes im Burst-Modus der Floppystation. Außerdem werden noch mehrere zusätzliche Parameter angezeigt. Dabei bedeuten:

Track	Spurnummer, die gerade bearbeitet wird
Sector	Sektornummer, die gerade bearbeitet wird
Format	Diskettenformat der letzten bearbeiteten Diskette
Sector width	Größe der Sektoren auf der aktuellen Spur
Capacity	mögliche Anzahl der Sektoren der aktuellen Spur

Generell gilt für BURSTMON 3.1: Wenn irgend wo ein Fehler aufgetreten ist, so wechselt die Schriftfarbe der Fehlermeldung auf Rot. Bei einer Bemerkung, die die Eingabe einer Anweisung oder eines Kommandos erwartet, ist die Schriftfarbe Gelb. Ansonsten ist die Schriftfarbe Cyan eingestellt.

Nun gleich zur Anzeige des unteren Fensters auf dem Bildschirm. Hier wird entweder ein Kommando eingegeben, auf einen Tastendruck gewartet oder die Statusmeldung der Floppy (nicht der Burst-Status) ausgegeben. Es erfolgen auch sämtliche Parametereingaben über dieses Fenster.

Im Augenblick wartet der Monitor auf die Eingabe eines Kommandos. Bis auf das Kommando "x" für "exit", das Verlassen des Monitors, sind alle Kommandos oder Kommandokomplexe auf Funktionstasten gelegt. Sie können aber auch direkt eingegeben werden, wobei der Anfangsbuchstabe genügt.

Die Kommandos in der Reihenfolge der Funktionstasten:

fI	BURST – READ; Lesen eines Sektors von der Diskette
f2	BURST - WRITE; Schreiben eines Sektors auf eine Diskette
f3	DIRECTORY; Anzeige des Directory für das angewählte Laufwerk
f4	DISK-ST A TUS anzeigen
f5	INQUIRE DISK; Initialisieren einer Diskette
f6	QUERY DISK; Analysieren des Diskformats auf einer Spur
f7	USER -0- Befehlskomplex
f8	EDIT; Aufrufen des Editors

BURST-READ und BURST-WRITE bedürfen keiner weiteren Erläuterung. Diese Kommandos erwarten noch die Angabe von Spur- und Sektornummer sowie Diskettenseite und lesen, beziehungsweise schreiben danach den gewünschten Sektor. Nach dem Lesen eines Sektors wird automatisch der EDIT - Modus aufgerufen!

Alle Kommandos, die entweder den Burst-Status im oberen oder den Diskstatus im unteren Fenster anzeigen, erwarten vom Benutzer einen Tastendruck als Bestätigung, daß nun fortgefahren werden kann.

Der DIRECTOR Y - Befehl spricht ebenfalls für sich. Hier wird das Directory von der Floppystation angezeigt, deren Gerätenummer gerade angewählt ist (zur Anwahl der Gerätenummer gleich mehr).

DISK -STATUS zeigt den aktuellen Floppystatus, nicht den Burst-Status im unteren Fenster an. Diese Funktion wird auch durch Eintippen des @-Zeichens aufgerufen, wobei dieses Zeichen auch noch weitere Funktionen hat.

Das INQUIRE-DISK-Kommando initialisiert eine Diskette im Sinne von Kapitel 12.1.3. Dieses Kommando *muß* nach jedem Diskettenwechsel gegeben werden. Es wurde bewußt darauf verzichtet, dem Monitor irgendwelche automatischen Einrichtungen mitzugeben, da diese die Wirkung der einzelnen Befehle unter Umständen verschleiern. Bleibt das nötige INQUIRE-DISK-Kommando aus, so kann es passieren, das sich die Floppy oder der Computer bei einem Wechsel des Diskettenformates "aufhängen". Hier hilft dann nur noch die RUNjSTOP-RESTORE-Funktion beziehungsweise bei der Floppy das Aus- und wieder Einschalten.

Das QUERY-DISK-FORMAT-Kommando kann als Ersatz für INQUIREDISK verwendet werden. Hier wird eine gewünschte Spur einer fremden Diskette analysiert. Handelt es sich dabei um eine MFM-Diskette, so werden im Arbeitsfenster des BURSTMON 3.1 zusätzliche Informationen über das Diskettenformat angezeigt.

Im USER-O-Komplex wird ein Menü aufg_iufen, das es dem Benutzer gestattet, unter sämtlichen U0-Befehlen der Floppy auszuwählen und diese zu bearbeiten. Das erlaubt vor allem dem Einsteiger in Sachen Burst ein b_quemes Ausprobieren aller Funktionen, wobei die Parameter der einzelnen Befehle jeweils nach Wahl des entsprechenden Kommandos anzugeben sind. Wird mit der Taste 9 der UTILITY - Befehl aufgerufen, so erscheint ein weiteres Menü, das es dem Benutzer wiederum gestattet, unter den verschiedenen UTILITY - Befehlen auszuwählen.

Über den Sinn und Zweck der einzelnen USER-O-Befehle werden Sie in Kapitel 12 ausführlich unterrichtet.

Der EDIT -Befehl. Hier wird im Arbeitsfenster auf einen Blockcursor umgeschaltet, wobei nun ein jeweils gewünschter Sektor editiert werden kann. Das Arbeitsfenster zeigt dabei jeweils 256 Bytes an und wird bei größeren Sektoren und entsprechender Cursorbewegung nach oben oder unten gerollt. Sie können so jedes Byte eines Sektors gezielt anwählen, indem Sie den Cursor darauf setzen. Eine einfache Änderung des Wertes wird sofort in den Speicher übertragen. Das Drücken der RETURN-Taste ist nicht notwendig. Der Monitor verhindert das Abändern fester Werte oder das Eingeben unerlaubter Zeichen, so daß Bedienungsfehler weitgehend ausgeschlossen werden.

Wenn Sie den Cursor mit den Steuertasten nach unten bewegen, so erfolgt bei größeren Sektoren das schon erwähnte Aufrollen des Bildschirms bis zum Ende des jeweils eingelesenen Sektors. Danach rollt der Cursor über den unteren Bildschirmrand hinaus und erscheint im oberen Bildrand erneut. Lassen Sie den Cursor nach oben wandern, so wird der gesamte Bild

schirm beim Erreichen des oberen Bildschirmrandes nach unten gerollt, bis wieder der Anfang des Sektors erreicht worden ist.

Wie groß der editierbare Ausschnitt ist, wird jeweils durch die Größe der Sektoren auf der eingelegten und initialisierten Diskette bestimmt.

Mit Drücken der Taste "+-" gelangen Sie wieder in den Kommandomodus zurück, wo Sie den abgeänderten Sektor dann zum Beispiel auf eine Diskette zurückschreiben können. Es ist auch möglich, einen GCR-Sektor von 265-Bytes Größe auf eine MFM-Diskette zu schreiben. Ist der zu schreibende Sektor kleiner als der, in den er geschrieben wird, so wird der Editierausschnitt auf die Größe des Sektors der eingelegten Diskette eingestellt und der Inhalt dann dementsprechend geschrieben.

So viel zu den grundsätzlichen Befehlen des BURSTMON 3.1. Zusätzlich existieren noch drei verschiedene Möglichkeiten im Zusammenhang mit dem @-Zeichen.

Die Eingabe von @ ohne einen Zusatz zeigt, wie schon erwähnt, den aktuellen Diskstatus an.

Bei Eingabe von @ und der Nummer 8 oder 9 dahinter, kann der Monitor auf die entsprechende Geräteadresse umgestellt werden. Alle weiteren Kommandos beziehen sich dann auf die neu angewählte Gerätenummer. Bei @9 wird also zum Beispiel auf eine Floppy mit der Geräteadresse 9 umgeschaltet.

Mit Hilfe des @-Zeichens kann schließlich auch noch ein System befehl an die Floppy gesendet werden. Hier wird der Befehl einfach hinter das @Zeichen geschrieben. So zum Beispiel "@n:neue diskette,nd", wobei die Diskette im aktuellen Laufwerk neu formatiert wird.

Wichtig für das Arbeiten mit BURSTMON 3.1 ist: Wenn Sie einen Systembefehl an die Floppy schicken oder sich das Directory ansehen, so ist die Diskstatusmeldung im unteren Bildschirmfenster von Bedeutung. Arbeiten Sie hingegen im Burst-Modus, so müssen Sie den Controllerstatus im oberen Fenster beachten.

Bei Fehleingaben ertönt ein Klingelzeichen und die Eingabe wird erneut gefordert. Wollen Sie ein Laufwerk ansprechen, das nicht vorhanden ist, so kehrt der BURSTMON 3.1 mit einem Klingelzeichen in den KommandoStatus zurück.

Listing 13.1: Der Diskmonitor BURSTMON 3.1

```
100 rem burstmon 3.1 (c) 1986 by karsten schramm
101 fast:poke2603,64:printchr$(27)"u":bank 15:poke251,0:poke252,11:gosub371
102 clr:print"{clr}{home}{home}{cyn}";chr$(14);:poke54784,26:poke54785,2
103 t$="00":s$="00":poke dec("1700"),0
104 f$="GCR":w$="0256":k$="21"
105 cs$="ok":dd$=ds$:dd=ds
106 bs$="":u=8:d$="0":n$="0":gosub107:goto109
107 restore 109:for x=1to8:readx$:key x,x$+chr$(13):next:return
108 forx=1to8:key x,"":next:return
109 data "read","write","$","@","inquire disk","query disk","user 0","edit"
110 cs=0:gosub308:trap365:goto125:trap365
111 rem unterroutinen
112 window 2,2,77,3,1:ifcs$="ok"then printchr$(14);"{gry3}{home} Track:"t$"  Sector:"s$"  Format: "f$"  Sector
width: "w$" Bytes  Kapacity: "k$:else print"{lred}";
113 print " Controller Status: ";cs$;"  on Side ";d$;"  Unit#";u"{cyn}":return
114 window 2,2,77,3,1:printchr$(14);"{gry3}{home} Track:"t$"  Sector:"s$"  Format: "f$"  Sector width: "w$"
Bytes  Kapacity: "k$
115 print "{gry3} Controller Status: ";cs$;"  on Side ";d$;"  Unit#";u"{cyn}":return
116 window 2,5,77,20,0:return
117 window 2,5,77,20,1:return
118 rem menue einblenden
119 gosub116:printchr$(14);"{clr}{blk}*** BURSTMON v1.0  Table Of Commands ***"
120 window 2,23,77,23,1:printchr$(14);:input"{yel}Command: ";c$:print"{cyn}":return
121 window 2,23,77,23,1:printchr$(14);:input"{yel}Track  1{left}{left}{left}";t$:print"{cyn}":t$=left$("00",2-
len(t$))+t$:return
122 window 2,23,77,23,1:printchr$(14);:input"{yel}Sector  0{left}{left}{left}";s$:print"{cyn}":s$=left$("00",2-
len(s$))+s$:return
123 window 2,23,77,23,1:printchr$(14);:input"{yel}Side  0{left}{left}{left}";d$:print"{cyn}":d$=left$("0",1-
len(d$))+d$:return
124 window 2,23,77,23,1:printchr$(14);:return
125 rem hauptprogramm
126 printchr$(14);"           {wht}****  BURSTMON v3.1  (c) 1986 by Karsten Schramm  ****{cyn}"
127 print chr$(27)"m";chr$(142);
128 print" U-------------------------------------------------------------------------------I"
129 fora=1to2:print" |                                                                               |":nexta
130 print" +-------------------------------------------------------------------------------+"
131 fora=1to16:print" |                                                                                |":nexta
132 print" J-------------------------------------------------------------------------------K"
133 print" U-------------------------------------------------------------------------------I"
134 print" |                                                                               |"
135 print" J-------------------------------------------------------------------------------K"
136 gosub114
137 rem befehlseingabe
138 c$="":gosub120
139 x=1:restore 234
140 read a$:ifleft$(c$,1)=left$(a$,1) then 143
141 x=x+1:ifa$<>"*" then140
142 printchr$(7):goto 137
143 on x gosub 147,155,163,171,189,202,205,222,228
144 :
145 :
146 goto 137
```

```
147 rem @
148 gosub 124
149 if c$="@" then begin:ifdd>1 then print"{lred}";:else print"{yel}";
150 printchr$(14);"Disk Status: ";dd$;"{cyn}":getkeya$:return:bend
151 if c$="@8" then u=8:poke284,u:printchr$(14);"{yel}Unit#";u;"{cyn}":sleep1:return
152 if c$="@9" then u=9:poke284,u:printchr$(14);"{yel}Unit#";u;"{cyn}":sleep1:return
153 bs$=mid$(c$,2):open 1,u,15:print#1,bs$:close1:c$="@":goto147
154 print#1chr$(7):return
155 rem read
156 t$="":gosub121:ifval(t$)<0 or val(t$)>70 then 156
157 s$="":gosub122:ifval(s$)<0 or val(s$)>255 then 157
158 d$="":gosub123:ifval(d$)<0 or val(d$)>1 then 158:d=val(d$)
159 gosub112
160 open 1,u,15:bs$="u0"+chr$(0or(16*d))+chr$(val(t$))+chr$(val(s$))+chr$(1):gosub361:sys
dec("1460"),0,1,6:dd=ds:dd$=ds$:close1
161 gosub308:gosub112
162 c$="@":gosub 147:ifcs$="ok"then goto171:else return
163 rem write
164 t$="":gosub121:ifval(t$)<0 or val(t$)>70 then 164
165 s$="":gosub122:ifval(s$)<0 or val(s$)>255 then 165
166 d$="":gosub123:ifval(d$)<0 or val(d$)>1 then 166:d=val(d$)
167 gosub112
168 open 1,u,15:bs$="u0"+chr$(2or(16*d))+chr$(val(t$))+chr$(val(s$))+chr$(1):gosub361:sys
dec("1460"),0,2,6:dd=ds:dd$=ds$:close1
169 gosub308:gosub112
170 c$="@":gosub 147:ifcs$="ok"then goto171:else return
171 m=14:rem edit
172 gosub108:poke 252,11:gosub124:print"edit":gosub117:printchr$(m);
173 forx=0to15:poke251,x*16:sys dec("14a8"):print:next x
174 p=2816:printchr$(27)"s";"{home}";
175 getkeya$:ifa$="□"then188
176 if a$="{up}"ora$="{down}"ora$="{rght}"ora$="{left}"thengoto179:else ifa$=" "then a$="{rght}":goto179
177 ifa$=chr$(13)ora$=chr$(141)orpos(0)>53thenprint:goto175
178 if peek(211)=2anda$=chr$(160)then m=xor(m,128):goto172:else:goto 182
179 ifpeek(235)>=20anda$="{down}"andp<2816+wthenp=p+16:poke252,int((p+240)/256):poke251,(p+240)-
peek(252)*256:printchr$(27)"v{up}":sysdec("14a8"):print"{up}":goto175
180 if peek(235)<=5anda$="{up}"andp>2816 then p=p-16:printchr$(27)"w";:poke252,int(p/256):poke 251,p-
peek(252)*256:print"{home}";:sysdec("14a8"):print"{home}";:goto175
181 printa$;:goto175
182 if((a$>"/"anda$<":")or(a$>"@"anda$<"g"))andpos(0)>6andpos(0)<54thenbegin:z=peek(235)-5:s=pos(0)-
7:b=p+z*16+int(s/3):bn=s/1.5
183 if(bn-int(bn))=0 then ne=1:nn=16:goto185
184 if(bn-int(bn))>.6then ne=16:nn=1:else print" ";:goto175
185 we=val(a$):ifa$>"@"thenwe=asc(a$)-55
186 by=((peek(b)and(ne*15))or(nn*we)):pokeb,by:printa$;:bend:goto175
187 goto175
188 gosub107:printchr$(27)"u"chr$(27);chr$(14);:return
189 rem query disk
190 t$="":gosub121:ifval(t$)<0 orval(t$)>70 then 190
191 d$="":gosub123:ifd$<>"0"andd$<>"1" then191
192 d=val(d$):open 1,u,15:bs$="u0"+chr$(138or(d*16))+chr$(val(t$)):gosub361
193 sys dec("1460"),7,3,4:dd=ds:dd$=ds$:close1:gosub 308:gosub117
194 if f$="MFM" then begin
195 print"Analyzing the disk brought the following parameters:":print:print
196 print"Logical Track:";peek(dec("1704")):print
197 print"Number of Sectors:";peek(dec("1705")):print
198 print"Minimum Sector:";peek(dec("1703")):print
199 print"Maximum Sector:";peek(dec("1702")):print
200 print"MFM-Sector interleave:";peek(dec("1701")):print:bend
201 gosub112:return
202 rem exit
203 poke 54784,26:poke 54785,0:print"{home}{home}{cyn}{clr}";chr$(27)"l";chr$(27)"s";chr$(142)
204 end
205 rem user
```

```
206 gosub124:print"user 0"
207 gosub117:print"The Following BURST-Commands Are Available:":print
208 print"1 - BURST-READ (only 1 sector)"
209 print"2 - BURST-WRITE (only 1 sector)"
210 print"3 - INQUIRE DISK"
211 print"4 - FORMAT MFM"
212 print"5 - FORMAT GCR (no directory)"
213 print"6 - SECTOR{$a0}INTERLEAVE"
214 print"7 - QUERY{$a0}DISK{$a0}FORMAT"
215 print"8 - INQUIRE STATUS"
216 print"9 - UTILITY{$a0}MODE"
217 print"x - EXIT"
218 print:print:print"Please select (1-9,x): ";
219 getkey a$:ifa$<"1"or(a$>"9"anda$<>"x") then 219
220 gosub117:on val(a$)+1 goto 221,155,163,222,235,249,255,189,267,280
221 gosub117:return
222 rem inquire disk
223 d$="0":s$="00":t$="00":cs$="ok":f$="GCR":w$="0256":k$="21"
224 open 1,u,15:bs$="u0"+chr$(4):gosub361
225 sys dec("1460"),1,3,3:dd=ds:dd$=ds$:close1
226 gosub308:gosub112
227 return
228 rem $
229 gosub124:print"directory on unit#";u
230 gosub117:printchr$(27)+"l":slow:if u=9 then directory on u9:goto232
231 directory on u8
232 print:print"{yel}Press any key!{cyn}":dd=ds:dd$=ds$:printchr$(27)+"m":fast
233 getkeya$:gosub117:c$="@":goto147
234 data "@","read","write","edit","query disk","x","user","inquire disk","$","*"
235 rem format mfm
236 gosub124:a$="":input"Index address mark written (y/n)  y{left}{left}{left}";a$:ifa$="y"then
by=(byor64):else ifa$<>"n"then236
237 gosub124:a$="":input"Format double sided (y/n)  y{left}{left}{left}";a$:ifa$="y"thenby=(byor32):goto
239:else ifa$<>"n"then237
238 gosub124:a$="":input"Which side do you wish to format (0/1)  0{left}{left}{left}";a$:ifa$="1"then
by=(byor16):else ifa$<>"0"then238
239 bs$="u0"+chr$(by):a$="":gosub124:input"Logical starting sector (0-63)
0{left}{left}{left}";a$:a=val(a$):ifa<0ora>63then239:else bs$=bs$+chr$(aor128)
240 gosub124:a$="":input"Sector interleave (0-255)  0{left}{left}{left}";a$:a=val(a$):ifa<0ora>255then240:else
bs$=bs$+chr$(a)
241 gosub124:a$="":input"Sector size (0,1,2,3=128,256,512,1024)
1{left}{left}{left}";a$:a=val(a$):ifa<0ora>3then241:elsebs$=bs$+chr$(a)
242 gosub124:a$="":input"Last track number (0-39)  39{left}{left}{left}{left}";a$:a=val(a$):ifa<0ora>39
then242:else bs$=bs$+chr$(a)
243 gosub124:a$="":input"Number of sectors (depends on sector size)";a$:a=val(a$):ifa<0ora>26 then243:else
bs$=bs$+chr$(a)
244 gosub124:a$="":input"Logical starting track  0{left}{left}{left}";a$:a=val(a$):ifa<0ora>39then244:else
bs$=bs$+chr$(a)
245 gosub124:a$="":input"Starting track offset  0{left}{left}{left}";a$:a=val(a$):ifa<0ora>39then245:else
bs$=bs$+chr$(a)
246 gosub124:a$="":input"Fill byte  229{left}{left}{left}{left}{left}{left}";a$:a=val(a$):ifa<0ora>255then246:else
bs$=bs$+chr$(a)
247 open 1,8,15,bs$:dd=ds:dd$=ds$:close1
248 c$="@":goto 147
249 rem format gcr
250 by=6:a$="":gosub124:input"Partial format (y/n)  y{left}{left}{left}";a$:ifa$="y"then by=(byor128):else
ifa$<>"n"then250
251 bs$="u0"+chr$(by)+chr$(0)
252 gosub124:a$="":input"Character for ID 1  x{left}{left}{left}";a$:a$=left$(a$,1):bs$=bs$+a$
253 gosub124:a$="":input"Character for ID 2  x{left}{left}{left}";a$:a$=left$(a$,1):bs$=bs$+a$
254 goto 247
255 rem sector interleave
256 by=8:a$=""
257 gosub124:input"Want to read or write interleave (r/w)  r{left}{left}{left}";a$
```

```
258 if a$<>"w"anda$<>"r"thenprint"{$07}":goto257
259 if a$="r" then begin:by=(byor128):a$=""
260 gosub124:print"Sector interleave is now: ";
261 bs$="u0"+chr$(by):gosub361:open
1,u,15:sysdec("1460"),1,3,3:dd=ds:dd$=ds$:close1:printpeek(dec("00fa")):getkeya$:goto266:bend
262 a$=""
263 gosub124:input"Please enter new sector interleave (dec 0-255/hex $00-$ff)  5{left}{left}{left}";a$:if
len(a$)>3 orlen(a$)<1 then 263
264 if left$(a$,1)="$"thena$=chr$(dec(mid$(a$,2,2))):else a$=chr$(val(a$))
265 bs$="u0"+chr$(by)+a$:gosub361:open 1,u,15:sysdec("1460"),0,3,4:dd=ds:dd$=ds$:close1:goto266
266 return
267 rem inquire status
268 by=12 :a$=""
269 gosub124:input"Want to read or write status (r/w)  r{left}{left}{left}";a$
270 if a$<>"w"anda$<>"r"thenprint"{$07}":goto269
271 if a$="r" then begin:by=(byor128):a$=""
272 gosub124:input"Want to test disk change (y/n)
y{left}{left}{left}";a$:ifa$<>"y"anda$<>"n"thenprint"{$07}":goto272
273 ifa$="y"thenby=(byor64)
274 bs$="u0"+chr$(by):gosub361:open
1,u,15:sysdec("1460"),1,3,3:dd=ds:dd$=ds$:close1:gosub308:gosub112:goto279:bend
275 a$=""
276 gosub124:input"Please enter new BURST-Statusbyte (dec 0-255/hex $00-$ff)  0{left}{left}{left}";a$:if
len(a$)>3 orlen(a$)<1 then 276
277 if left$(a$,1)="$" thena$=chr$(dec(mid$(a$,2,2))):else a$=chr$(val(a$))
278 bs$="u0"+chr$(by)+a$:gosub361:open 1,u,15:sysdec("1460"),0,3,4:dd=ds:dd$=ds$:close1:goto279
279 return
280 rem utility mode
281 gosub124:print"utility mode"
282 gosub117:print"Please select one of the following commands:":print
283 print"1 - Set sector interleave for GCR written disk"
284 print"2 - Set the number of retries in case of failed job"
285 print"3 - Analysis of ROM signature"
286 print"4 - Mode select (1541 or 1570/71)"
287 print"5 - Head select (1541 mode only)"
288 print"6 - Set device number"
289 print"x - Exit option"
290 getkey a$:ifa$<"1" or (a$>"6" and a$<>"x") then 290
291 on val(a$)+1 goto 307,292,294,296,298,300,302
292 gosub124:input"GCR sector interleave  6{left}{left}{left}";a$:a=val(a$):ifa<1ora>255then 292
293 bs$="u0>s"+chr$(a):goto305
294 gosub124:input"Number of retries  5{left}{left}{left}";a$:a=val(a$):ifa<0ora>255then294
295 bs$="u0>r"+chr$(a):goto305
296 gosub124:print"Please wait a moment ..."
297 bs$="u0>t":goto 305
298 gosub124:input"1570/71 mode or 1541 mode (1/0)  1{left}{left}{left}";a$:ifa$<>"0"anda$<>"1"then298
299 bs$="u0>m"+a$:goto305
300 gosub124:input"Head 0 or 1 (1/0)  0{left}{left}{left}";a$:ifa$<>"0"anda$<>"1"then300
301 bs$="u0>h"+a$:goto305
302 gosub124:input"Device number (8-30)  8{left}{left}{left}";a$:ifval(a$)<8orval(a$)>30then302
303 bs$="u0>"+chr$(z)
304 open1,u,15,bs$:close1:return
305 open1,u,15,bs$:dd=ds:dd$=ds$:close1
306 gosub117:c$="@":goto147
307 return
308 rem burst-meldungen auswerten
309 data "ok"
310 data "ok"
311 data "sector not found"
312 data"no sync"
313 data"data block not found"
```

```
314 data"data block checksum error"
315 data"format error"
316 data"verify error"
317 data"write protect error"
318 data"header block checksum error"
319 data"data extends into next block"
320 data"disk id mismatch/disk change"
321 data"no defined error message"
322 data"no defined error message"
323 data"syntax error"
324 data"no drive present"
325 :
326 data "ok"
327 data "ok"
328 data "sector not found"
329 data"no address mark"
330 data"no defined error message"
331 data"data crc error"
332 data"format error"
333 data"verify error"
334 data"write protect error"
335 data"header block checksum error"
336 data"no defined error message"
337 data"disk change"
338 data"no defined error message"
339 data"no defined error message"
340 data"syntax error"
341 data"no drive present"
342 ifpeek(dec("1700"))=128then
begin:gosub117:print"{down}{down}{down}{down}{down}{down}{down}{down}{lred}CAUTION!!! The connected drive is
not in the 1570/71 mode!"
343 print:print:print"{$07}{yel}Press any key to continue{cyn}":getkeya$
344 absichtlicher fehler in dieser zeile
345 bend
346 cs=peek(dec("00fa"))
347 if cs>127 then f$="MFM":restore326:else f$="GCR":restore309
348 n$=str$(sgn(csand64))
349 if f$="MFM" then begin
350 if(csand48)=0 then w=0:w$="0128":poke176,1:poke177,128:k$="26"
351 if(csand48)=16 then w=0:w$="0256":poke176,1:poke177,0:k$="16"
352 if(csand48)=32 then w=256:w$="0512":poke176,2:poke177,0:k$="09"
353 if(csand48)=48 then w=768 :w$="1024":poke176,4:poke177,0:k$="05"
354 bend:else begin:w=0:w$="0256":poke176,1:poke177,0:t=val(t$):ift>35 thent=t-35
355 if t<36 then k$="17"
356 if t<31 then k$="18"
357 if t<25 then k$="19"
358 if t<18 then k$="21":bend
359 forx=0to(csand15):readcs$:nextx
360 return
361 rem befehlsstring in speicher schreiben
362 forx=1tolen(bs$)
363 poke 5167+x,asc(mid$(bs$,x,1)):nextx:poke251,0:poke252,11
364 return
365 rem fehler
366 ifer=14andel=20468then print"{$07}":resume276
367 ifer=14andel=20368then print"{$07}":resume263
368 ifer=11 then resume 137
369 if er=5 then resume 142
370 resume next
371 rem maschinenprogramm initialisieren
372 print"{clr}datas werden initialisiert!"
373 restore 376
374 forx=4864to5373:read a:poke x,a:next x
375 return
376 data 120 , 44 , 13 , 220 , 166 , 176 , 173 , 0 , 221 , 73
```

```
377 data 16 , 141 , 0 , 221 , 169 , 8 , 44 , 13 , 220 , 240
378 data 251 , 173 , 0 , 221 , 73 , 16 , 141 , 0 , 221 , 173
379 data 12 , 220 , 133 , 250 , 41 , 15 , 201 , 2 , 176 , 34
380 data 160 , 0 , 169 , 8 , 44 , 13 , 220 , 240 , 251 , 173
381 data 0 , 221 , 73 , 16 , 141 , 0 , 221 , 173 , 12 , 220
382 data 145 , 251 , 200 , 196 , 177 , 208 , 231 , 230 , 252 , 202
383 data 208 , 226 , 24 , 36 , 56 , 88 , 96 , 234 , 234 , 234
384 data 120 , 169 , 64 , 133 , 253 , 166 , 176 , 160 , 0 , 173
385 data 5 , 213 , 9 , 8 , 141 , 5 , 213 , 169 , 127 , 141
386 data 13 , 220 , 169 , 0 , 141 , 5 , 220 , 169 , 3 , 141
387 data 4 , 220 , 173 , 14 , 220 , 41 , 128 , 9 , 85 , 141
388 data 14 , 220 , 44 , 13 , 220 , 173 , 0 , 221 , 205 , 0
389 data 221 , 208 , 248 , 69 , 253 , 41 , 64 , 240 , 242 , 165
390 data 253 , 73 , 64 , 133 , 253 , 177 , 251 , 141 , 12 , 220
391 data 169 , 8 , 44 , 13 , 220 , 240 , 251 , 200 , 196 , 177
392 data 208 , 219 , 230 , 252 , 202 , 208 , 214 , 169 , 8 , 141
393 data 14 , 220 , 173 , 5 , 213 , 41 , 247 , 141 , 5 , 213
394 data 44 , 13 , 220 , 173 , 0 , 221 , 9 , 16 , 141 , 0
395 data 221 , 169 , 8 , 44 , 13 , 220 , 240 , 251 , 173 , 12
396 data 220 , 133 , 250 , 173 , 0 , 221 , 41 , 239 , 141 , 0
397 data 221 , 165 , 250 , 41 , 15 , 201 , 2 , 176 , 2 , 24
398 data 36 , 56 , 88 , 96 , 120 , 170 , 240 , 67 , 44 , 13
399 data 220 , 173 , 0 , 221 , 73 , 16 , 141 , 0 , 221 , 169
400 data 8 , 44 , 13 , 220 , 240 , 251 , 173 , 0 , 221 , 73
401 data 16 , 141 , 0 , 221 , 173 , 12 , 220 , 133 , 250 , 234
402 data 16 , 33 , 41 , 15 , 201 , 2 , 176 , 29 , 202 , 240
403 data 24 , 169 , 8 , 44 , 13 , 220 , 240 , 251 , 173 , 0
404 data 221 , 73 , 16 , 141 , 0 , 221 , 173 , 12 , 220 , 157
405 data 0 , 23 , 76 , 10 , 20 , 24 , 36 , 56 , 88 , 96
406 data 0 , 0 , 0 , 0 , 32 , 32 , 32 , 32 , 32 , 32
407 data 32 , 32 , 32 , 32 , 32 , 32 , 32 , 32 , 32 , 32
408 data 32 , 32 , 32 , 32 , 32 , 32 , 32 , 32 , 32 , 32
409 data 32 , 32 , 32 , 32 , 32 , 32 , 32 , 32 , 32 , 32
410 data 32 , 32 , 32 , 32 , 32 , 32 , 32 , 32 , 32 , 32
411 data 32 , 32 , 72 , 138 , 72 , 173 , 28 , 10 , 41 , 191
412 data 141 , 28 , 10 , 162 , 1 , 32 , 201 , 255 , 162 , 0
413 data 189 , 48 , 20 , 32 , 210 , 255 , 232 , 136 , 208 , 246
414 data 32 , 204 , 255 , 104 , 170 , 104 , 160 , 0 , 140 , 0
415 data 23 , 44 , 28 , 10 , 80 , 15 , 202 , 208 , 3 , 76
416 data 0 , 19 , 202 , 208 , 3 , 76 , 80 , 19 , 76 , 224
417 data 19 , 160 , 128 , 140 , 0 , 23 , 96 , 0 , 0 , 0
418 data 0 , 0 , 0 , 0 , 169 , 46 , 32 , 210 , 255 , 165
419 data 252 , 56 , 233 , 11 , 170 , 165 , 251 , 32 , 159 , 184
420 data 32 , 125 , 255 , 32 , 0 , 160 , 0 , 177 , 251 , 32
421 data 194 , 184 , 169 , 32 , 32 , 210 , 255 , 200 , 192 , 16
422 data 144 , 241 , 32 , 125 , 255 , 32 , 32 , 18 , 0 , 160
423 data 0 , 177 , 251 , 72 , 41 , 127 , 201 , 32 , 104 , 176
424 data 2 , 105 , 64 , 32 , 210 , 255 , 200 , 192 , 16 , 208
425 data 236 , 169 , 0 , 133 , 244 , 133 , 245 , 169 , 146 , 76
426 data 210 , 255 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
```

14 Fehler im DOS 3.0 der 1570/71

Wie schon der Vorgänger, das DOS 2.6 der 1541, so enthält auch das DOS 3.0 einige Fehler, deren Aufklärung dringend erforderlich ist, um Fehlfunktionen der Floppy oder zerstörte Disketten zu vermeiden.

Da die 1570/71 das DOS der 1541 fast unverändert enthält, sind auch einige Fehler des DOS 2.6 wieder mit in das neue Gerät übertragen worden, und das, obwohl die Fehler der 1541 bei der Entwicklung der 1570/71 wohl ausreichend bekannt gewesen sein dürften.

Hier ist nun eine Aufstellung der Fehler, die entweder schon bekannt waren oder durch die Analyse des DOS-Listings neu entdeckt worden sind:

14.1 Der Befehl BLOCK-READ

Wie wir im Verlauf dieses Buches schon erfahren haben, wird der Befehl BLOCK-READ generell durch den U1-Befehl ersetzt und das aus folgendem Grund:

Beim Lesen eines Blockes mit dem B-R-Befehl, wird das allererste Datenbyte, also Byte 0, grundsätzlich nicht als Datenbyte eingelesen. Dieses Byte wird vielmehr der Anzahl der Bytes dieses Blocks gleichgesetzt (wie im letzten Block einer Datei) und an den BUFFER-POINTER-Befehl übergeben. Wollen Sie nun die Datenbytes aus dem gelesenen Sektor in den Computer holen, so können Sie maximal die Anzahl der Bytes lesen, die durch das allererste Byte im Sektor angegeben sind. In einem Diskmonitor ist dieser Befehl deshalb nicht anwendbar. Stellen Sie sich vor, Sie wollen einen Block einer Programmdatei einzeln lesen, und dieser Block steht auf Spur 2. Wenn der Zeiger auf die folgende Spurnummer dann auch auf Spur 2 zeigt, können Sie immer maximal die ersten zwei Bytes des eingelesenen Sektors mit GET# auslesen, da die Spurnummer zur Maximalzahl der Bytes des Blocks umgewandelt wird.

14.2 Der Befehl BLOCK-WRITE

Dieser Befehl hat prinzipiell den gleich Mangel wie der BLOCK-READ-Befehl, nur für den Schreibbetrieb zugeschnitten. Hier wird der aktuelle Pufferzeiger immer in den behandelten Sektor an die Position von Byte 0 geschrieben. Das dort vorhandene Datenbyte geht damit verloren.

14.3 Der Befehl BLOCK-ALLOCATE

Dieser Befehl ist prinzipiell in Ordnung, wenn der Block, der durch diesen Befehl belegt werden soll, auch wirklich frei ist. Wie Sie aus Kapitel 7.5 wissen, zeigt dieser Befehl im Falle eines belegten Blocks an, wo der nächste freie Block auf der Spur zu finden ist. Auch das ist noch in Ordnung. Es passiert hier jedoch zusätzlich, daß mit der Anzeige des nächsten freien Sektors alle Sektoren der Spur, auf der dieser freie Sektor zu finden ist, belegt werden. Dabei ist es der Floppy egal, ob es sich um Spur 18 oder eine andere wichtige Spur handelt.

14.4 Der Befehl REPLACE (@)

Dieser Befehl wurde im Verlauf dieses Buches schon erwähnt, wobei auf seine Gefährlichkeit nachdrücklich aufmerksam gemacht worden ist. Der Fehler dieses Befehls liegt in der Tatsache, daß die 1570/71 und auch die 1541 das abgemagerte Betriebssystem von Doppellaufwerken enthalten. Wenn Sie sich das DOS ansehen, so werden Sie sicherlich die äußerst umständliche Kanal- und Pufferhandhabung mit Belegungstabellen und Reserve-Belegungstabellen und dergleichen bemerkt haben. Diese Handhabung erlaubt ein schnelles und effektives Arbeiten bei Doppellaufwerken. Bei einem Einzellaufwerk ist das jedoch sinnlos, wenn nur mehr ein einziger Prozessor an der Arbeit ist. Im Fall der 1570/71 und auch der 1541 wurde es sogar gefährlich. Die Entwickler bei Commodore haben nämlich vergessen, alle Routinen der Floppy auf ein Einzellaufwerk mit nur mehr der halben Speicherkapazität gegenüber den Doppellaufwerken umzuschreiben. Einige der Routinen greifen deshalb in Extremsituationen noch auf Puffer zu, die gar nicht mehr existieren, und greifen natürlich ins Leere. In einem solchen Fall gehen der Floppy bei einem REPLACE-Befehl jedoch wichtige Daten verloren, so daß unter Umständen andere Dateien gelöscht werden, während die neu abgespeicherte Datei auf einmal unter einem ganz anderen Programmnamen auf der Diskette erscheint (vom Benutzer natürlich im ersten Moment nicht bemerkt).

Seien Sie also bei diesem Befehl auf der Hut. Es gehen unter anderem Gerüchte um, daß man den Fehler des REPLACE-Befehls dann vermeiden kann, wenn man bei jedem Diskettenzugriff immer die Drivenummer 0 mit angibt. Diese Gerüchte können der Wahrheit entsprechen oder auch nicht. Sie können es natürlich ausprobieren. Bedenken Sie jedoch immer: Sie wurden gewarnt! Es gibt nämlich auch Computeranwender, die hatten mit diesem Befehl noch nie irgend welche Probleme, da der Fehler äußerst selten auftritt. Was Sie hier jedoch zu hören bekommen, mußte ich am eigenen Leib schmerzlich erfahren, als sich ein mehrere Seiten langer Artikel in

Nichts auflöste und stattdessen ein Schnelladeprogramm auf der Diskette stand, das eigentlich unter einem ganz anderen Dateinamen dort abgespeichert worden war. Nun, immerhin hatte ich dadurch eine Sicherheitskopie meines Schnelladers, auch wenn der ganze Artikel neu getippt werden mußte.

14.5 Fehler im Handbuch zur 1570/71 und zum C128

Im Anleitungsbuch sowohl zur Floppystation 1570/71 als auch zum Commodore-128-Computer hat sich ein Fehler bei der Erläuterung des RECORD-Befehls in BASIC 7.0 eingeschlichen. Dort steht, daß die Angabe der Byteposition hinter der Recordnummer optional, das heißt nicht notwendig ist. Diese Aussage ist schlicht und einfach verkehrt. Bei einem RECORD-Befehl müssen Sie immer die Bytenummer angeben, auf die in einem Record positioniert werden soll.

Im Commodore-Handbuch zur 1570/71 fällt Ihnen vielleicht folgender Satz ins Auge: "SICHERHEITSMASSNAHME: JEDER RECORD#-BEFEHL MUSS ZWEIMAL ANGEGEBEN WERDEN. Um die höchst unwahrscheinliche Möglichkeit der Zerstörung relativer Dateidaten auszuschließen, müssen die RECORD#-Befehle zweimal angegeben werden, bevor ein Datensatz gelesen wird..."

Was sagt man nun dazu? Ich kann Sie an dieser Stelle beruhigen. Dieser Fehler, der offenbar Commodore selbst so mysteriös ist, daß eine Warnung an den Benutzer erfolgt, anstatt den Fehler aus dem Gerät zu entfernen, ist in Wirklichkeit nicht vorhanden. Hier sind die Hersteller auf einen Fehler im eigenen Bedienungshandbuch hereingefallen. Die "Möglichkeit der Zerstörung relativer Dateidaten" tritt nämlich nur dann auf, wenn im RECORD#-Befehl die Byteangabe, die angeblich optional ist, weggelassen wird. In diesem Fall kann es tatsächlich vorkommen, daß die Floppy ihre Daten ins Leere schreibt, wobei sie dann zwangsläufig verlorengehen.

Und selbst, wenn ein solcher Fehler vorhanden wäre, was er nicht ist, dann könnte die zweimalige Angabe des RECORD#-Befehls auch nicht helfen. Geben Sie also immer die Bytezahl zum Positionieren an, und Sie werden sehen, daß alles einwandfrei funktioniert, auch wenn der RECORD#-Befehl nur einmal angegeben wird.

14.6 Allgemeine Bemerkungen zur Fehlerbehandlung

Wie Sie aus den oben aufgeführten, schwerwiegenden Beispielen sehen, gibt es eine ganze Reihe von Fehlern im DOS der 1570/71 oder an einer anderen Stelle der Geräte. Die Angaben, die in diesem Buch gemacht wurden, beruhen auf meinen eigenen Erfahrungen und der Suche nach Lösungen. Wenn also ein Fehler angegeben ist, so existiert der Fehler in der gegebenen DOS-Version auf jeden Fall. Ist auch eine Lösung angegeben, die den Fehler behebt, so wurde dabei nicht auf Gerüchte geachtet, sondern eine Lösung erarbeitet, die sicher funktioniert (zum Beispiel beim RECORD#-Befehl).

Das Aufzählen dieser Fehler kann natürlich nicht davor schützen, daß noch andere Fehler existieren, die bisher nicht bekannt geworden sind. Sie werden im DOS-Listing auf eine ganze Reihe kleiner Unstimmigkeiten oder überflüssiger Details stoßen, die jedoch den Floppybetrieb bis zur Erstellung dieses Buches nicht nennenswert gestört haben.

A Die RAM-Belegung der 1570/71

A.1 Die RAM-Belegung in der Übersicht

Byte	Bedeutung
$0000-00ff	Zeropage; Arbeitsspeicher des DOS
$0100-0145	Stackbereich des Prozessors
$0146-01ff	Pufferbereich für zweiseitigen Diskbetrieb; Ausweichpuffer für Diskettenoperationen
$0200-02ff	Pufferbereich des DOS für Fehlermeldungen, Directory und Befehlsstrings
$0300-03ff	Puffer 0: enthält immer den aktuellen Block einer Datei, das gerade bearbeitet wird.
$0400-04ff	Puffer 1: enthält beim Suchen einer Datei den Teil des Directory, der die gesuchte Datei beinhaltet.
$0500-05ff	Puffer 2: USER-Puffer. Auf diesen Bereich beziehen sich unter anderem die U-Befehle. Er wird vom DOS für den Anwender freigehalten.
$0600-06ff	Puffer 3: Directory-Puffer. Enthält immer den aktuellen Block des Directory für die weitere Bearbeitung.
$0700-07ff	Puffer 4: BAM-Puffer. Enthält immer die aktuelle BAM der eingelegten Diskette. Dieser Bereich überschneidet sich bei der 1570/71 mit dem Puffer 5, der eigentlich hardwaremäßig nicht mehr existiert.

A.2 Die RAM-Belegung im Detail

Adresse	Bedeutung der Speicherstelle(n)
$0000-0005	Diese Speicherstellen bilden die Schnittstelle zwischen dem Hauptprogramm und dem Diskcontroller. Das Hauptprogramm des DOS schreibt dabei sogenannte Jobcodes in die entsprechende Speicherstelle, wobei die Jobcodes die nachfolgend aufgeführten Bedeutungen haben: $80/81 Lesen eines Sektors $88/89 Lesen eines Sektors auf dem aktuellen Track $90/91 Schreiben eines Sektors $a0/a1 Verify eines Sektors $b0/b1 Suchen einer Spur/eines Sektors (SEEK) $c0/c1 Positionieren des Kopfes auf Track 0 (BUMP) $d0/d1 Programm im Puffer ausführen $e0/e1 Diskettenprogramm in der Jobschleife ausführen $f0/f1 Diskette formatieren Nach der Ausführung eines Jobs hinterläßt der Diskcontroller spezielle Rückmeldungen, die die Ausführung eines Jobs bestätigen und den Status anzeigen: $00 ok $01 ok $02 Blockheader nicht gefunden $03 SYNC-Markierung auf der Diskette nicht gefunden $04 Datenblock nicht gefunden

$05 Datenprüfsumme falsch
$06 Fehler beim Formatieren aufgetreten
$07 Fehler bei Verify
$08 Diskette schreibgeschützt
$09 Headerprüfsumme falsch
$0a Datenblock zu lang
$0b falsche ID im Blockheader; Diskette gewechselt
$0c n.v.
$0d Indexloch wurde nicht gefunden
$0e Fehler in Befehlssyntax
$0f keine Diskette im Laufwerk
$10 Fehler bei Rückcodierung von GCR nach Binär

$0000 Jobspeicher für Puffer 0
$0001 Jobspeicher für Puffer 1
$0002 Jobspeicher für Puffer 2
$0003 Jobspeicher für Puffer 3
$0004 Jobspeicher für Puffer 4
$0005 Jobspeicher für Puffer 5 (im RAM nicht vorhanden)
$0006/7 Track und Sektor für Job mit Puffer 0
$0008/9 Track und Sektor für Job mit Puffer 1
$000a/b Track und Sektor für Job mit Puffer 2
$000c/d Track und Sektor für Job mit Puffer 3
$000e/f Track und Sektor für Job mit Puffer 4
$0010/1 Track und Sektor für Job mit Puffer 5 (n.v.)
$0012/3 ID 1 und ID 2 der Diskette im ASCII-Code
$0014/5 s.o. für Drive 1 (nicht implementiert)
$0016-$001a Zwischenspeicher für den Blockheader nach dem Lesen vonder Diskette. Dabei haben die Bytes folgende Bedeutung:
$0016 ID 1
$0017 ID 2
$0018 Tracknummer des Blockheaders
$0019 Sektornummer des Blockheaders
$001a Prüfsumme über den Blockheader
$001b Zwischenspeicher für Kommandobyte bei MFM-Steuerroutine ab $86e6
$001c Flag für Diskettenwechsel bei Änderung der Schreibschutz-Lichtschranke
$001d s.o. für Drive 1 (nicht implementiert)
$001e Zustand der Schreibschutzlichtschranke
$001f s.o. für Drive 1 (nicht implementiert)
$0020 Flags für Drivestatus:
Bit 4: Motor im Ausschaltmodus? l=ja; Motor läuft nach
Bit 5: Drivemotor 1=an; 0=aus
Bit 6: Steppermotor aktivi? 1=ja
Bit 7: Drive bereit? l=nein; Motor läuft gerade an
$0021 s.o. für Drive 1 (nicht implementiert)
$0022 Nummer des aktuellen Tracks für Diskcontroller
$0023 Flag für seriellen Bus: <>0 heißt 'schneller Busmodus'
$0024-$002d Bereich für Daten des Blockheaders nach dem Lesen von der Diskette im GCR-Code; ID-Feld bei MFM-Format
$002e/f Zwischenspeicher für Pufferadresse bei GCR-Umwandlung
$0030/1 aktuelle Pufferadresse
$0032/3 Zeiger auf Track- und Sektornummer im Jobspeicher
$0034 Pufferzeiger bei der GCR-Konvertierung (GCR nach Binär)
$0035 Verzögerungszähler Hi für Ausschalten des Drivemotors
$0036 Pufferzeiger bei Binärkonvertierung (Binär nach GCR)
$0037 Flags für Busbetrieb des seriellen Bus:
Bit 0: 1=letzer Sektor einer Datei wird behandelt
Bit 1,2: n.v.

Bit 3: Status der CLOCK-Leitung für nächstem Buszugriff
Bit 4: n.v.
Bit 5: n.v.
Bit 6: Flag für Burst-Modus: 0=langsamer Bus; 1=Burst
Bit 7: Flag für Systemtakt: 0= 1 MHz; 1= 2 MHz

$0038 Kennzeichen des letzten Datenblocks (normalerweise $07)
$0039 Kennzeichen für Blockheader; $08
$003a Prüfsumme über den aktuellen Pufferinhalt/Sektor
$003b Kommandobyte des U0-Befehls
$003c Sektorabstand im MFM-Format; Interleave-Faktor normal 5
$003d aktuelle Drivenummer für Job bei der Ausführung
$003e gerade aktives Laufwerk ($ff = kein Laufwerk)
$003f Puffernummer für gerade aktuellen Jobcode
$0040 Tracknummer des letzten Jobs; Zähler für GCR-Umwandlung
$0041 Puffernummer des letzten Jobs
$0042 Trackdifferenz zwischen neuer und vorheriger Spur
$0043 Maximalzahl der Sektoren auf einem Track
$0044 Anzahl der 256-Bytes-Blöcke pro MFM-Sektor (1,1,2,4)
$0045 Zwischenspeicher für aktuellen Jobcode
$0046 Zwischenspeicher für Zeichen bei der Busübertragung
$0047 Kennzeichen für Beginn eines Datenblocks (normal $07)
$0048 Zähler für Ausschaltverzögerung des Drivemotors Lo
$0049 Zwischenspeicher für Stackpointer
$004a Anzahl der Schritte beim Kopf transport
$004b Zwischenspeicher
$004c Nummer des zuletzt gelesenen Sektors
$004d Nummer des nächsten zu lesenden Sektors
$004e Zwischen speicher Pufferadresse Hi bei GCR-Umwandlung
$004f Zwischenspeicher Pufferadresse Lo bei GCR-Umwandlung
$0050 Flag für Puffer im GCR-Code: 1=GCR; 0=Binär
$0051 Tracknummer für Formatierung; sonst $ff
$0052-$0055 Zwischenspeicher für 4 Binärbytes bei der Umwandlung in die GCR-Äquivalente
$0056-$005d Zwischenspeicher für 8 (aufgeteilte) GCR-Bytes bei der Umwandlung in die Binäräquivalente
$005e Zwischenspeicher für Burst-Statusbyte (siehe 12.10)
$005f Zwischenspeicher für Jobcode bei Burst-Kommandos
$0060 minimale Sektornummer eines Tracks im MFM-Format
$0061 maximale Sektornummer eines Tracks im MFM-Format
$0062/3 Adresse der aktuellen Steppermotor- Routine
$0064 Anzahl der zu fahrenden Schritte bei Kopfpositionierung
$0065/6 Zeiger auf 'kurze' RESET-Routine; ohne Speichertest
$0067 Tracknummer, auf die positioniert werden soll
$0068 Flag zum Ermöglichen (0) oder Sperren (1) der automatischen Initialisierung einer neu eingelegten Diskette
$0069 GCR-Interleave-Wert; Sektorabstand normalerweise 6
$006a Steuerung des Kopfes bei Lesefehlern:
Bit 0-5: Anzahl der Leseversuche bei Leseproblemen
Bit 6: Kopf steht auf (1) oder neben (0) dem Track
Bit 7: BUMP bei Lesefehlern ausführen? 0=ja; 1=nein
$006b/c Zeiger auf Sprungtabelle der USER-Vektoren; $ffea
$006d/e Zeiger auf Bitmuster für einen Track in der BAM
$006f Anzahl der Jobs für Drive 0
$0070 Anzahl der Jobs für Drive 1 (nicht implementiert)
$0071 Zwischenspeicher
$0072 Zwischenspeicher
$0073 Anzahl der Side-Sektoren bei einer relativen Datei
$0074 Zwischenspeicher

$0075/6	Zeiger auf $0100; dient verschiedenen Zwecken
$0077	Gerätenummer +$20 für das LISTEN-Kommando
$0078	Gerätenummer +$40 für das TALK-Kommando
$0079	Flag für LISTEN; 1=gesetzt
$007a	Flag für TALK; 1=gesetzt
$007b	Byte für Kopfdejustierung bei Lesefehlern
$007c	Flag für ATN-Signal vom seriellen Bus
$007d	Flag für Prozessor im ATN-Modus
$007e	Tracknummer des letzten Diskettenzugriffs
$007f	aktuelle Drivenummer; enthält immer 0
$0080	aktuelle Tracknummer
$0081	aktuelle Sektornummer
$0082	aktuelle Kanalnummer
$0083	aktuelle Sekundäradresse für Befehlsausführung
$0084	interne Sekundäradresse für Dateibetrieb
$0085	aktuelles Datenbyte für Ein- und Ausgabe auf 1541-Bus
$0086	Zwischenspeicher
$0087	Zwischenspeicher
$0088	Zwischenspeicher
$0089	Zwischenspeicher
$008a	Zwischenspeicher
$008b-$008e	Speicher für Berechnungen und Rechenergebnisse
$008f-$0093	Akkumulator für Berechnungen
$0094/5	Zeiger auf Directorypuffer; normalerweise $0205
$0096	Sektornummer des aktuelle MFM-Sektors
$0097	Maximalzahl der Sektoren einer Spur im MFM-Format
$0098	Bitzähler für den seriellen 1541-Bus
$0099/a	Pufferadresse für Puffer 0: $0300
$009b/c	Pufferadresse für Puffer 1: $0400
$009d/e	Pufferadresse für Puffer 2: $0500
$009f/0	Pufferadresse für Puffer 3: $0600
$00a1/2	Pufferadresse für Puffer 4: $0700
$00a3/4	Adresse für INPUT-Puffer: $0200
$00a5/6	Adresse für ERROR-Puffer: $02d5
$00a7-$00ad	Pufferbelegungstabelle bei der Kanalzuweisung; $ff bedeutet dabei Puffer frei
$00ae-$00b4	Pufferbelegungstabelle 2 bei der Kanalzuweisung; $ff bedeutet Puffer frei; Tabelle für Zweipufferbetrieb
$00b5-$00ba	Tabelle der Lo-Bytes der Recordnummern für jeden Kanal
$00bb-$00c0	Tabelle der Hi-Bytes der Recordnummern für jeden Kanal
$00c1-$00c6	Tabelle der Zeiger auf das aktuelle Datenbyte für die Übertragung einer Datei
$00c7-$00cc	Tabelle der Recordlängen für jeden Kanal bei der Bearbeitung einer relativen Datei
$00cd-$00d2	Tabelle der Side-Sektoren für eine relative Datei
$00d3	Zeiger auf ersten Dateinamen
$00d4	Zeiger auf aktuellen Record
$00d5	Nummer des aktuellen Side-Sektors
$00d6	Zeiger in Side-Sektor
$00d7	Zeiger in aktuellen Record
$00d8-$00dc	Sektornummer des Dateieintrags im Directory

$00dd-$00e1	Tabelle der Zeiger in die Directory-Einträge
$00e2-$00e6	Standardwerte für Drivenummem jeder Datei
$00e7-$00eb	Tabelle der behandelten Dateitypen
$00ec-$00f1	Kanal-Dateityp; Tabelle für Zuordnung der Dateitypen zu einem jeweiligen Kanal
$00f2-$00f7	Kanalstatus für jeden Kanal
$00f8	Flag für EOI-Signal
$00f9	Aktuelle Puffernummer
$00fa-$00fe	Tabelle für verschiedene Anwendungen
$00ff	Flags für Drivestatus und aktives Laufwerk
$0100	s.o. für Drive 1 (nicht implementiert)
$0101	Formatkennzeichen der Diskette aus der BAM
$0102	s.o. für Drive 1 (nicht implementiert)
$0103-$0145	Hardware-Stack des Prozessors
$0146-$01ba	BAM Zusatz puffer für zweiseitige Disketten
$01bb-$01ff	Ausweichpuffer bei der Umwandlung von Werten in das GCRFormat, da hierbei deren Länge zunimmt
$0200-$0229	INPUT-Puffer für Befehlsstring des Computers
$022a	Codenummer des auszuführenden Befehls
$022b-$023d	Kanaltabellej enthält für jeden Kanal den Statuswert:$ff: unbenutzt; $8x: Schreibkanal; $4x Lesekanal
$023e-$0243	aktuelles Datenbyte für jeden Kanal
$0244-$0249	Anzahl der noch zu übertragenden Zeichen jedes Kanals
$024a	gerade behandelter Dateityp
$024b	Länge des Befehlsstrings vom Computer
$024c	Zwischenspeicher für Sekundäradresse
$024d	Zwischenspeicher für aktuellen Jobcode
$024e	Arbeitsspeicher beim Suchen eines Sektors
$024f/0	Pufferbelegung; Bit=1 heißt 'Puffer belegt'
$0251	Flag für 'BAM dirty', d.h. BAM wurde im Puffer geändert
$0252	s.o. für Drive 1 (nicht implementiert)
$0253	Flag für 'Eintrag im Directory gefunden'
$0254	Flag für '$'; Ausgabe des Directory
$0255	Flag für Befehlsausführung; <>0, wenn Befehl anliegt
$0256	Flag für Belegung der Kanalnummern
$0257	Zeiger auf aktiven Puffer bei Zweipufferbetrieb
$0258	Recordlänge
$0259	Tracknummer für aktuellen Side-Sektor-Block
$025a	Sektornummer für aktuellen Side-Sektor-Block
$025b-$025f	Tabelle; enthält letzten Jobcode für Puffer
$0260-$0265	Sektornummern der Directoryeinträge in den Puffern
$0266-$026b	Zeiger auf die Directoryeinträge in den Puffern
$026c	Speicher für Duplikat der Fehlernummer; Fehlerflag
$026d	Flag für LED-Blinken bei Fehler
$026e	Nummer des letzten aktiven Laufwerks

$026f	Nummer des letzten bearbeiteten Sektors
$0270	aktuelle Kanalnummer
$0271	Anzahl der Bytes in einem Sektorblock ($7f,ff,ff,ff)
$0272/3	Speicher für Anzahl der Blöcke auf der Diskette
$0274	Länge des Befehlsstrings im INPUT-Puffer
$0275	Zeichen zum Suchen im Befehlsstring
$0276	letztes Zeichen plus 1 im INPUT-Puffer
$0277	Länge von Dateiname 1
$0278	Anzahl der Kommata; Länge von Dateiname 2
$0279	Zeiger auf 2. Dateinamen
$027a-$027f	Zeiger auf Dateitabelle
$0280-$0284	Tracknummern der Dateien für den aktuellen Puffer
$0285-$0289	Sektornummern der Dateien für den aktuellen Puffer
$028a	Joker (*) Flag
$028b	Flags für Befehlssyntax
$028c	Anzahl der Drivezugriffe
$028d	Flag für Diskettenzugriff
$028e	Nummer des zuletzt benutzten Laufwerks
$028f	Flag für Dateieintrag im Directory gefunden
$0290	Sektornummer des aktuellen Directoryblocks
$0291	Sektornummer des aktuellen Directoryeintrags
$0292	Zeiger auf ersten gültigen Directoryeintrag
$0293	zeigt letzten Block an; enthält dann 0
$0294	aktueller Pufferzeiger für Directory
$0295	Zähler für Dateieinträge
$0296	aktueller Dateityp
$0297	Betriebsart der aktuellen Datei (W,R,A,M)
$0298	zeigt Fehler bei der Ausführung eines Jobs an
$0299	Zeiger für Kopfdejustage bei Lesefehlern
$029a	Byte für Kopfpositionierung
$029b	Flag für BAM wieder hergestellt
$029c	s.o. für Drive 1 (nicht implementiert)
$029d	Tracknummer der BAM 1
$029e	Tracknummer der BAM 2
$029f	s.o. 1 für Drive 1 (nicht implementiert)
$02a0	s.o. 2 für Drive 1 (nicht implememtiert)
$02a1-$02a8	Zwischenspeicher für BAM-Eintragungen
$02a9/a	IRQ- Vektor für aktuelle Jobschleife (1571 oder 1541)
$02ab	Zähler für Anlaufen des Motors beim Disketteneinlegen
$02ac	Maximale Tracknummer auf der Diskette (36 oder 71)
$02ad	Zwischenspeicher
$02ae	Zwischenspeicher
$02af	Flag für Umschalten des Floppy-Betriebsmodus
$02b0-$02d4	Puffer für Erstellung des Directory bei der Ausgabe
$02d5-$02f8	ERROR-Puffer; enthält den Diskstatus im Klartext
$02f9	Flag für BAM neu auf Diskette schreiben; weil 'dirty'
$02fa	Anzahl der 'BLOCKS FREE' Lo
$02fb	s.o. für Drive 1 (nicht implementiert)
$02fc	Anzahl der 'BLOCKS FREE' Hi
$02fd	s.o. für Drive 1 (nicht implementiert)
$02fe	Parameter für Kopf transport
$02ff	s.o. für Drive 1 (nicht implementiert)

$0300-$03ff Puffer 0 (Hauptarbeitspuffer)
$0400-$04ff Puffer 1 (enthält aktuellen Teil des Directory)
$0500-$05ff Puffer 2 (USER-Pufferj normalerweise frei)
$0600-$06ff Puffer 3 (enthält letzten Block des Directory)
$0700-$07ff Puffer 4 (BAM-Pufferj enthält Block 18,0)
$0800-$7fff keine RAM-Belegung
$8000-$ffff DOS 3.0 der 1570/71

B Tabellen mit Daten der Interfacebausteine

B.1 Der VIA 6522

Pin	Bedeutung
01	Vss-; Versorgungsspannung Masse
02-09	I/O-Port A; 8 Datenleitungen
10-17	I/O-Port B; 8 Datenleitungen
18	CB1; Handshakeleitung 1 für Port B
19	CB2; Handshakeleitung 2 für Port B
20	Vcc+; Versorgungsspannung +5 Volt
21	-IRQ; IRQ-Leitung an Prozessor
22	R/W; Read/Write-Leitung
23	-CS2; Chip select für Bausteinadressierung
24	CS1; Chip select für Bausteinadressierung
25	02; Systemtakteingang
26-33	Datenleitungen zum Prozessor
34	-RES; RESET-Eingang
35-38	RS3-RSO; Register Select für Registeradressierung
39	CA2; Handshakeleitung 2 für Port A
40	CA1; Handshakeleitung 1 für Port A

Tabelle B.1: Pinbelegung des VIA 6522

Register	Bedeutung
00 $0	PB; Datenregister für Port B
01 $1	PA; Datenregister für Port A
02 $2	DDRB; Datenrichtungsregister für B (Bit=l Ausgang)
03 $3	DDRA; Datenrichtungsregister für A (Bit=l Ausgang)
04 $4	T1L; Timer 1 Lo setzen und lesen
05 $5	T1H; Timer 1 Hi setzen, lesen und Start
06 $6	T1LL; Zugriff auf Timer 1 Lo
07 $7	T1LH; Zugriff auf Timer 1 Hi
08 $8	T2L; Timer 2 Lo lesen; IRQ löschen Timer 2 Lo schreiben; IRQ nicht löschen
09 $9	T2H; Zugriff auf Timer 2 Hi; IRQ löschen
10 $a	SR; serielles I/O-Schieberegister
11 $b	ACR; Hilfs- Kontrollregister: Bit 0 - 1 gibt Eingangszwischenspeicher Port A frei Bit 1 - 1 gibt Eingangszwischenspeicher Port B frei Bit 2-3:00 - Sperrt Schieberegister 01 - Schiebetakt kommt von Timer 2 10 - Schiebetakt ist der Systemtakt 11 - Schiebetakt wird extern geliefert Bit 4 - 1 SR ist Ausgang; 0 SR ist Eingang Bit 5 - 0 Timer 2 zählt im Systemtakt abwärts 1 Timer 2 zählt im externen Takt von PB6 Bit 6 - 1 Takt von Timer 1 über PB7 ausgeben

	Bit 7 - 1 'free running mode' von Timer 1 0 einmaliges Herunterzählen von Timer 1
12 $c	PCR; Peripherie-Kontrollregister: Bit 0 - 0 IRQ von CA1 bei fallender Flanke 1 IRQ von CA1 bei steigender Flanke Bit 1-3:000 - IRQ von CA2 bei fallender Flanke 001 - s.o. aber unabhängige Betriebsart 010 - IRQ von CA2 bei steigender Flanke 011 - s.o. aber unabhängige Betriebsart 100 - CA2 Ausgang Lo bei Zugriff 101 - CA2 Ausgang Lo-Impuls bei Zugriff 110 - CA2 Ausgang auf Lo 111 - CA2 Ausgang auf Hi Bit 4-7: siehe Bits 0-3 für CB2
13 $d	IFR; Interrupt-Flag-Register; IRQ anzeigen Bit 0 - 1 Flanke an CA2 Bit 1 - 1 Flanke an CA1 Bit 2 - 1 Schieberegister ein- oder ausgegeben Bit 3 - 1 Flanke an CB2 Bit 4 - 1 Flanke an CB1 Bit 5 - 1 Unterlauf von Timer 2 Bit 6 - 1 Unterlauf von Timer 1 Bit 7 - 1 zeigt irgendeinen Interrupt an
14 $e	IER; Interrupt-Freigabe-Register; IRQs auswählen Diese Bits wählen den IRQ an, der dann analog im IFR angezeigt wird.
15 $f	Port A ohne Handshake

Tabelle B.2: Registerbeschreibung des VIA 6522

Bit/Pin	**Bedeutung**
0	Leitung für DATA IN (invertierender Eingang)
1	Leitung für DATA OUT (invertierender Ausgang)
2	Leitung für CLOCK IN (invertierender Eingang)
3	Leitung für CLOCK OUT (invertierender Ausgang)
4	ATN A; 1, jedes ATN wird von Hardware der Floppy autom. beantwortet
5	Lo-Bit der Hardware-Gerätenummer (DIP-Schalter 1)
6	Hi-Bit der Hardware-Gerätenummer (DIP-Schalter 2)
7	Leitung für ATN IN (invertierend; auch über CA1/VIA 1)

Tabelle B.3: Belegung von Port B des VIA 1 bei $1800

Pin/Bit	**Bedeutung**
0	Bit=0 zeigt an, daß der Kopf auf Track 0 steht
1	Bit=1 serieller Bus auf Ausgang Bit=0 serieller Bus auf Eingang
2	Diskettenseite für Tonkopf (1571) 0 oder 1
3	n.v.
4	n.v.
5	Systemtaktfrequenz; Bit=1 heißt 2 MHz; sonst 1 MHz
6	n.v.
7	zeigt BYTE-READY an; auch über CAI/VIA 2

Tabelle B.4: Belegung des Port A von VIA 1 bei $1801

Bit/Pin	Bedeutung
0	Ansteuerung für 1. Steppermotorspule
1	Ansteuerung für 2. Steppermotorspule
2	Steuerung für Drivemotor; Bit=1 heißt Motor an
3	Steuerung für LED am Laufwerk; 1 heißt LED an
4	WRITE PROTECT; Zustand der Schreibschutzlichtschranke; Bit=1 heißt Diskette ungeschützt; auch CA2/VIA 1
5/6	Steuerung der Aufzeichnungsrate auf der Diskette: Bit 6/5: 00 - 250000 Bits/s auf Track 31-35/66-70 01 - 266664 Bits/s auf Track 25-30/60-65 10 - 285712 Bits/s auf Track 18-24/53-59 11 - 307688 Bits/s auf Track 01-17/36-52
7	SYNC-Signal beim Lesen von Diskette

Tabelle B.5: Belegung des Port B von VIA 2 bei $lc00

Pin/Bit	Bedeutung
0-7	Paralleler Datenbus zum Diskcontroller (GCR)

Tabelle B.6: Belegung des Port B von VIA 2 bei $lc0l

B.2 Der CIA 6526

Pin	Bedeutung
01	VSS-; Versorgungsspannung Masse
02-09	PA0-PA7; Leitungen für I/O-Port A
10-17	PB0-PB7; Leitungen für I/O-Port B
18	-PC; Geht auf Lo, wenn Daten an einem Port anliegen
19	TOD; Eingang für Echtzeituhr (50 oder 60 Hz Signal)
20	Vcc+; Versorgungsspannung +5 Volt
21	-IRQ; Interruptleitung zum Prozessor
22	R/W; geht auf Lo, wenn der Prozessor Daten ausgibt
23	-CS; Leitung adressiert den Baustein; ghet dann auf Lo
24	-FLAG; Eingang zur Anzeige von Daten (s.o. -PC)
25	02; Eingang für Systemtakt
26-33	D7-D0; Datenbus zum Prozessor
34	-RES; RESET-Eingang vom Prozessor
35-38	RS3-RS0; Registerauswahlleitungen
39	SP; 'serial port'; Aus- und Eingang für Schieberegister
40	CNT; Taktleitung für Schiebetakt bei SP

Tabelle B.7: Pinbelegung des CIA 6526

Register	Bedeutung
00 $0	PRA; I/O-Port A
01 $1	PRB; I/O-Port B
02 $2	DDRA; Datenrichtungsregister Port A; Bit=1 Ausgang
03 $3	DDRB; Datenrichtungsregister Port B; Bit=1 Ausgang

04 $4 TAL; Timer A Lo; Lesen und Schreiben
05 $5 TAH; Timer A Hi; Lesen und Schreiben
06 $6 TBL; Timer B Lo; Lesen und Schreiben
07 $7 TBH; Timer B Hi; Lesen und Schreiben
08 $8 TOD 10THS; 1/10 Sekunden der Echtzeituhr:
Bit 0-3: Lesen: 1/10 Sekunden der Uhrzeit (BCD)
Bit 4-7: O; n.v.
oder
Bit 0-3: Schreiben: 1/10 Sekunden für Uhrzeit
Bit 4-7: 0; n.v.
oder
Bit 0-3: Schreiben: 1/10 Sekunden für Alarmzeit
Bit 4-7: 0; n.v.
09 $9 TOD SEC; Sekunden der Echtzeituhr (BCD):
Bit 0-3: Lesen: Einersekunden (BCD)
Bit 4-6: Lesen: Zehnersekunden (BCD)
Bit 7: immer 0; n.v.
oder
Bit 0-3: Schreiben: Sekunden für Echtzeit setzen
Bit 4-6: Schreiben: 10er Sekunden für Echtzeit
Bit 7: immer 0; n.v.
oder
Bit 0-3: Schreiben: Sekunden für Alarmzeit setzen
Bit 4-6: Schreiben: 10er Sekunden für Alarmzeit
Bit 7: immer 0; n.v.
10 $a TOD MIN; Minuten der Echtzeituhr (BCD):
Belegung siehe Register 9 für Minuten
11 $b TOD HR; Stunden der Echtzeituhr (BCD):
Bit 0-3: Lesen: Einerstunden der Echtzeituhr (BCD)
Bit 4: Lesen: 10er Stunden der Echtzeituhr
Bit 5,6: 0; n.v.
Bit 7: Lesen: 0=AM; l=PM
Schreibfunktionen analog zu Register 9
12 $c SDR; Port für serielles Schieberegister
13 $d ICR; Interrupt Control Register:
Lesen: Interruptursache abfragen
Bit 0: 1= Unterlauf von Timer A
Bit 1: 1= Unterlauf von Timer B
Bit 2: 1= eingestellte Alarmzeit erreicht
Bit 3: 1= Daten im SDR ausgegeben oder empfangen
Bit 4: 1= aktiver Lo-Pegel am Pin -FLAG aufgetreten
Bit 5: n.v.
Bit 6: n.v.
Bit 7: 1 zeigt an, daß ein Interrupt, der in der Maske gewählt wurde, aufgetreten ist.
Schreiben: Interruptmaske setzen (entspr. Bit 7)
Bit 0: 1= Unterlauf von Timer A wird überwacht
Bit 1: 1= Unterlauf von Timer B wird überwacht
Bit 2: 1= Erreichen der Alarmzeit wird überwacht
Bit 3: 1= SDR-Betrieb wird überwacht
Bit 4: 1= Pin -FLAG wird überwacht
Bit 5: n.v.
Bit 6: n.v.
Bit 7: 0= Maskenbits entsprechend löschen
1= Maskenbits entsprechend setzen
14 $e CRA: Control Register A
Steuerung für Timer A; Lesen und Schreiben
Bit 0: 1= Timer A starten

Bit 1: 1= Unterlauf an PB6 anzeigen
Bit 2: 1= Unterlauf dreht den Zustand von PB6 um
0= Unterlauf löst einen Hi-Impuls an PB6 aus
Bit 3: 1= Timer hält nach Unterlauf an
0= Timer startet nach jedem Unterlauf neu
Bit 4: 1= Timer neu laden und starten
Bit 5: 1= Trigger für Timer ist CNT Pin (Hi-FIanken)
0= Trigger für Timer ist der Systemtakt
Bit 6: 1= Schieberegister auf Ausgang
0= Schieberegister auf Eingang
Bit 7: 1= Frequenz für Echtzeituhr 50 Hz
0= Frequenz für Echtzeituhr 60 Hz

15 $f CRB: Control Register B
Steuerung für Timer B; Lesen und Schreiben
Bit 0: 1= Timer B starten
Bit 1: 1= Unterlauf an PB7 anzeigen
Bit 2: 1= Unterlauf dreht Zustand von PB7 um
0= Unterlauf löst einen Hi-Impuls an PB7 aus
Bit 3: 1= Timer hält nach Unterlauf an
0= Timer startet nach jedem Unterlauf neu
Bit 4: 1= Timer neu laden und starten
Bit 6,5: 00 - Trigger für Timer ist der Systemtakt
01 - Trigger für Timer ist der CNT-Pin
10 - Trigger für Timer sind Unterläufe von Timer A
11 - Trigger für Timer sind Unterläufe von Timer A, wenn CNT auf Hi-Pegel liegt

Tabelle B.8: Registerbelegung des CIA 6526 bei Adresse $4000

B.3 Der Diskcontroller WD 1770

Pin	Bedeutung
01	-CS; Chip select Eingang für Adressierung des WD 1770
02	R/W; geht auf Lo, wenn Daten vom Prozessor kommen
03-04	A0-A1; Registerauswahlleitungen: Beim Lesen ergeben sich folgende Belegungen: A1,A0: 00 - Status Register 01 - Track Register 10 - Sektor Register 11 - Datenregister beim Schreiben: A1,A0: 00 - Befehlsregister 01 - Track Register 10 - Sektor Register 11 - Datenregister
05-12	DAL0-DAL7; Datenbus vom/zum Prozessor
13	-MR; RESET-Leitung vom Prozessor
14	GND; Versorgungsspannung Masse
15	Vcc; Versorgungsspannung +5 Volt
16	STEP; Signal für Steppermotor zum Positionieren
17	DIRC; Direction; Ausgang für Kopfbewegung; Lo-Pegel bewegt den Kopf nach außen Hi-Pegel bewegt den Kopf nach innen

18	CLK; Clock; Eingang für 8 MHz Systemtakt
19	-RD; Read Data; Datenleitung zum Lesen von Diskette
20	MO; Motor On; Ausgang Hi-Pegel schaltet den Drivemotor an
21	WG; Write Gate; Ausgang bei Schreibbetrieb auf Hi-Pegel
22	WD; Write Data; Ausgang für Daten zum Schreiben
23	-TR00; Eingang; geht auf Lo, wenn der Kopf auf Track 0 steht; Lichtschranke unterbrochen
24	-IP; Index Pulse; Eingang geht auf Lo, um dem WD 1770 mitzuteilen, daß das Indexloch gefunden wurde
25	-WPRT; Write Protect; Eingang, verhindert Schreiben, wenn ein Lo-Pegel anliegt
26	-DDEN; Double Density Enable; Eingang; Controller schreibt im FM-Format bei Hi-Pegel und im MFM-Format bei Lo-Pegel an -DDEN
27	DRQ; Data Request; Ausgang ist Hi, wenn Daten gelesen oder geschrieben worden sind
28	INTRQ; Interrupt Request; Ausgang geht auf Hi, wenn ein Kommando ausgeführt worden ist; wird beim Lesen des Statusregisters zurückgesetzt.

Tabelle B.9: Pinbelegung des WD 1770 von Western Digital

Register	**Bedeutung**
00 $0	Schreiben: Befehlsregister des WD 1770 Lesen: Statusregister des WD 1770: Bit 0: 1= Controller führt gerade einen Befehl aus Bit 1: je nach Kommandotyp: Typ 1: Zustand des Index-Pins; 1= Indexloch Typ 2,3: 1= Datenregister voll/leer (Lesen/Schreiben) Bit 2: je nach Kommandotyp: Typl: Zustand des Track 0 Pins; 0 bedeutet, daß der Kopf auf Track 0 steht Typ 2,3: 1 bedeutet, daß das Programm die vom Kopf kommenden Daten aus $2003 nicht rechtzeitig übernommen hat; sie sind deshalb verlorengegangen. Bit 3: 1= Prüfsummenfehler aufgetreten (CRC Error) Bit 4: 1= Sektor nicht gefunden Bit 5: je nach Kommandotyp: Typ 1: 1= Diskette hat nach Anlaufen 6 Umdrehungen hinter sich, das heißt, Motor hat jetzt konstante Drehzahl Typ 2,3: gibt Format an: 1= Datenmarke vorhanden 0= keine Datenmarke vorhanden Bit 6: bei Schreiben: Zustand des Schreibschutzes 1= Diskette ist schreibgeschützt Bit 7: 1= Drivemotor eingeschaltet
01 $1	Track Register; enthält zu bearbeitende Tracknummer
02 $2	Sektor Register; enthält zu bearbeitende Sektornummer
03 $3	Datenregister; enthält das Byte für Schreiben oder Lesen auf die oder von der Diskette

Tabelle B.10: Registerbeschreibung des WD 1770 bei Adresse $2000

C Das dokumentierte ROM-Listing der 1570/71

Hier haben Sie sozusagen das Kernstück des Buches vor Augen. Dieses ROM-Listing wurde mit viel Geduld und Arbeit so ausführlich wie nur irgend möglich dokumentiert und erlaubt es wohl jeden Assemblerprogrammierer, einen guten und tiefen Einblick in die Funktionsweise des FloppyBetriebssystems zu bekommen. Die Dokumentation des ROM-Listings hat sich zwar insgesamt auf über fünf Monate erstreckt, wobei ständig neu erscheinende Versionen von Commodore (dies ist Version -03 für die 1571 und -01 für die 1570) eine immer neue Überarbeitung nötig machten, es kann jedoch trotz allem nicht ausgeschlossen werden, daß dennoch Fehler vorhanden sind. Für Hinweise in dieser Richtung sind Verlag und Autor jederzeit dankbar.

Die Bedienung des DOS-Listings ist eigentlich recht einfach. Der Kommentar wurde so gestaltet, daß er prinzipiell auch ohne die nebenstehenden Maschinenbefehle durchgelesen werden kann, um so im "Schnellverfahren" einen Überblick über die einzelnen Routinen zu bekommen.

Wenn es nötig war, so wurde in den Kopf der jeweiligen Routinen noch eine detailliertere Erklärung der Funktionsweise eingefügt, so daß ein Aufruf dieser Routinen von eigenen Programmen aus ermöglicht wird.

Da sich die 1570 und die 1571 nur in sehr geringen Punkten unterscheiden, wurden beide Versionen in einem ROM-Listing zusammengefaßt (andernfalls wäre das Buch doppelt so dick geworden). Bei Unterschieden sind den Befehlen dann entweder ein ,,*1" oder ein "*0" vorangestellt. Das ,,*1" steht dann vor den Befehlen für die 1571, und das "*0" beschreibt die Befehle in der 1570.

Gefundene Betriebssystemfehler wurden in dem Listing ebenfalls, wenn möglich, gekennzeichnet. Es folgt dann zum Beispiel die Bemerkung "unsinniger Befehl" als Dokumentation.

So, nun aber viel Spaß bei der Analyse des DOS 3.0 der 1570/71 und bei der sicherlich davon profitierenden Programmierübung der 1570/71.

```
Das Listing des DOS 3.0 der 1570/71 Version -03 (1571) und -01 (1570)
------------------------------------------------------------------------------
8000   ROM-Prüfsumme zur Fehlererkennung bei Hardwaredefekten.
*1 8000 92 25                          ROM-Version -03 1571
*0 8000 98 75                          ROM-Version -01 1570
------------------------------------------------------------------------------
8002                                   Copyright-Statement.
8002 53 2f 57 20 2d 20 44 41           s/w - da
800a 56 49 44 20 47 20 53 49           vid g si
8012 52 41 43 55 53 41 0d 48           racusa h
801a 2f 57 20 2d 20 47 52 45           /w - gre
8022 47 20 42 45 52 4c 49 4e           g berlin
802a 0d 31 39 38 35 0d                  1985
------------------------------------------------------------------------------
8030                                   Routine wird vom U0-Befehl aufgerufen und ist für die
                                       weitere Bearbeitung der zusätzlichen U0-Befehle
                                       verantwortlich.
8030   ad  74  02    lda   $0274       Länge des Befehlsstrings
8033   c9  03        cmp   #$03        kleiner als 3?
8035   90  2e        bcc   $8065       verzweige, wenn ja
8037   ad  02  02    lda   $0202       drittes Zeichen aus Befehlsstring
803a   85  3b        sta   $3b         in Zwischenspeicher für weitere Bearbeitung
803c   29  1f        and   #$1f        Bits 0 bis 4 isolieren; bestimmen den Befehl
803e   aa            tax               als Index
803f   0a            asl               mal zwei
8040   a8            tay               als Adressindex
8041   b9  8e  80    lda   $808e,y     Adresse Lo des Befehls holen
8044   85  75        sta   $75         und setzen
8046   b9  8f  80    lda   $808f,y     Adresse Hi des Befehls holen
8049   85  76        sta   $76         und setzen
804b   e0  1e        cpx   #$1e        Befehle der 1571 erlaubt?
804d   f0  07        beq   $8056       Befehl ausführen, wenn ja
804f   ad  0f  18    lda   $180f       sonst auf 1541-Modus
8052   29  20        and   #$20        prüfen
8054   f0  0f        beq   $8065       verzweige, wenn Floppy im 1541-Modus
8056   a5  37        lda   $37         unsinnige Befehlsfolge:
8058   29  eb        and   #$eb        Ausblenden der Bits 2 und 4 ohne
805a   85  37        sta   $37         Belang!
805c   bd  6e  80    lda   $806e,x     Jobcode für Befehlsausführung holen
805f   8d  02  02    sta   $0202       und setzen
8062   6c  75  00    jmp   ($0075)     Befehl ausführen
------------------------------------------------------------------------------
8065                                   USER-Befehlsadresse für 1541-Modus setzen.
8065   a9  ea        lda   #$ea        Adresse Lo
8067   85  6b        sta   $6b         setzen
8069   a9  ff        lda   #$ff        Adresse Hi
806b   85  6c        sta   $6c         setzen; ergibt $ffea (USER-Befehlstabelle)
806d   60            rts               Ende
```

```
------------------------------------------------------------------------
806e                                Jobcodes für die einzelnen Befehle des U0-Kom- plexes. Die
                                    Jobcodes bedeuten im einzelnen:
                                    00/01 - kein Job erforderlich
                                    80/81 - Lesen eines Sektors
                                    88/89 - Lesen eines Sektors der gleichen Spur
                                    90/91 - Schreiben eines Sektors
                                    a0/a1 - Verify eines Sektors
                                    b0/b1 - Suchen eines Sektors (SEEK)
                                    c0/c1 - Nullanschlag des Kopfes (BUMP)
                                    d0/d1 - Programm im Puffer ausführen
                                    e0/e1 - Jobprogramm im Puffer ausführen
                                    f0/f1 - Formatieren einer Diskette
                                    Von diesen Jobcodes finden sich nicht alle in der Tabelle.
                                    Diese Aufstellung beschreibt alle vorhandenen Jobcodes,
                                    wobei der Wert +1 jeweils für das Laufwerk 1 bestimmt ist,
                                    das in der 1570 und der 1571 natürlich nicht existiert.
806e 80 81 90 91 b0 b1 f0 f1 00 01 b0 01 00 01 00 01
807e 80 81 90 91 b0 b1 f0 f1 00 01 b0 01 00 01 00 80
------------------------------------------------------------------------
808e                                Adressen der einzelnen U0-Routinen. Die Bits der
                                    Befehlsnummer haben dabei folgende Bedeutung:
                                    Bit 0       - Drivenummer (0 oder 1 (n.v.))
                                    Bit 1 bis 3 - eigentliche Befehlskennung
                                    Bit 4       - Diskettenseite (nur MFM-Betrieb)
                                    Bit 5 bis 7 - verschiedene Funktionen

     Bytes    Befehlscode           Beschreibung
808e 71 83    $00                   $8371 - MFM-Sektor von Diskette lesen (Drive 0)
8090 7f 83    $01                   $837f - s.o. für Drive 1 ('DRIVE NOT READY')
8092 ec 83    $02                   $83ec - MFM-Sektor auf Diskette schreiben
8094 f8 83    $03                   $83f8 - s.o. für Drive 1 ('DRIVE NOT READY')
8096 8b 84    $04                   $848b - Diskette initialisieren (MFM und GCR)
8098 7f 83    $05                   $837f - s.o. für Drive 1 ('DRIVE NOT READY')
809a b7 84    $06                   $84b7 - Diskette formatieren (MFM und GCR)
809c b7 84    $07                   $84b7 - s.o. für Drive 1 ('DRIVE NOT READY')
809e f1 84    $08                   $84f1 - Sektorabstand neu setzen (für MFM)
80a0 f1 84    $09                   $84f1 - Sektorabstand neu setzen (für MFM)
80a2 17 85    $0a                   $8517 - Diskettenformat analysieren
80a4 7f 83    $0b                   $837f - s.o. für Drive 1 ('DRIVE NOT ADY')
80a6 6b 85    $0c                   $856b - Floppy-Status neu setzen
80a8 7f 83    $0d                   $837f - s.o. für Drive 1 ('DRIVE NOT READY')
80aa a5 85    $0e                   $85a5 - Befehl BACKUP; '31, SYNTAX ERROR'
80ac a5 85    $0f                   $85a5 - Befehl BACKUP; '31, SYNTAX ERROR'
80ae 71 83    $10                   $8371 - MFM-Sektor von Diskette lesen (Drive 0)
80b0 7f 83    $11                   $837f - s.o. für Drive 1 ('DRIVE NOT READY')
80b2 ec 83    $12                   $83ec - MFM-Sektor auf Diskette schreiben
80b4 f8 83    $13                   $83f8 - s.o. für Drive 1 ('DRIVE NOT READY')
80b6 8b 84    $14                   $848b - Diskette initialisieren (MFM und GCR)
80b8 7f 83    $15                   $837f - s.o. für Drive 1 ('DRIVE NOT READY')
80ba b7 84    $16                   $84b7 - Diskette formatieren (MFM und GCR)
80bc b7 84    $17                   $84b7 - s.o. für Drive 1 ('DRIVE NOT READY')
80be 6d 80    $18                   $806d - (RTS)
80c0 6d 80    $19                   $806d - (RTS)
80c2 17 85    $1a                   $8517 - Diskettenformat analysieren
80c4 7f 83    $1b                   $837f - s.o. für Drive 1 ('DRIVE NOT READY')
```

```
80c6 6d 80    $1c                   $806d - (RTS)
80c8 6d 80    $1d                   $806d - (RTS)
80ca e5 8f    $1e                   $8fe5 - Modus-Umschaltbefehle behandeln
80cc 80 90    $1f                   $9080 - schnelles Laden einer Datei aktivieren
--------------------------------------------------------------------------
80ce                                Bedienung des seriellen Bus nach Auftreten eines ATN vom
                                    Computer.
80ce   78          sei              Diskcontroller inaktivieren
80cf   a9  00      lda   #$00       Flags löschen:
80d1   85  7c      sta   $7c        Flag für ATN löschen
80d3   85  79      sta   $79        Flag für LISTEN löschen
80d5   85  7a      sta   $7a        Flag für TALK löschen
80d7   a2  45      ldx   #$45       Stackpointer
80d9   9a          txs              zurücksetzen
80da   20  b2  81  jsr   $81b2      seriellen Bus für Empfang setzen
80dd   a9  80      lda   #$80       Flag für EOI
80df   85  f8      sta   $f8        löschen
80e1   85  7d      sta   $7d        Flag für ATN-Modus setzen
80e3   20  b7  e9  jsr   $e9b7      CLOCK OUT Hi setzen
80e6   20  a5  e9  jsr   $e9a5      DATA OUT Lo setzen
80e9   ad  00  18  lda   $1800      Bus lesen
80ec   09  10      ora   #$10       Antwortleitung für ATN löschen
80ee   8d  00  18  sta   $1800      und auf Bus setzen
80f1   ad  00  18  lda   $1800      Bus wieder lesen
80f4   10  64      bpl   $815a      und auf ATN testen
80f6   29  04      and   #$04       ATN noch vorhanden: CLOCK IN testen
80f8   d0  f7      bne   $80f1      warten, bis CLOCK IN Hi wird
80fa   20  ca  82  jsr   $82ca      Kommandobyte vom Bus holen
80fd   c9  3f      cmp   #$3f       UNLISTEN?
80ff   d0  0c      bne   $810d      verzweige, wenn nein
8101   a5  37      lda   $37        Busbetrieb wieder
8103   29  bf      and   #$bf       auf den langsamen
8105   85  37      sta   $37        1541 Bus stellen
8107   a9  00      lda   #$00       Flag für LI STEN
8109   85  79      sta   $79        löschen
810b   f0  0e      beq   $811b      unbedingter Sprung
810d   c9  5f      cmp   #$5f       UNTALK?
810f   d0  0d      bne   $811e      verzweige, wenn nein
8111   a5  37      lda   $37        Busbetrieb wieder
8113   29  bf      and   #$bf       auf den langsamen
8115   85  37      sta   $37        1541 Bus stellen
8117   a9  00      lda   #$00       Flag für TALK
8119   85  7a      sta   $7a        löschen
811b   4c  92  81  jmp   $8192      auf Ende des ATN warten
811e   c5  78      cmp   $78        TALK-Adresse?
8120   d0  0a      bne   $812c      verzweige, wenn nein
8122   a9  01      lda   #$01       Flag für TALK
8124   85  7a      sta   $7a        setzen
8126   a9  00      lda   #$00       Flag für LISTEN
8128   85  79      sta   $79        löschen
812a   f0  29      beq   $8155      unbedingter Sprung
812c   c5  77      cmp   $77        LISTEN-Adresse?
812e   d0  0a      bne   $813a      verzweige, wenn nein
8130   a9  01      lda   #$01       Flag für LISTEN
8132   85  79      sta   $79        setzen
8134   a9  00      lda   #$00       Flag für TALK
8136   85  7a      sta   $7a        löschen
```

```
8138   f0  1b       beq   $8155       unbedingter Sprung
813a   aa           tax               Kommandobyte merken
813b   29  60       and   #$60        Bit 5 und 6 testen
813d   c9  60       cmp   #$60        sind beide gesetzt?
813f   d0  4c       bne   $818d       verzweige, wenn nein
8141   8a           txa               Kommandobyte zurückholen
8142   85  84       sta   $84         und als Befehlssekundäradresse merken
8144   29  0f       and   #$0f        reine Sekundäradresse isolieren
8146   85  83       sta   $83         und setzen
8148   a5  84       lda   $84         Befehlssekundäradresse holen
814a   29  f0       and   #$f0        Kommandobits isolieren
814c   c9  e0       cmp   #$e0        CLOSE-Kommando?
814e   d0  42       bne   $8192       verzweige, wenn nein
8150   58           cli               Diskcontroller wieder aktivieren
8151   20  c0  da   jsr   $dac0       CLOSE-Routine; Kanal/File schließen
8154   78           sei               Diskcontroller inaktivieren
8155   2c  00  18   bit   $1800       ATN immer noch gesetzt?
8158   30  a0       bmi   $80fa       verzweige, wenn ja
815a   a9  00       lda   #$00        ATN-Modus ist
815c   85  7d       sta   $7d         beendet; Flag löschen
815e   ad  00  18   lda   $1800       Bus lesen
8161   29  ef       and   #$ef        Antwortsignal für Computer setzen
8163   8d  00  18   sta   $1800       und senden
8166   a5  79       lda   $79         Flag für LISTEN gesetzt?
8168   f0  0d       beq   $8177       verzweige, wenn nein
816a   24  37       bit   $37         ist der Bus im 1541-Modus?
816c   50  03       bvc   $8171       verzweige, wenn ja
816e   20  99  81   jsr   $8199       sonst schnellen Busbetrieb aktivieren
8171   20  42  83   jsr   $8342       Byte vom Bus holen
8174   4c  6b  83   jmp   $836b       zurück zur Systemwarteschleife
8177   a5  7a       lda   $7a         Flag für TALK gesetzt?
8179   f0  0f       beq   $818a       verzweige, wenn nein
817b   20  9c  e9   jsr   $e99c       DATA OUT-Leitung auf Hi setzen
817e   20  ae  e9   jsr   $e9ae       CLOCK OUT-Leitung auf Lo setzen
8181   20  83  a4   jsr   $a483       Verzögerung von ca. 70 Taktzyklen
8184   20  ea  81   jsr   $81ea       Daten auf Bus ausgeben
8187   20  83  a4   jsr   $a483       Verzögerung von ca. 70 Taktzyklen
818a   4c  66  83   jmp   $8366       zur Systemwarteschleife
818d   a9  10       lda   #$10        alle Leitungen bis auf ATN ACK
818f   8d  00  18   sta   $1800       (ATN-Antwortleitung) auf Hi setzen
8192   2c  00  18   bit   $1800       Bus prüfen
8195   10  c3       bpl   $815a       verzweige, wenn kein ATN
8197   30  f9       bmi   $8192       sonst warten, bis ATN zurückgesetzt
--------------------------------------------------------------------------
8199                                  Routine teilt dem Computer mit, daß er mit dem schnellen
                                      seriellen Busbetrieb (1571-Modus) arbeiten kann. Zu diesem
                                      Zweck wird ein sogenanns DRF-Signal ('device request fast')
                                      an den Rechner gesendet, damit er weiß, daß die
                                      angeschlossene Floppy auf schnellen Busbetrieb eingerichtet
                                      ist.
8199   20  59  ea   jsr   $ea59       auf ATN-Signal prüfen
819c   20  c0  e9   jsr   $e9c0       CLOCK IN-Leitung prüfen
819f   29  04       and   #$04        und CLOCK IN Bit isolieren
81a1   d0  f6       bne   $8199       auf Reaktion des Computers warten
81a3   20  ce  81   jsr   $81ce       1571 Bus für Ausgabe bereitmachen
81a6   a9  00       lda   #$00        Byte als DRF-Signal
81a8   8d  0c  40   sta   $400c       in das serielle Schieberegister
```

```
81ab   a9  08       lda   #$08        und
81ad   2c  0d  40    bit   $400d       Status prüfen
81b0   f0  fb       beq   $81ad       warten, bis Byte ausgegeben wurde
-------------------------------------------------------------------------
81b2                                  Seriellen Bus für schnellen Busbetrieb auf Ein- gang
                                      schalten.
81b2   08           php               Prozessorflags merken
81b3   78           sei               Diskcontroller inaktivieren
81b4   ad  0e  40    lda   $400e       Kontrollregister lesen
81b7   29  bf       and   #$bf        serielles Schieberegister auf Eingang
81b9   8d  0e  40    sta   $400e       schalten
81bc   ad  0f  18    lda   $180f       Steuerregister für Buscontroller lesen
81bf   29  fd       and   #$fd        Bus auf Eingang schalten
81c1   8d  0f  18    sta   $180f       und Wert setzen
81c4   a9  84       lda   #$84        (müßte #$88 heißen)
81c6   8d  0d  40    sta   $400d       Flag für Schieberegister initialisieren
81c9   2c  0d  40    bit   $400d       Status löschen
81cc   28           plp               Prozessorflags zurückholen
81cd   60           rts               Ende
-------------------------------------------------------------------------
81ce                                  Seriellen Bus für schnellen Busbetrieb auf Aus- gang
                                      schalten.
81ce   08           php               Prozessorflags merken
81cf   78           sei               Diskcontroller inaktivieren
81d0   ad  0f  18    lda   $180f       Steuer register für Buscontroller lesen
81d3   09  02       ora   #$02        Bus auf Ausgang schalten
81d5   8d  0f  18    sta   $180f       Wert setzen
81d8   ad  0e  40    lda   $400e       Steuerregister lesen
81db   09  40       ora   #$40        serielles Schieberegister auf Ausgang
81dd   8d  0e  40    sta   $400e       schal ten
81e0   a9  08       lda   #$08        (müßte #$88 heißen)
81e2   8d  0d  40    sta   $400d       Flag für Schieberegister initialisieren
81e5   2c  0d  40    bit   $400d       Status löschen
81e8   28           plp               Prozessorflags zurückholen
81e9   60           rts               Ende
-------------------------------------------------------------------------
81ea                                  Routine zum Senden von Daten auf den Bus als Folge eines
                                      TALK-Kommandos vom Computer. Die analoge Routine dazu
                                      befindet sih bei $e909 und steuert diesen Betrieb für den
                                      1541-Modus.
81ea   78           sei               Diskcontroller inaktivieren
81eb   20  eb  d0    jsr   $d0eb       freien Kanal zum Lesen suchen
81ee   b0  06       bcs   $81f6       verzweige, wenn kein Kanal frei
81f0   a6  82       ldx   $82         Kanalnummer holen
81f2   b5  f2       lda   $f2,x       Kanalstatus prüfen
81f4   30  01       bmi   $81f7       verzweige, wenn Status ok
81f6   60           rts               Ende
81f7   20  59  ea    jsr   $ea59       auf ATN-Signal prüfen
81fa   20  c0  e9    jsr   $e9c0       warten, bis CLOCK IN Lo wird
81fd   29  01       and   #$01        Datenbit isolieren
81ff   08           php               und dessen ertigkeit merken
8200   20  b7  e9    jsr   $e9b7       CLOCK OUT-Leitung auf Hi setzen
8203   28           plp               Datenbit zurückholen
8204   f0  12       beq   $8218       verzweige, wenn Bit gleich 0
8206   20  59  ea    jsr   $ea59       auf ATN-Signal prüfen
8209   20  c0  e9    jsr   $e9c0       warten, bis CLOCK IN Lo wird
820c   29  01       and   #$01        Datenbit isolieren
820e   d0  f6       bne   $8206       verzweige, wenn Bit gleich 1
```

```
8210   a6  82       ldx   $82        Kanalnurtll1er
8212   b5  f2       lda   $f2,x      Kanalstatus holen
8214   29  08       and   #$08       auf EOI testen
8216   d0  14       bne   $822c      verzweige, wenn kein EOI

                                     Die folgenden Befehle senden das EOI zum Computer weiter:
8218   20  59  ea   jsr   $ea59      auf ATN-Signal prüfen
821b   20  c0  e9   jsr   $e9c0      EOI senden; DATA-Leitung prüfen
821e   29  01       and   #$01       Datenbit isolieren
8220   d0  f6       bne   $8218      auf Reaktion des Computers warten
8222   20  59  ea   jsr   $ea59      auf ATN-Signal prüfen
8225   20  c0  e9   jsr   $e9c0      DATA IN-Leitung prüfen
8228   29  01       and   #$01       Datenbit isolieren
822a   f0  f6       beq   $8222      verzweige, wenn Bit gleich 0
822c   20  ae  e9   jsr   $e9ae      CLOCK OUT-Leitung auf Lo setzen
822f   20  59  ea   jsr   $ea59      auf ATN-Signal prüfen
8232   20  c0  e9   jsr   $e9c0      DATA IN-Leitung prüfen
8235   29  01       and   #$01       Datenbit isolieren
8237   d0  f3       bne   $822c      verzweige, wenn Bit gleich 1
8239   24  37       bit   $37        für Busbetrieb testen
823b   50  39       bvc   $8276      verzweige, wenn Bus im 1541-Modus
-------------------------------------------------------------------------
823d                                 Ein Byte auf den schnellen Bus der 1571 ausgeben. Als Byte
                                     wird dabei das aktuelle Datenbyte des aktuellen Kanals
                                     ausgegeben-
823d   ad  0f  18   lda   $180f      Steuerregister des Buscontrollers
8240   09  02       ora   #$02       seriellen Bus auf Ausgang
8242   8d  0f  18   sta   $180f      schalten
8245   ad  0e  40   lda   $400e      Steuerregister lesen
8248   09  40       ora   #$40       serielles Schieberegister auf Ausgang
824a   8d  0e  40   sta   $400e      setzen
824d   2c  0d  40   bit   $400d      Flag für Schieberegister löschen
8250   a6  82       ldx   $82        Kanalnummer
8252   bd  3e  02   lda   $023e,x    aktuelles Datenbyte holen
8255   8d  0c  40   sta   $400c      und ins Schieberegister schreiben
8258   ad  0d  40   lda   $400d      warten, bis
825b   29  08       and   #$08       das Byte
825d   f0  f9       beq   $8258      auf den Bus ausgegeben worden ist
825f   ad  0e  40   lda   $400e      Steuerregister lesen
8262   29  bf       and   #$bf       serielles Schieberegiser auf Eingang
8264   8d  0e  40   sta   $400e      schalten
8267   ad  0f  18   lda   $180f      Steuerregister des Buscontrollers lesen
826a   29  fd       and   #$fd       und den Bus auf Eingang
826c   8d  0f  18   sta   $180f      schalten
826f   a9  84       lda   #$84       (müßte #$88 heißen)
8271   8d  0d  40   sta   $400d      Flag für serielles Schieberegister setzen
8274   d0  3c       bne   $82b2      unbedingter Sprung; Ende
-------------------------------------------------------------------------
8276                                 Routine gibt das aktuelle Datenbyte auf den Bus aus. Es
                                     handelt sich hierbei um eine Aus- gabe auf den langsamen
                                     Bus der 1541, wobei eine analoge Version der Routine bei
                                     $e958 zu finden ist.
8276   a9  08       lda   #$08       Zähler auf 8 Bits für seriellen Bus
8278   85  98       sta   $98        setzen
827a   20  c0  e9   jsr   $e9c0      Port lesen
827d   29  01       and   #$01       Datenbit isolieren
827f   d0  43       bne   $82c4      verzweige, wenn Bit gleich 1
```

```
8281   a6  82       ldx   $82         Kanalnummer
8283   bd  3e  02   lda   $023e,x     aktuelles Byte des Kanals holen
8286   6a           ror               nächstes Bit in Prozessorstatus schieben
8287   9d  3e  02   sta   $023e,x     und restliches Byte merken
828a   b0  05       bcs   $8291       verzweige, wenn Datenbit gleich 1
828c   20  a5  e9   jsr   $e9a5       DATA OUT-Leitung auf Lo setzen
828f   d0  03       bne   $8294       unbedingter Sprung
8291   20  9c  e9   jsr   $e99c       DATA OUT-Leitung auf Hi setzen
8294   20  7e  a4   jsr   $a47e       Verzögerung von ca. 30 Taktzyklen
8297   a5  23       lda   $23         auf Busverzögerung prüfen
8299   d0  e6       bne   $8281       verzweige, wenn keine Busverzögerung nötig
829b   20  83  a4   jsr   $a483       sonst ca. 70 Zyklen warten (für 1541)
829e   20  b7  e9   jsr   $e9b7       CLOCK OUT-Leitung auf Hi setzen
82a1   20  7e  a4   jsr   $a47e       Verzögerung von ca. 30 Taktzyklen
82a4   a5  23       lda   $23         auf Busverzögerung prüfen
82a6   d0  03       bne   $82ab       verzweige, wenn keine Busverzögerung nötig
82a8   20  83  a4   jsr   $a483       sonst ca. 70 Zyklen warten
82ab   20  fb  fe   jsr   $fefb       CLOCK OUT auf Lo und DATA OUT auf Hi setzen
82ae   c6  98       dec   $98         Zähler für Anzahl der Bits vermindern
82b0   d0  c8       bne   $827a       weitermachen, wenn noch Bits auszugeben sind
82b2   20  59  ea   jsr   $ea59       auf ATN-Signal prüfen
82b5   20  c0  e9   jsr   $e9c0       DATA IN-Leitung prüfen
82b8   29  01       and   #$01        Datenbit isolieren
82ba   f0  f6       beq   $82b2       verzweige, wenn Bit gleich 0
82bc   58           cli               Diskcontroller wieder aktivieren
82bd   20  aa  d3   jsr   $d3aa       nächstes Byte aus Puffer holen
82c0   78           sei               Diskcontroller inaktivieren
82c1   4c  f0  81   jmp   $81f0       und zur Ausgabe
82c4   4c  62  83   jmp   $8362       zurück zur Systemwarteschleife; Ende
--------------------------------------------------------------------------
82c7                                  Routine holt ein Datenbyte vom schnellen Bus der 1571 als
                                      Folge eines LISTEN-Kommandos vom Computer. Analog dazu ist
                                      die Routine bei $e9c9 im DOS.
82c7   2c  0d  40   bit   $400d       ICR löschen
82ca   a9  08       lda   #$08        Anzahl der Bits im Byte für seriellen Bus
82cc   85  98       sta   $98         setzen
82ce   20  59  ea   jsr   $ea59       auf ATN-Signal prüfen
82d1   20  c0  e9   jsr   $e9c0       CLOCK IN-Leitung prüfen
82d4   29  04       and   #$04        CLOCK IN-Bit isolieren
82d6   d0  f6       bne   $82ce       verzweige, wenn Bit gleich 1
82d8   20  9c  e9   jsr   $e99c       DATA OUT-Leitung Hi setzen
82db   a9  01       lda   #$01        DATA IN-Leitung
82dd   2c  00  18   bit   $1800       testen
82e0   d0  fb       bne   $82dd       verzweige, wenn Leitung gleich Lo
82e2   8d  05  18   sta   $1805       Timer auf 256 Taktzyklen
82e5   20  59  ea   jsr   $ea59       auf ATN-Signal prüfen
82e8   ad  0d  18   lda   $180d       Steuerregister lesen
82eb   29  40       and   #$40        Timer schon abgelaufen?
82ed   d0  09       bne   $82f8       verzweige, wenn ja
82ef   20  c0  e9   jsr   $e9c0       sonst CLOCK IN-Leitung prüfen
82f2   29  04       and   #$04        und Bit testen
82f4   f0  ef       beq   $82e5       verzweige, wenn CLOCK IN gleich Hi
82f6   d0  19       bne   $8311       unbedingter Sprung
82f8   20  a5  e9   jsr   $e9a5       DATA OUT-Leitung auf Lo setzen
82fb   a2  18       ldx   #$18        und eine
82fd   ca           dex               Verzögerung von ca. 120 Taktzyklen
82fe   d0  fd       bne   $82fd       abwarten
```

```
8300   20  9c  e9    jsr   $e99c       DATA OUT-Leitung Hi setzen
8303   20  59  ea    jsr   $ea59       auf ATN-Signal prüfen
8306   20  c0  e9    jsr   $e9c0       CLOCK IN-Leitung prüfen
8309   29  04        and   #$04        CLOCK IN-Bit isolieren
830b   f0  f6        beq   $8303       verzweige, wenn Bit gleich Null
830d   a9  00        lda   #$00        EOI-Signal
830f   85  f8        sta   $f8         setzen; Flag setzen
8311   ad  00  18    lda   $1800       Port des Buscontrollers lesen
8314   49  01        eor   #$01        Datenbit invertieren und
8316   aa            tax               merken
8317   ad  0d  40    lda   $400d       Status des seriellen Schieberegisters
831a   29  08        and   #$08        auf 'Byte empfangen' testen
831c   f0  08        beq   $8326       verzweige, wenn kein Byte empfangen
831e   ad  0c  40    lda   $400c       sonst Byte aus Register holen
8321   85  85        sta   $85         und merken
8323   4c  3c  83    jmp   $833c       zurück zur Systemwarteschleife; Ende
8326   8a            txa               Datenbit zurückholen
8327   4a            lsr               und ins Carry schieben
8328   29  02        and   #$02        CLOCK IN-Leitung prüfen
832a   d0  e5        bne   $8311       verzweige, wenn Lo
832c   66  85        ror   $85         Datenbit ins Datenbyte schieben
832e   20  59  ea    jsr   $ea59       auf ATN-Signal prüfen
8331   20  c0  e9    jsr   $e9c0       CLOCK IN-Leitung prüfen
8334   29  04        and   #$04        Bit isolieren
8336   f0  f6        beq   $832e       verzweige, wenn CLOCK IN gleich Hi ist
8338   c6  98        dec   $98         Zähler für Bits vermindern
833a   d0  d5        bne   $8311       verzweige, wenn noch Bits folgen
833c   20  a5  e9    jsr   $e9a5       DATA OUT-Leitung auf Lo setzen
833f   a5  85        lda   $85         Datenbyte holen
8341   60            rts               und Ende
--------------------------------------------------------------------------
8342                                   Daten vom schnellen Bus der 1571 holen und in den aktuellen
                                       Puffer schreiben. Die analoge Routine für den 1541-Modus
                                       befindet sich bei $ea2e.
8342   78            sei               Diskcontroller inaktivieren
8343   20  07  d1    jsr   $d107       freien Schreibkanal suchen
8346   b0  05        bcs   $834d       verzweige, wenn kein Kanal frei
8348   b5  f2        lda   $f2,x       analstatus holen
834a   6a            ror               testen, ob Kanal inaktiv
834b   b0  0b        bcs   $8358       verzweige, wenn Kanal aktiv
834d   a5  84        lda   $84         Befehlssekundäradresse
834f   29  f0        and   #$f0        Kommando isolieren
8351   c9  f0        cmp   #$f0        und mit OPEN vergleichen
8353   f0  03        beq   $8358       verzweige, wenn OPEN
8355   4c  66  83    jmp   $8366       sonst zurück zur Systemwarteschleife
8358   20  c7  82    jsr   $82c7       Byte vom Bus holen
835b   58            cli               Diskcontroller wieder aktivieren
835c   20  b7  cf    jsr   $cfb7       Byte in Puffer schreiben
835f   4c  42  83    jmp   $8342       und weitermachen
--------------------------------------------------------------------------
8362                                   Flags für Busbetrieb löschen; Bus zurücksetzen.
8362   a9  00        lda   #$00        Flags für Busbetrieb
8364   85  37        sta   $37         löschen
--------------------------------------------------------------------------
8366                                   Bus zurücksetzen.
8366   a9  00        lda   #$00        Register für seriellen Bus
8368   8d  00  18    sta   $1800       löschen
```

```
------------------------------------------------------------------------
836b                                    Bus auf Eingang schalten und zur Systemwarte- schleife
                                        zurückkehren.
836b    20  b2  81    jsr   $81b2       Bus auf Eingang für Empfang von Befehlen
836e    4c  e7  eb    jmp   $ebe7       zurück zur Systemwartescleife; Ende
------------------------------------------------------------------------
8371                                    BURST-READ-Kommando. Diese Routine wird vom U0.Befehl
                                        aufgerufen und liest einen MFM-Sektor von der Diskette.
                                        Dieser Sektor kann direkt zum Computer übertragen, oder nur
                                        in den Puffer der Floppy gelesen werden; oder es können
                                        auch nur Pufferdaten übertragen werden - von einem Sektor,
                                        der schon früher gelesen wurde.
8371    8d  4d  02    sta   $024d       Jobcode merken
8374    85  5f        sta   $5f         und noch einmal merken
8376    ad  0d  18    lda   $180d       testen, ob die Diskette
8379    4a            lsr               gewechselt wurde
837a    90  18        bcc   $8394       verzweige, wenn nein
837c    a2  0b        ldx   #$0b        Nummer für '29, DISK ID MISMATCH'
837e    2c            .byte $2c         nächsten Befehl überspringen
837f    a2  4f        ldx   #$4f        Nummer für '74, DRIVE NOT READY'
8381    20  e9  85    jsr   $85e9       Fehlernummer initialisieren
8384    20  81  85    jsr   $8581       Burst-Statusbyte auf Bus ausgeben
8387    e0  02        cpx   #$02        wurde Job ordnungsgemäß ausgeführt?
8389    b0  01        bcs   $838c       verzweige, wenn nein
838b    60            rts               Ende
838c    8a            txa               Nummer der Fehlermeldung
838d    29  0f        and   #$0f        isolieren
838f    a2  00        ldx   #$00        und mit Puffernummer 0
8391    4c  0a  e6    jmp   $e60a       ausgeben; Ende
------------------------------------------------------------------------
8394                                    Sektor von der Diskette lesen.
8394    20  ce  81    jsr   $81ce       Bus der 1571 auf Ausgang schalten
8397    24  5e        bit   $5e         Burst-Status testen; MFM-Diskette?
8399    10  05        bpl   $83a0       verzweige, wenn nein
839b    a9  09        lda   #$09        sonst
839d    4c  e6  86    jmp   $86e6       MFM-formatierten Sektor lesen
83a0    20  3d  c6    jsr   $c63d       GCR-Diskette initialisieren
83a3    58            cli               Diskcontroller aktivieren
83a4    a5  3b        lda   $3b         Befehlsbyte holen
83a6    29  20        and   #$20        sollen nur Daten aus dem Puffer geholt werden?
83a8    d0  26        bne   $83d0       verzweige, wenn ja
83aa    ad  03  02    lda   $0203       sonst viertes Zeichen aus Befehlsstring
83ad    85  06        sta   $06         als Tracknummer setzen
83af    ad  04  02    lda   $0204       fünftes Zeichen
83b2    85  07        sta   $07         als Sektornummr setzen
83b4    a2  00        ldx   #$00        als Puffernummer die 0 wählen
83b6    a5  5f        lda   $5f         Jobcode für Befehl holen
83b8    95  00        sta   $00,x       und an Diskcontroller übergeben
83ba    20  5e  86    jsr   $865e       Job ausführen
83bd    78            sei               Diskcontroller inaktivieren
83be    20  e9  85    jsr   $85e9       Burst-Status setzen
83c1    24  3b        bit   $3b         sollen Fehler ignoriert werden?
83c3    70  04        bvs   $83c9       verzweige, wenn ja
83c5    e0  02        cpx   #$02        sonst Rückmeldung des Jobs prüfen
83c7    b0  b8        bcs   $8381       verzweige, wenn Fehler
83c9    20  f9  85    jsr   $85f9       Burst-Statusbyte zum Computer
```

```
83cc   a5  3b       lda   $3b        Befehlsbyte; soll der Sektor gesendet werden?
83ce   30  0d       bmi   $83dd      verzweige, wenn nein
83d0   a0  00       ldy   #$00       sonst Index setzen
83d2   b9  00  03   lda   $0300,y    Byte aus Puffer holen
83d5   85  46       sta   $46        und zum Computer
83d7   20  f9  85   jsr   $85f9      senden
83da   c8           iny              nächstes Byte
83db   d0  f5       bne   $83d2      und weitermachen
83dd   ce  05  02   dec   $0205      Zahl der zu lesenden Sektoren minus 1
83e0   f0  06       beq   $83e8      Ende, wenn alle Sektoren gelesen
83e2   20  1e  86   jsr   $861e      sonst Nummer des nächsten Sektors holen
83e5   4c  a3  83   jmp   $83a3      und nächsten Sektor lesen
83e8   58           cli              Diskcontroller aktivieren
83e9   4c  af  85   jmp   $85af      Kopf auf neuen Track positionieren; Ende
-------------------------------------------------------------------------------
83ec                                 BURST-WRITE-Befehl. Schreiben eines Sektors aus dem Puffer
                                     auf die Diskette (MFM oder GCR).
83ec   8d  4d  02   sta   $024d      Jobcode merken
83ef   ad  0d  18   lda   $180d      testen ,ob die Diskette
83f2   4a           lsr              gewechselt wurde
83f3   90  0d       bcc   $8402      verzweige, wenn nein
83f5   a2  0b       ldx   #$0b       Nummer für '29, DISK ID MISMATCH'
83f7   2c           .byte $2c        nächsten Befehl überspringen
83f8   a2  4f       ldx   #$4f       Nummer für '74, DRIVE NOT READY'
83fa   86  46       stx   $46        setzen
83fc   a5  3b       lda   $3b        Befehlsbyte
83fe   09  08       ora   #$08       Fehlerflag
8400   85  3b       sta   $3b        setzen
8402   24  5e       bit   $5e        Burst-Status prüfen
8404   10  05       bpl   $840b      verzweige, wenn GCR-formatierte Diskette
8406   a9  0a       lda   #$0a       sonst
8408   4c  e6  86   jmp   $86e6      MFM-Sektor schreiben; Ende
840b   20  3d  c6   jsr   $c63d      GCR Diskette initialisieren
840e   a5  3b       lda   $3b        soll nur Pufferinhalt geschrieben werden?
8410   30  29       bmi   $843b      verzweige, wenn ja
8412   78           sei              sonst Diskcontroller inaktivieren
8413   a0  00       ldy   #$00       Pufferindex setzen
8415   ad  00  18   lda   $1800      Port für seriellen Bus lesen
8418   49  08       eor   #$08       CLOCK OUT-Leitung invertieren
841a   2c  0d  40   bit   $400d      Status für serielles Schieberegister löschen
841d   8d  00  18   sta   $1800      Bus neu setzen
8420   ad  00  18   lda   $1800      Bus lesen
8423   10  03       bpl   $8428      verzweige, wenn kein ATN
8425   20  59  ea   jsr   $ea59      auf ATN-Signal prüfen
8428   ad  0d  40   lda   $400d      wurde Byte im seriellen Schieberegister
842b   29  08       and   #$08       empfangen?
842d   f0  f1       beq   $8420      verzweige, wenn nein
842f   ad  0c  40   lda   $400c      Byte aus Register holen
8432   99  00  03   sta   $0300,y    und in Puffer schreiben
8435   c8           iny              nächstes Byte
8436   d0  dd       bne   $8415      und weitermachen
8438   20  b7  e9   jsr   $e9b7      CLOCK OUT-Leitung auf Hi setzen
843b   58           cli              Diskcontroller aktivieren
843c   a5  3b       lda   $3b        Befehlsbyte
843e   29  20       and   #$20       Daten nur in den Puffer schreiben?
8440   d0  37       bne   $8479      verzweige, wenn ja
8442   a5  3b       lda   $3b        Befehlsbyte
8444   29  08       and   #$08       (Bit 3 des Befehlsbytes muß immer 0 sein!)
```

```
8446   f0  05       beq   $844d       verzweige unbedingt
8448   a6  46       ldx   $46         Byte für Ausgabe
844a   4c  81  83    jmp   $8381       $8381 Burst-Statusbyte ausgeben; Ende
844d   ad  03  02    lda   $0203       viertes Zeichen aus Befehlsstring
8450   85  06       sta   $06         als Tracknummer setzen
8452   ad  04  02    lda   $0204       fünftes Zeichen aus Befehlsstring
8455   85  07       sta   $07         als Sektornummer setzen
8457   a2  00       ldx   #$00        Index für Puffer 0
8459   a9  90       lda   #$90        Jobcode für Schreiben
845b   95  00       sta   $00,x       an Diskcontroller übergeben
845d   20  5e  86    jsr   $865e       Job ausführen
8460   78           sei               Diskcontroller inaktivieren
8461   20  ce  81    jsr   $81ce       seriellen Bus auf Ausgang
8464   20  e9  85    jsr   $85e9       Burst-Status setzen
8467   20  f9  85    jsr   $85f9       und zum Computer schicken
846a   20  a0  86    jsr   $86a0       auf Reaktion des Computers warten
846d   20  b2  81    jsr   $81b2       seriellen Bus auf Eingang schalten
8470   58           cli               Diskcontroller aktivieren
8471   24  3b       bit   $3b         Befehlsbyte
8473   70  04       bvs   $8479       verzweige, wenn Fehler ignoriert werden sollen
8475   e0  02       cpx   #$02        sonst Rückmeldung prüfen
8477   b0  0b       bcs   $8484       verzweige, wenn Fehler
8479   ce  05  02    dec   $0205       Anzahl der Sektoren minus 1
847c   f0  09       beq   $8487       Ende, wenn alle Sektoren geschrieben
847e   20  1e  86    jsr   $861e       neue Sektornummer setzen
8481   4c  12  84    jmp   $8412       und Sektor schreiben
8484   4c  8c  83    jmp   $838c       Diskstatus bereitstellen; Ende
8487   58           cli               Diskcontroller aktivieren
8488   4c  af  85    jmp   $85af       Kopf auf neuen Track positionieren; Ende
-------------------------------------------------------------------------------
848b                                  INQUIRE-DISK-Befehl. Diese Routine initiali- siert eine
                                      beliebige Diskette im MFM oder GCR- Format, indem durch das
                                      Lesen eines Blockheaders das jeweilige Format festgestellt
                                      wird. Dieses Kommando ist nach jedem Diskettenwechsel
                                      aufzu- rufen, sonst ist ein 'DISK ID MISMATCH' Fehler die
                                      Folge des nächsten Disketten-Schreib- oder Lesezugriffs.
848b   ad  02  02    lda   $0202       drittes Zeichen aus Befehlsstring
848e   29  01       and   #$01        Drivenummer isolieren
8490   d0  20       bne   $84b2       bei 1; 'DRIVE NOT READY' ausgeben
8492   a9  01       lda   #$01        Flag für Diskettenwechsel
8494   8d  0d  18    sta   $180d       löschen
8497   a9  05       lda   #$05        und
8499   20  e6  86    jsr   $86e6       versuchen, einen MFM-Header zu lesen
849c   ae  b0  01    ldx   $01b0       Rückmeldung prüfen
849f   e0  02       cpx   #$02        Fehler?
84a1   90  11       bcc   $84b4       verzweige, wenn MFM-Header
84a3   a2  00       ldx   #$00        sonst Fehlermeldung
84a5   86  5e       stx   $5e         löschen
84a7   a9  b0       lda   #$b0        und versuchen, einen GCR-Header
84a9   8d  4d  02    sta   $024d       zu lesen
84ac   95  00       sta   $00,x       Job an Diskcontroller
84ae   20  5e  86    jsr   $865e       Job ausführen; Rückmeldung für Diskstatus in X
83b1   2c           .byte $2c         nächsten Befehl überspringen
83b2   a2  4f       ldx   #$4f        Nummer für '74, DRIVE NOT READY'
84b4   4c  81  83    jmp   $8381       Burst-Status zum Computer; Ende
```

```
------------------------------------------------------------------------
84b7                                    FORMAT-DISK-Befehl. Dieser Befehl formatiert eine Diskette
                                        im MFM oder im GCR-Format, je nach Angabe; allerdings ohne
                                        Directory.
84b7    ad  02  02      lda   $0202     drittes Zeichen aus Befehlsstring
84ba    29  01          and   #$01      Drivenummer isolieren
84bc    d0  2b          bne   $84e9     Drive 1: '74, DRIVE NOT READY' ausgeben; Ende
84be    ad  03  02      lda   $0203     MFM oder GCR-Format für Formatierung?
84c1    10  05          bpl   $84c8     verzweige, wenn GCR-Format
84c3    a9  08          lda   #$08      sonst
84c5    4c  e6  86      jmp   $86e6     Diskette im MFM-Format formatieren
84c8    a9  00          lda   #$00      GCR-Format: Fehlerflags
84ca    85  5e          sta   $5e       löschen
84cc    85  ff          sta   $ff       Drivestatus löschen
84ce    ad  04  02      lda   $0204     fünftes Zeichen aus Befehlsstring
84d1    85  12          sta   $12       als ID 1 merken
84d3    ad  05  02      lda   $0205     sechstes Zeichen aus Befehlsstring
84d6    85  13          sta   $13       als ID 2 merken
84d8    20  07  d3      jsr   $d307     alle Kanäle schließen
84db    a9  01          lda   #$01      Tracknummer 1
84dd    85  80          sta   $80       setzen
84df    a9  ff          lda   #$ff      Flag für Rückmeldung
84e1    8d  98  02      sta   $0298     setzen
*1 84e4 20  89  a9      jsr   $a989     Diskette im 1571-Format formatieren
*0 84e4 20  4d  aa      jsr   $aa4d     Diskette im 1570-Format formatieren
84e7    aa              tax             Rückmeldung nach X
83e8    2c              .byte $2c       nächsten Befehl überspringen
83e9    a2  4f          ldx   #$4f      Nummer für '74, DRIVE NOT READY'
84eb    20  e9  85      jsr   $85e9     Burst-Status für Ausgabe setzen
84ee    4c  87  83      jmp   $8387     und zum Computer; Ende
------------------------------------------------------------------------
84f1                                    SECTOR-INTERLEAVE-Befehl. Dieser Befehl holt einen Wert vom
                                        Computer oder sendet den Wert zum Computer, der den Abstand
                                        der Sek- toren beim Beschreiben angibt, da die Sektoren auf
                                        der Diskette nicht direkt hintereinander beschrieben
                                        werden. Ist dieser Wert bspw. 5, so wird auf einer Spur
                                        zuerst Sektor Null, dann Sektor 5, 10, 15 usw. beschrieben
                                        und ebenso werden sie wieder gelesen.
84f1    78              sei             Diskcontroller inaktivieren
84f2    24  3b          bit   $3b       Befehlsbyte; soll Interleave neu gesetzt werden?
84f4    10  0a          bpl   $8500     verzweige, wenn nein
84f6    20  ce  81      jsr   $81ce     Bus auf Ausgang schalten
84f9    a5  3c          lda   $3c       Sektorabstand (Interleave)
84fb    85  46          sta   $46       holen und setzen
84fd    4c  f9  85      jmp   $85f9     Byte zum Computer; Ende
8500    ae  74  02      ldx   $0274     Länge des Befehlsstrings
8503    e0  04          cpx   #$04      größer gleich 4?
8505    b0  0a          bcs   $8511     verzweige, wenn ja
8507    a2  0e          ldx   #$0e      Nummer für SYNTAX ERROR
8509    20  e9  85      jsr   $85e9     Burst-Status entsprechend setzen
850c    a9  31          lda   #$31      Nummer der Fehlermeldung
850e    4c  c8  c1      jmp   $c1c8     '31, SYNTAX ERROR' ausgeben ;Ende
8511    ad  03  02      lda   $0203     viertes Zeichen aus Befehlsstring
8514    85  3c          sta   $3c       als neuen Sektorabstand setzen
8516    60              rts             Ende
```

```
------------------------------------------------------------------------
8517                                    QUERY-DISK-Befehl. Dieser Befehl erlaubt eine ziemlich
                                        genaue Analyse einer eingelegten Diskette, wobei die Floppy
                                        folgende Rückmeldungen bringt:
                                        - Burst-Status; war die Diskette GCR-formatiert, so ist
                                        dies die einzige Meldung,
                                        - Anzahl der Sektoren des jeweiligen Tracks,
                                        - Nummer des Tracks, der analysiert wurde,
                                        - kleinste Sektornummer dieses Tracks,
                                        - größte Sektornummer dieses Tracks,
                                        - Sektorabstand beim Beschreiben (Interleave).
8517   20  8b  84    jsr   $848b       INQUIRE-DISK aufrufen; Diskette initialisieren
851a   24  5e        bit   $5e         Burst-Status testen
851c   10  48        bpl   $8566       Ende, wenn Diskette im GCR-Format
851e   a9  0d        lda   #$0d        sonst
8520   20  e6  86    jsr   $86e6       Sektorfolge des Tracks ermitteln
8523   ae  b0  01    ldx   $01b0       Rückmeldung
8526   e0  02        cpx   #$02        Fehler aufgetren?
8528   b0  08        bcs   $8532       verzweige, wenn ja
852a   20  61  89    jsr   $8961       Nummer des kleinsten und größten Sektors holen
852d   20  86  89    jsr   $8986       Sektorabstand (Interleave) holen
8530   8a            txa               und
8531   48            pha               merken
8532   78            sei               Diskcontroller inaktivieren
8533   20  ce  81    jsr   $81ce       Bus auf Ausgang schalten
8536   a5  5e        lda   $5e         Burst-Statusbyte
8538   85  46        sta   $46         zum
853a   20  f9  85    jsr   $85f9       Computer schicken
853d   ae  b0  01    ldx   $01b0       Rückmeldung des Jobs
8540   e0  02        cpx   #$02        Fehler aufgetreten?
8542   b0  23        bcs   $8567       verzweige, wenn ja
8544   a5  97        lda   $97         sonst Anzahl der Sektoren des Tracks
8546   85  46        sta   $46         zum
8548   20  f9  85    jsr   $85f9       Computer schicken
854b   a5  67        lda   $67         aktuelle Tracknummer
854d   85  46        sta   $46         zum
854f   20  f9  85    jsr   $85f9       Computer schicken
8552   a5  60        lda   $60         kleinste Sektornummer des Tracks
8554   85  46        sta   $46         zum
8556   20  f9  85    jsr   $85f9       Computer schicken
8559   a5  61        lda   $61         größte Sektornummer
855b   85  46        sta   $46         zum
855d   20  f9  85    jsr   $85f9       Computer schicken
8560   68            pla               Sektorabstand (Interleave) zurückholen
8561   85  46        sta   $46         und zum
8563   20  f9  85    jsr   $85f9       Computer schicken
8566   60            rts               Ende
8567   68            pla               Stack b Fehler wieder herstellen
8568   4c  8c  83    jmp   $838c       Fehlermeldung ausgeben; Ende
------------------------------------------------------------------------
856b                                    INQUIRE-STATUS-Befehl. Diese Routine gibt den aktuellen
                                        Status an den Computer aus oder setzt einen vom Computer
                                        kommenden _ert als neuen Burst-Status fest.
856b   24  3b        bit   $3b         Befehlsbyte; soll Status neu gesetzt werden?
856d   10  27        bpl   $8596       verzweige, wenn nein
856f   24  3b        bit   $3b         soll auf Diskettenwechsel geprüft werden?
```

```
8571   50  0e       bvc   $8581       verzweige, wenn nein
8573   ad  0d  18   lda   $180d       sonst Steuerport lesen
8576   4a           lsr               und auf Diskettenwechsel prüfen
8577   90  08       bcc   $8581       verzweige, wenn kein Diskettenwechsel
8579   a5  5e       lda   $5e         Burst-Statusbyte holen
857b   29  f0       and   #$f0        und Status isolieren
857d   09  0b       ora   #$0b        Nummer für 'DISK ID MISMATCH' setzen
857f   85  5e       sta   $5e         und Burst-Statusbyte neu setzen...
-------------------------------------------------------------------------------
8581                                  Burst-Statusbyte an den Computer ausgeben.
8581   78           sei               Diskcontroller inaktivieren
8582   20  ce  81   jsr   $81ce       Bus auf Ausgang schalten
8585   a5  5e       lda   $5e         Burst-Statusbyte holen
8587   85  46       sta   $46         und zum
8589   20  f9  85   jsr   $85f9       Computer schicken
858c   a9  00       lda   #$00        Flag für Feh ler
858e   8d  6c  02   sta   $026c       löschen
8591   20  b2  81   jsr   $81b2       Bus wieder auf Eingang schalten
8594   58           cli               Diskcontroller wieder aktivieren
8595   60           rts               Ende
-------------------------------------------------------------------------------
8596                                  Burst-Statubyte mit ert vom Computer setzen.
8596   ad  03  02   lda   $0203       viertes Zeichen aus Befehlsstring
8599   85  5e       sta   $5e         als Status setzen
859b   24  3b       bit   $3b         Befehlsbyte: Diskettenwechsel löschen?
859d   50  05       bvc   $85a4       verzweige, wenn nein
859f   a9  01       lda   #$01        Flag für Diskettenwechsel
85a1   8d  0d  18   sta   $180d       löschen
85a4   60           rts               Ende
-------------------------------------------------------------------------------
85a5                                  BACKUP-Befehl. In der 1571 nicht verwendet!
85a5   a2  0e       ldx   #$0e        Nummer für 'SYNTAX ERROR'
85a7   20  e9  85   jsr   $85e9       Burst-Statusbyte setzen
85aa   a9  31       lda   #$31        Nummer der Fehlermeldung
85ac   4c  c8  c1   jmp   $c1c8       '31, SYNTAX ERROR' ausgeben; Ende
-------------------------------------------------------------------------------
85af                                  Kopf auf einen gewählten Track positionieren.
85af   ad  74  02   lda   $0274       Länge des Befehlsstrings
85b2   c9  07       cmp   #$07        kleiner 7?
85b4   90  32       bcc   $85e8       verzweige, wenn ja
85b6   a5  06       lda   $06         Tracknummer des letzten Jobs
85b8   a8           tay               merken
85b9   e9  01       sbc   #$01        minus 1
85bb   0a           asl               mal 2
85bc   85  64       sta   $64         als Anzahl der Schritte für Steppermotor
85be   c0  24       cpy   #$24        letzter Track auf zweiter Diskettenseite?
85c0   08           php               Antwort merken
85c1   ac  06  02   ldy   $0206       siebtes Zeichen aus Befehlsstring
85c4   84  22       sty   $22         als Zieltrack für Positionierung setzen
85c6   88           dey               minus 1
85c7   84  67       sty   $67         setzen
85c9   c0  23       cpy   #$23        mit 35 vergleichen; Spur auf 2. Diskettenseite?
85cb   6a           ror               Antwort in Bit 7
85cc   28           plp               vorherige Antwort zurückholen
85cd   29  80       and   #$80        jetzige Antwort isolieren
85cf   90  0b       bcc   $85dc       verzweige, wenn letzte Spur auf Seite 1
85d1   30  12       bmi   $85e5       verzweige, wenn jetzige Spur auf Seite 2
85d3   18           clc               letzte Spur auf Seite 2, neue auf Seite 1
```

```
85d4   a5  67       lda   $67         neue Tracknummer
85d6   69  23       adc   #$23        plus 35 ergibt
85d8   85  67       sta   $67         Trackäquivalent für die gleiche Seite
85da   30  09       bmi   $85e5       $85e5 verzweige unbedingt (s.u.)
85dc   10  07       bpl   $85e5       verzweige
85de   38           sec               neue Spur ist auf Seite 2 und muß auf
85df   a5  67       lda   $67         Seite 1 umgerechnet werden, um das
85e1   e9  23       sbc   #$23        Äquivalent für die Kopfbewegung zu
85e3   85  67       sta   $67         erhalten
85e5   4c  ba  87    jmp   $87ba       Kopf auf Track positionieren; Ende
85e8   60           rts               Ende
--------------------------------------------------------------------------
85e9                                  Burst-Statusbyte für die Ausgae zum Computer vorbereiten.
85e9   86  46       stx   $46         Fehlernummer merken
85eb   a5  5e       lda   $5e         Status holen
85ed   29  f0       and   #$f0        Statusflags isolieren
85ef   05  46       ora   $46         und mit Fehlernummer verknüpfen
85f1   85  5e       sta   $5e         neuen Status setzen
85f3   85  46       sta   $46         und für Ausgabe bereitmachen
85f5   60           rts               Ende
--------------------------------------------------------------------------
85f6                                  Ein Byte im schnellen Busmodus zum Computer senden. Das
                                      Byte muß dabei in $46 enthalten seIn.
85f6   20  59  ea    jsr   $ea59       auf ATN-Signal prüfen
85f9   ad  00  18    lda   $1800       Bus lesen
85fc   cd  00  18    cmp   $1800       und warten, bis
85ff   d0  f8        bne   $85f9       konstanter Wert anliegt
8601   29  ff        and   #$ff        auf ATN prüfen
8603   30  f1        bmi   $85f6       $85f6 verzweige, wenn gesetzt
8605   45  37        eor   $37         mit Busflags verknüpfen
8607   29  04        and   #$04        und CLOCK invertieren
8609   f0  ee        beq   $85f9       verzweige, wenn CLOCK IN Hi ist
860b   a5  46        lda   $46         Byte für Ausgabe
860d   8d  0c  40    sta   $400c       ins serielle Schieberegister
8610   a5  37        lda   $37         Flags für Busbetrieb
8612   49  04        eor   #$04        CLOCK-Bit invertieren
8614   85  37        sta   $37         und setzen
8616   a9  08        lda   #$08        Flag für serielles Schieberegister
8618   2c  0d  40    bit   $400d       testen
861b   f0  fb        beq   $8618       warten, bis Byte ausgegeben wurde
861d   60           rts               Ende
--------------------------------------------------------------------------
861e                                  Bei einem Befehl mit mehreren Sektoren wird hier die Nummer
                                      des darauffolgenden Sektors errechnet.
861e   ad  03  02    lda   $0203       Tracknummer aus Befehlsstring
8621   c9  24        cmp   #$24        Track auf Seite 1 der Diskette?
8623   90  02        bcc   $8627       verzweige, wenn nein (Track auf Seite 0)
8625   e9  23        sbc   #$23        sonst 35 abziehen
8627   aa           tax               und 'reinen' Track als Index nehmen
8628   bd  2b  94    lda   $942b,x     Anzahl der Sektoren des Tracks holen
862b   aa           tax               und merken
862c   ca           dex               minus 1 gibt höchste Sektornummer
862d   86  46        stx   $46         setzen
862f   18           clc               Flag für Addition löschen
8630   ad  04  02    lda   $0204       Sektornummer aus Befehlsstring
```

```
8633   65  3c       adc   $3c         plus Sektorabstand (Interleave)
8635   c5  46       cmp   $46         mit neuer Nummer vergleichen
8637   90  0a       bcc   $8643       verzweige, wenn kleiner
8639   e5  46       sbc   $46         sonst neue Nummer abziehen
863b   f0  04       beq   $8641       verzweige, wenn gleich Null
863d   38           sec               Flag setzen für Subtraktion
863e   e9  01       sbc   #$01        Sektornummer minus 1
8640   2c           .byte $2c         nächsten Befehl überspringen
8641   a5  46       lda   $46         maximale Sektornummer zurückholen
8643   8d  04  02   sta   $0204       merken
8646   a9  88       lda   #$88        Jobcode für Block lesen
8648   85  5f       sta   $5f         setzen
864a   60           rts               Ende
--------------------------------------------------------------------------
864b                                  Job ausführen; bei Fehler wird die Ausführung noch ein
                                      zweites Mal versucht.
864b   a6  f9       ldx   $f9         Puffernummer
864d   08           php               Prozessorstatus merken
864e   58           cli               Diskcontroller aktivieren
864f   20  b6  9f   jsr   $9fb6       Job ausführen
8652   c9  02       cmp   #$02        Fehler aufgetreten?
8654   90  05       bcc   $865b       verzweige, wenn nein
8656   20  83  86   jsr   $8683       sonst weiter versuchen
8659   b5  00       lda   $00,x       Rückmeldung holen
865b   aa           tax               und merken
865c   28           plp               Prozessorstatus holen: Controller inaktiv
865d   60           rts               Ende
--------------------------------------------------------------------------
865e                                  LED am Laufwerk einschalten; Job ausführen; LED
                                      ausschalten. Bei Fehler wird noch öfter versucht, den Job
                                      auszuführen.
865e   a2  00       ldx   #$00        Puffernummer
8660   08           php               Prozessorstatus retten
8661   78           sei               Diskcontroller inaktivieren
8662   ad  00  1c   lda   $1c00       Steuerport des Diskcontrollers
8665   09  08       ora   #$08        LED-Bit setzen
8667   8d  00  1c   sta   $1c00       LED einschalten
866a   58           cli               Diskcontroller aktivieren
866b   20  b6  9f   jsr   $9fb6       Job ausführen
866e   c9  02       cmp   #$02        Fehler aufgetreten?
8670   90  03       bcc   $8675       verzweige, wenn nein
8672   20  83  86   jsr   $8683       sonst nochmal versuchen
8675   78           sei               Diskcontroller inaktivieren
8676   ad  00  1c   lda   $1c00       Steuerport des Diskcontrollers
8679   29  f7       and   #$f7        LED-Bit löschen
867b   8d  00  1c   sta   $1c00       LED ausschalten
867e   b5  00       lda   $00,x       Rückmeldung holen
8680   aa           tax               und merken
8681   28           plp               Prozessorstatus zurückholen
8682   60           rts               Ende
--------------------------------------------------------------------------
8683                                  Einen weiteren Versuch starten, einen Job nach
                                      aufgetretenem Fehler noch auszuführen.
8683   a9  ff       lda   #$ff        Flag für Job
8685   8d  98  02   sta   $0298       setzen
8688   86  f9       stx   $f9         Puffernummer
868a   ad  02  02   lda   $0202       drittes Zeichen aus Befehlsstring (Jobcode)
868d   85  5f       sta   $5f         holen und setzen
```

```
868f    8d  4d  02    sta   $024d       setzen
8692    9d  5b  02    sta   $025b,x     setzen
8695    85  00        sta   $00         an Diskcontroller übergeben
8697    20  b6  9f    jsr   $9fb6       Job ausführen
869a    4c  99  d5    jmp   $d599       und Ende der Ausführung abwarten; Ende
--------------------------------------------------------------------------
869d                                    Routine wartet auf Reaktion des Computers.
869d    20  59  ea    jsr   $ea59       auf ATN-Signal prüfen
86a0    ad  00  18    lda   $1800       Bus lesen
86a3    cd  00  18    cmp   $1800       auf konstanten Wert
86a6    d0  f8        bne   $86a0       warten
86a8    29  ff        and   #$ff        auf ATN-Leitung testen
86aa    30  f1        bmi   $869d       verzweige, wenn ATN aktiv
86ac    45  37        eor   $37         sonst mit Flags verknüpfen
86ae    29  04        and   #$04        CLOCK-Bit isolieren
86b0    f0  ee        beq   $86a0       verzweige, wenn CLOCK IN gleich Hi ist
86b2    a5  37        lda   $37         Flags für Busbetrieb
86b4    49  04        eor   #$04        CLOCK-Bit invertieren
86b6    85  37        sta   $37         und Flags wieder setzen
86b8    60            rts               Ende
--------------------------------------------------------------------------
86b9                                    Bytewerte für die Befehlsausführung in der
                                        Hauptsteuerroutine für Diskettenzugriffe in der 1571. Diese
                                        Bytes bestehen aus Bitmustern mit den folgenden Aufgaben
                                        (Bit=1 heißt 'wird ausgeführt'):
                                        Bit 0 - Ende ohne Fehlerbehandlung
                                        Bit 1 - Kopf anhand jetziger Position setzen
                                        Bit 2 - Warten, bis Drive bereit für Zugriff
                                        Bit 3 - Kopf auf gewünschten Track positionieren
                                        Bit 4 - Drivemotor anlaufen lassen
                                        Bit 5 - Auf Schreibschutz testen
                                        Bit 6 - Sektorabstand/Sektornummer holen
                                        Bit 7 - Startsektor/Tracknummer holen
86b9 00   %00000000
86ba 15   %00010101
86bb 00   %00000000
86bc 00   %00000000
86bd 00   %00000000
86be 15   %00010101
86bf 00   %00000000
86c0 bc   %10111100
86c1 34   %00110100
86c2 de   %11011110
86c3 fe   %11111110
86c4 dc   %11011100
86c5 15   %00010101
86c6 15   %00010101
86c7 00   %00000000
--------------------------------------------------------------------------
86c8                                    Adressen der Routinen, die von der Hauptsteuerroutine
                                        aufgerufen werden.
86c8 ec 89                              $89ec - RESET-Routine
86ca ef 89                              $89ef - BUMP; Kopf auf Track Null setzen
86cc fd 89                              $89fd - Schreibschutz prüfen
86ce 03 8a                              $8a03 - Parameter für Track übernehmen
86dO 08 8a                              $8a08 - (RTS)
86d2 09 8a                              $8a09 - Blockheader lesen; SEEK
```

```
86d4 ba 87                              $87ba - (RTS)
86d6 86 8a                              $8a86 - Track im MFM-Format formatieren
86d8 57 8c                              $8c57 - Diskette im MFM-Format formatieren
86da 67 8d                              $8d67 - MFM-Sektor lesen
86dc f6 8d                              $8df6 - MFM-Sektor schreiben
86de c6 8e                              $8ec6 - MFM-Sektor verifizieren
86e0 18 8f                              $8f18 - MFM-Sektor auf Füllbyte verifizieren
86e2 5f 8f                              $8f5f - MFM-Sektorfolge ermitteln
86e4 b3 89                              $89b3 - MFM-Diskette initialisieren
-------------------------------------------------------------------------
86e6                                    Zentrale Steuerroutine des MFM-Betriebs der 1571. Diese
                                        Routine wird mit einem Wert im Akku angesprungen und holt
                                        je nach Bedarf die wichtigsten Parameter und macht das
                                        Laufwerk bereit.
86e6   78          sei                  Diskcontroller inaktivieren
86e7   48          pha                  Parameterwert merken
86e8   aa          tax                  und als Index
86e9   bd  b9  86   lda   $86b9,x        Bitmuster für Drivebetrieb holen
86ec   85  1b       sta   $1b           und merken
86ee   a5  5e       lda   $5e           Burst-Statusbyte lesen
86f0   09  80       ora   #$80          Flag für MFM-Betriebsart setzen
86f2   85  5e       sta   $5e           und Status wieder abspeichern
86f4   06  1b       asl   $1b           siebtes Steuerbit prüfen
86f6   90  05       bcc   $86fd         verzweige, wenn nicht gesetzt
86f8   ad  03  02   lda   $0203         sonst Tracknummer oder Startsektor holen
86fb   85  67       sta   $67           und setzen
86fd   06  1b       asl   $1b           sechstes Steuerbit prüfen
86ff   90  05       bcc   $8706         verzweige, wenn nicht gesetzt
8701   ad  04  02   lda   $0204         Sektorabstand oder Sektornummer holen
8704   85  43       sta   $43           und setzen
8706   06  1b       asl   $1b           fünftes Steuerbit prüfen
8708   90  11       bcc   $871b         verzweige, wenn nicht gesetzt
870a   ad  00  1c   lda   $1c00         auf Schreibschutz
870d   29  10       and   #$10          testen
870f   d0  0a       bne   $871b         verzweige, wenn kein Schreibschutz
8711   a5  3b       lda   $3b           sonst
8713   09  08       ora   #$08          Flag für Schreibschutz
8715   85  3b       sta   $3b           setzen
8717   a2  08       ldx   #$08          und nochmal
8719   86  46       stx   $46           setzen
871b   06  1b       asl   $1b           viertes Steuerbit prüfen
871d   90  03       bcc   $8722         verzweige, wenn nicht gesetzt
871f   20  94  87   jsr   $8794         sonst Drivemotor anlaufen lassen
8722   06  1b       asl   $1b           drittes Steuerbit prüfen
8724   90  03       bcc   $8729         verzweige, wenn nicht gesetzt
8726   20  ba  87   jsr   $87ba         Kopf auf Track positionieren
8729   06  1b       asl   $1b           zweites Steuerbit prüfen
872b   90  03       bcc   $8730         verzweige, wenn nicht gesetzt
872d   20  b0  87   jsr   $87b0         warten, bis Drive bereit für Zugriff
8730   20  54  89   jsr   $8954         Hardware für Diskseite setzen
8733   06  1b       asl   $1b           erstes Steuerbit prüfen
8735   90  03       bcc   $873a         verzweige, wenn nicht gesetzt
8737   20  2a  89   jsr   $892a         Sektorheader lesen; Kopf positionieren
873a   a9  00       lda   #$00          löschen (unsinnig)
873c   68          pla                  Steuerbyte zurückholen
873d   0a          asl                  mal 2
873e   aa          tax                  als Index in Adressentabelle
```

```
873f   bd  c8  86    lda   $86c8,x     Adresse Lo für Routine
8742   85  6f       sta   $6f          setzen
8744   bd  c9  86    lda   $86c9,x     Adresse Hi für Routine
8747   85  70       sta   $70          setzen
8749   20  61  87    jsr   $8761        Routine ausführen
874c   20  8f  f9    jsr   $f98f        Motor nach Verzögerung abschalten
874f   ae  b0  01    ldx   $01b0        Rückmeldung des letzten Jobs
8752   e0  02       cpx   #$02         Fehler aufgetreten?
8754   08           php                Antwort merken
8755   06  1b       asl   $1b          nulltes Steuerbit prüfen
8757   b0  06       bcs   $875f        verzweige, wenn gesetzt
8759   28           plp                sonst Fehlerstatus zurückholen
875a   90  04       bcc   $8760        verzweige, wenn kein Fehler
875c   4c  8c  83    jmp   $838c        sonst Fehler ausgeben; Ende
875f   28           plp                Fehlerstatus holen
8760   60           rts                Ende
--------------------------------------------------------------------------
8761                                   Routine ausführen, deren Startadresse in $6f/70 steht.
8761   6c  6f  00    jmp   ($006f)      Sprung auf Routine
--------------------------------------------------------------------------
8764                                   Drivemotor einschalten.
8764   08           php                Prozessorstatus retten
8765   78           sei                Diskcontroller inaktivieren
8766   ad  00  1c    lda   $1c00        Steuerport des Diskcontrollers
8769   09  04       ora   #$04        Motor-Bit setzen
876b   8d  00  1c    sta   $1c00        Motor einschalten
876e   28           plp                Controller aktivieren
876f   60           rts                Ende
--------------------------------------------------------------------------
8770                                   Drivemotor ausschalten.
8770   08           php                Prozessorstatus retten
8771   78           sei                Diskcontroller inaktivieren
8772   ad  00  1c    lda   $1c00        Steuerport des Diskcontrollers
8775   29  fb       and   #$fb         Motor-Bit löschen
8777   8d  00  1c    sta   $1c00        Drivemotor ausschalten
877a   28           plp                Prozessorstatus zurückholen
877b   60           rts                Ende
--------------------------------------------------------------------------
877c                                   LED am Laufwerk einschalten.
877c   08           php                Prozessorstatus retten
877d   78           sei                Diskcontroller inaktivieren
877e   ad  00  1c    lda   $1c00        Steuerport des Diskcontrollers
8781   09  08       ora   #$08        LED-Bit setzen
8783   8d  00  1c    sta   $1c00        LED einschalten
8786   28           plp                Prozessorstatus zurückholen
8787   60           rts                Ende
--------------------------------------------------------------------------
8788                                   LED am Laufwerk ausschalten.
8788   08           php                Prozessorstatus retten
8789   78           sei                Diskcontroller inaktivieren
878a   ad  00  1c    lda   $1c00        Steuerport des Diskcontrollers
878d   29  f7       and   #$f7         LED-Bit löschen
878f   8d  00  1c    sta   $1c00        LED ausschalten
8792   28           plp                Controller aktivieren
8793   60           rts                Ende
```

```
-------------------------------------------------------------------------
8794                                    Drivemotor einschalten und Werte für Verzöge- rung beim
                                        Hochlaufen setzen.
8794   08          php                  Prozessorstatus retten
8795   78          sei                  Diskcontroller inaktivieren
8796   a5  20      lda   $20            Drivestatus
8798   c9  20      cmp   #$20           Drivemotor an?
879a   f0  0e      beq   $87aa          verzweige, wenn ja
879c   ad  02  02  lda   $0202          Drivenummer aus Befehlsstring
879f   29  01      and   #$01           isolieren
87a1   85  3e      sta   $3e            und setzen
87a3   20  64  87  jsr   $8764          Drivemotor einschalten
87a6   a9  a0      lda   #$a0           Flag für Motor an aber noch nicht bereit
87a8   85  20      sta   $20            setzen
87aa   a9  32      lda   #$32           Verzögerungszähler für Hochlaufzeit des Motors
87ac   85  48      sta   $48            setzen
87ae   28          plp                  Prozessorstatus zurückholen
87af   60          rts                  Ende
-------------------------------------------------------------------------
87b0                                    Warten, bis das Drive bereit ist.
87b0   08          php                  Prozessorstatus retten
87b1   58          cli                  Diskcontroller aktivieren
87b2   a5  20      lda   $20            Flags für Drivestatus
87b4   c9  20      cmp   #$20           Motor schon an und bereit?
87b6   d0  fa      bne   $87b2          warten, wenn nein
87b8   28          plp                  Prozessorstatus zurückholen
87b9   60          rts                  Ende
-------------------------------------------------------------------------
87ba                                    Kopf auf gewählten Track positionieren.
87ba   08          php                  Prozessorstatus retten
87bb   58          cli                  Diskcontroller aktivieren
87bc   a5  67      lda   $67            neue Tracknummer
87be   0a          asl                  mal 2
87bf   c5  64      cmp   $64            gleich Anzahl der berechneten Steps?
87c1   f0  1a      beq   $87dd          verzweige, wenn ja
87c3   a5  67      lda   $67            Tracknurrvner
87c5   0a          asl                  mal 2
87c6   c5  64      cmp   $64            gleich Anzahl der berechneten Steps?
87c8   f0  0e      beq   $87d8          verzweige, wenn ja
87ca   b0  06      bcs   $87d2          verzweige, wenn größer
87cc   20  e7  87  jsr   $87e7          einen Step nach außen
87cf   4c  c3  87  jmp   $87c3          weiter...
87d2   20  df  87  jsr   $87df          einen Step nach innen
87d5   4c  c3  87  jmp   $87c3          weiter...
87d8   a0  12      ldy   #$12           13 ms Verzögerung
87da   20  29  88  jsr   $8829          abwarten
87dd   28          plp                  Prozessorstatus zurückholen
87de   60          rts                  Ende
-------------------------------------------------------------------------
87df                                    Kopf einen Step nach innen bewegen.
87df   a5  64      lda   $64            Anzahl der Steps
87e1   18          clc                  vermindern
87e2   69  01      adc   #$01           und wieder
87e4   4c  14  88  jmp   $8814          abspeichern; Kopf bewegen
-------------------------------------------------------------------------
87e7                                    Kopf einen Step nach außen bewegen (Richtung Track 0).
87e7   a0  63      ldy   #$63           99 Schleifendurchgänge als Zeit für einen Step
```

```
87e9   ad  0f  18    lda   $180f       Steuerport lesen
87ec   6a            ror               Bit für Nullanschlag ins Carry
87ed   08            php               und merken
87ee   ad  0f  18    lda   $180f       Steuerport lesen
87f1   6a            ror               Bit für Nullanschlag
87f2   6a            ror               nach Bit 7 schieben
87f3   28            plp               vorheriger Zustand
87f4   29  80        and   #$80        neuen Zustand testen
87f6   90  04        bcc   $87fc       verzweige, wenn beim 1. Test Spur 0 erreicht
87f8   10  15        bpl   $880f       verzweige, wenn beim 2. Test Spur 0 auch n.err.
87fa   30  02        bmi   $87fe       unbedingter Sprung
87fc   30  11        bmi   $880f       verzweige, wenn Spur 0 in beiden Versuchen err.
87fe   88            dey               sonst Zähler vermindern und weiter prüfen
87ff   d0  e8        bne   $87e9       während sich der Kopf bewegt
8801   b0  0c        bcs   $880f       verzweige, wenn Spur 0 nicht erreicht
8803   ad  00  1c    lda   $1c00       Steuerport für Diskcontroller
8806   29  03        and   #$03        Bits für Steppermotor isolieren
8808   d0  05        bne   $880f       verzweige, wenn Steppermotor läuft
880a   a9  00        lda   #$00        sonst Zahl der Steps
880c   85  64        sta   $64         löschen
880e   60            rts               Ende
880f   a5  64        lda   $64         Zahl der Steps
8811   38            sec               minus 1
8812   e9  01        sbc   #$01        rechnen und wieder
8814   85  64        sta   $64         setzen
8816   29  03        and   #$03        Ansteuerung für Steppermotor herstellen
8818   85  6f        sta   $6f         und merken
881a   08            php               Prozessorstatus merken
881b   78            sei               Diskcontroller inaktivieren
881c   ad  00  1c    lda   $1c00       Steuerport für Diskcontroller
881f   29  fc        and   #$fc        Bits für Stepper ausblenden
8821   05  6f        ora   $6f         neue Ansteuerung setzen
8823   8d  00  1c    sta   $1c00       und Motor steuern
8826   28            plp               Prozessorstatus zurückholen
*1 8827a0  06        ldy   #$06        ca. 8 ms Verzögerung
*0 8827a0  08        ldy   #$08        ca. 10 ms Verzögerung
8829   20  30  88    jsr   $8830       abwarten
882c   88            dey               Zähler vermindern
882d   d0  fa        bne   $8829       und warten
882f   60            rts               Ende
--------------------------------------------------------------------------
8830                                   Verzögerung von ca. 1.3 ms abwarten.
8830   a2  02        ldx   #$02        Zähler Hi
8832   a9  00        lda   #$00        Zähler Lo
8834   69  01        adc   #$01        Zähler Lo plus 1
8836   d0  fc        bne   $8834       256 mal
8838   ca            dex               Zähler Hi minus 1
8839   d0  f9        bne   $8834       und weitermachen
883b   60            rts               Ende
--------------------------------------------------------------------------
883c                                   Status des MFM-Controllers (WD 1770) holen.
883c   ea            nop               Pause
883d   ad  00  20    lda   $2000       Status holen
8840   4a            lsr               Bits 3 und 4
8841   4a            lsr               an die Positionen 1 und 2
8842   4a            lsr               schieben und
8843   29  03        and   #$03        isolieren
8845   aa            tax               dann nach X
```

```
8846   bd  82  8a    lda   $8a82,x     entsprechende Fehlermeldung holen
8849   8d  b0  01    sta   $01b0       und setzen
884c   aa            tax               Fehlernummer merken
884d   60            rts               Ende
-------------------------------------------------------------------------
884e                                   Kommando an MFM-Controller übergeben.
884e   48            pha               Kommando merken
884f   20  7c  87    jsr   $877c       lED am laufwerk einschalten
8852   68            pla               Kommando zurückholen
8853   8d  00  20    sta   $2000       und an Controller übergeben
8856   a9  01        lda   #$01        Wert für Test, ob Controller aktiv
8858   ea            nop               Pause
8859   2c  00  20    bit   $2000       testen, ob Controller den BefehL ausführt
885c   f0  fb        beq   $8859       verzweige, wenn nein; warten
885e   4c  7e  a4    jmp   $a47e       sonst 45 Taktzyklen abwarten; Ende
-------------------------------------------------------------------------
8861                                   Ausführung eines Befehls des MFM-ControLLers abwarten.
8861   20  88  87    jsr   $8788       LED am laufwerk ausschalten
8864   a9  01        lda   #$01        Wert für Test
8866   2c  00  20    bit   $2000       ist der Controller noch beschäftigt?
8869   d0  fb        bne   $8866       warten, wenn ja
886b   60            rts               Ende
-------------------------------------------------------------------------
886c                                   Nummer des folgenden Sektors berechnen, indem der gültige
                                       Sektorabstand (Interleave) addiert wird.
886c   a5  60        lda   $60         kleinste Sektornummer eines Tracks
886e   38            sec               minus 1
886f   e9  01        sbc   #$01        da Zählung von Null beginnt
8871   85  46        sta   $46         Wert merken
8873   ad  04  02    lda   $0204       Sektornummer aus Befehlsstring
8876   18            clc               und
8877   65  3c        adc   $3c         Sektorabstand addieren
8879   c5  61        cmp   $61         mit größter Sektornummer des Tracks vergleichen
887b   f0  07        beq   $8884       verzweige, wenn gleich
887d   90  05        bcc   $8884       verzweige, wenn kleiner
887f   e5  61        sbc   $61         sonst größte Nummer abziehen
8881   18            clc               und
8882   65  46        adc   $46         Offset der Sektorverschiebung addieren
8884   8d  04  02    sta   $0204       neuen Sektor merken
8887   60            rts               Ende
-------------------------------------------------------------------------
8888                                   Sektortabelle für Formatieren (MFM) erstellen.
8888   a0  00        ldy   #$00        Index setzen
888a   a2  00        ldx   #$00        und setzen
888c   ad  03  02    lda   $0203       Startsektor holen
888f   29  3f        and   #$3f        und auf 0 bis 63
8891   8d  03  02    sta   $0203       begrenzen
8894   85  60        sta   $60         und setzen
8896   48            pha               Sektornummer merken
8897   ad  07  02    lda   $0207       Nummer des letzten Sektors
889a   48            pha               ebenfalls merken
889b   ee  04  02    inc   $0204       Sektor-Interleave
889e   ad  03  02    lda   $0203       Sektornummer holen
88a1   99  0b  02    sta   $020b,y     und in Tabelle schreiben
88a4   ee  03  02    inc   $0203       Sektornummer erhöhen
88a7   e8            inx               Anzahl der Sektoren erhöhen
```

```
88a8   98           tya             Sektorposition
88a9   18           clc             holen und
88aa   6d  04  02    adc   $0204     Sektorabstand (Interleave) addieren
88ad   a8           tay             neuer Sektor
88ae   c0  20       cpy   #$20      kleiner als 32?
88b0   b0  0c       bcs   $88be      verzweige, wenn nein
88b2   cc  07  02    cpy   $0207     kleiner als der letzte Sektor?
88b5   90  1a       bcc   $88d1      verzweige, wenn ja
88b7   d0  12       bne   $88cb      verzweige, wenn ungleich dem letzten Sektor
88b9   ec  07  02    cpx   $0207     Anzahl der Sektoren erreicht?
88bc   f0  0d       beq   $88cb      verzweige, wenn ja
88be   ce  04  02    dec   $0204     Sektor-Interleave minus 1
88c1   68           pla             Nummer des letzten Sektors holen
88c2   8d  07  02    sta   $0207     und wieder merken
88c5   68           pla             kleinste Sektornummer zurückholen
88c6   8d  03  02    sta   $0203     und wieder merken
88c9   38           sec             Fehlerflag setzen
88ca   60           rts             Ende
88cb   98           tya             Zeiger auf aktuellen Sektor
88cc   38           sec             auf definierten
88cd   ed  07  02    sbc   $0207     Sektorbereich begrenzen
88d0   a8           tay             und wieder übernehmen
88d1   ec  07  02    cpx   $0207     schon letzter Sektor erreicht?
88d4   d0  c8       bne   $889e      verzweige, wenn nein
88d6   86  97       stx   $97        sonst Sektornummer merken
88d8   ca           dex             minus 1
88d9   8a           txa             ergibt
88da   18           clc             plus der Nummer
88db   65  60       adc   $60        des kleinsten Sektors
88dd   85  61       sta   $61        die Nummer des größten Sektors
88df   c5  60       cmp   $60        mit kleinster Nummer vergleichen
88e1   90  db       bcc   $88be      verzweige, wenn kleiner
88e3   68           pla             höchste Sektornummer zurückholen
88e4   8d  07  02    sta   $0207     und wieder setzen
88e7   68           pla             kleinste Nummer zurückholen
88e8   8d  03  02    sta   $0203     und wieder setzen
88eb   ce  04  02    dec   $0204     Sektor-/nterleave minus 1
88ee   18           clc             Flag für alles ok
88ef   60           rts             Ende
--------------------------------------------------------------------------
88f0                                 Verify eines MFM-Sektors beim Formatieren. Diese Routine
                                     kontrolliert die geschriebenen Sektoren auf einwandfreien
                                     leerinhalt.
88f0   ad  b0  01    lda   $01b0     Tracknummer beim Formatieren
88f3   48           pha             merken
88f4   a0  00       ldy   #$00      Zähler für Sektornummer
88f6   84  24       sty   $24       setzen
88f8   a4  24       ldy   $24       Zähler holen
88fa   b9  0b  02    lda   $020b,y   Sektornummer aus Tabelle holen
88fd   8d  02  20    sta   $2002     und an Controller
8900   20  18  8f    jsr   $8f18     Sektor auf leerinhalt überprüfen
8903   ae  b0  01    ldx   $01b0     Rückmeldung
8906   e0  02       cpx   #$02      Fehler aufgetreten?
8908   b0  0b       bcs   $8915      verzweige, wenn ja
890a   e6  24       inc   $24       sonst Zähler für Sektoren erhöhen
890c   a4  24       ldy   $24       und Index holen
890e   cc  07  02    cpy   $0207     schon letzter Sektor erreicht?
8911   d0  e5       bne   $88f8      weitermachen, wenn nein
```

```
8913   18           clc              Flag für alles ok setzen
8914   24           .byte $24        nächsten Befehl überspringen
8915   38           sec              Flag für Fehler setzen
8916   68           pla              aktuelle Tracknummer bei Formatieren wieder
8917   8d  b0  01    sta   $01b0      setzen
891a   60           rts              Ende
-------------------------------------------------------------------------
891b                                 Tracknummer aus Befehlszeile beim Formatieren holen.
891b   ad  74  02    lda   $0274      Länge des Befehlsstrings
891e   c9  07       cmp   #$07       kleiner 7?
8920   90  f8       bcc   $891a      verzweige, wenn ja
8922   ad  06  02    lda   $0206      sonst Tracknummer holen
8925   85  67       sta   $67        und setzen
8927   4c  ba  87    jmp   $87ba      Kopf positionieren; Ende
-------------------------------------------------------------------------
892a                                 Versuch, einen MFM-Sektorheader zu lesen; bei Mißerfolg
                                     wird es ein zweites Mal versucht.
892a   ad  b0  01    lda   $01b0      Rückmeldung
892d   48           pha              merken
892e   20  27  8a    jsr   $8a27      Sektorheader lesen
8931   ae  b0  01    ldx   $01b0      Rückmeldung
8934   e0  02       cpx   #$02       Feh ler?
8936   90  0d       bcc   $8945      verzweige, wenn nein
8938   20  ef  89    jsr   $89ef      Kopf neu positionieren
893b   20  27  8a    jsr   $8a27      Lesen noch einmal versuchen
893e   ae  b0  01    ldx   $01b0      Rückmeldung prüfen
8941   e0  02       cpx   #$02       Fehler?
8943   b0  0a       bcs   $894f      verzweige, wenn ja
8945   a5  67       lda   $67        sonst Tracknummer holen
8947   0a           asl              mal 2
8948   c5  64       cmp   $64        mit Anzahl der Schritte vergleichen
894a   f0  03       beq   $894f      verzweige, wenn gleich
894c   20  ba  87    jsr   $87ba      Kopf wieder positionieren
894f   68           pla              Rückmeldung
8950   8d  b0  01    sta   $01b0      wieder setzen
8953   60           rts              Ende
-------------------------------------------------------------------------
8954                                 Hardware der Floppy für Diskettenseite 0 oder 1
                                     entsprechend $3b setzen.
8954   08           php              Prozessorstatus retten
8955   78           sei              Diskcontroller inaktivieren
8956   a5  3b       lda   $3b        Befehlsbyte holen
8958   29  10       and   #$10       Diskseite isolieren
895a   c9  10       cmp   #$10       gleich Seite 1?
895c   20  f3  93    jsr   $93f3      Hardware auf die entsprechende Seite einstellen
895f   28           plp              Controller aktivieren
8960   60           rts              Ende
-------------------------------------------------------------------------
8961                                 Sektortabelle nach kleinster und größter Sek- tornummer
                                     durchsuchen.
8961   a4  97       ldy   $97        Anzahl der Sektoren pro Track
8963   88           dey              minus 1
8964   a9  ff       lda   #$ff       maximal möglichen Wert mit
8966   d9  0b  02    cmp   $020b,y    Tabelle vergleichen
8969   90  03       bcc   $896e      verzweige, wenn Tabellenwert kleiner
896b   b9  0b  02    lda   $020b,y    Tabellenwert laden
896e   88           dey              Zähler minus 1
```

```
896f   10  f5       bpl   $8966       weitermachen, bis alles geprüft
8971   85  60       sta   $60         kleinste Nummer merken
8973   a4  97       ldy   $97         Anzahl der Sektoren
8975   88           dey               minus 1
8976   a9  00       lda   #$00        kleinstmöglicher Zählwert
8978   d9  0b  02   cmp   $020b,y     mit Tabellenwert vergleichen
897b   b0  03       bcs   $8980       verzweige, wenn Tabellenwert größer
897d   b9  0b  02   lda   $020b,y     Tabellenwert laden
8980   88           dey               Zähler minus 1
8981   10  f5       bpl   $8978       weitermachen, bis alles geprüft
8983   85  61       sta   $61         größten Wert merken
8985   60           rts               Ende
--------------------------------------------------------------------------
8986                                  Abstände der Sektoren beim Schreiben und lesen (Interleave)
                                      aus der Sektorfolge berechnen.
8986   a6  97       ldx   $97         Anzahl der Sektoren
8988   a0  00       ldy   #$00        Index setzen
898a   b9  0b  02   lda   $020b,y     Sektornummer holen
898d   c5  60       cmp   $60         mit kleinster Nummer vergleichen
898f   f0  05       beq   $8996       verzweige, wenn gleich
8991   c8           iny               nächster Wert
8992   c4  97       cpy   $97         schon alles geprüft?
8994   d0  f4       bne   $898a       weitermachen, wenn nein
8996   84  5f       sty   $5f         Index des kleinsten Sektors merken
8998   a5  60       lda   $60         kleinste Sektornummer
899a   18           clc               plus 1
899b   69  01       adc   #$01        ergibt
899d   85  46       sta   $46         die Nummer des nächsten Sektors
899f   a2  ff       ldx   #$ff        Zähler für Maximalwert
89a1   b9  0b  02   lda   $020b,y     Sektornummer holen
89a4   c5  46       cmp   $46         mit zweitem Sektor vergleichen
89a6   f0  0a       beq   $89b2       verzweige, wenn gleich
89a8   e8           inx               Zähler erhöhen
89a9   c8           iny               Zeiger auf nächsten Wert
89aa   c4  97       cpy   $97         schon alles geprüft?
89ac   d0  f3       bne   $89a1       weitermachen, wenn nein
89ae   a0  00       ldy   #$00        Zeiger löschen
89b0   f0  ef       beq   $89a1       unbedingter Sprung
89b2   60           rts               Ende
--------------------------------------------------------------------------
89b3                                  MFM-Controller auf eingelegte Diskette einstel- len und auf
                                      einwandfreien Betrieb prüfen.
89b3   a5  6f       lda   $6f         Zwischenspeicher
89b5   48           pha               merken
89b6   08           php               Prozessorstatus merken
89b7   78           sei               Diskcontroller inaktivieren
89b8   ad  01  20    lda   $2001       'alte' Tracknummer
89bb   8d  03  20    sta   $2003       neu übernehmen
89be   a9  18       lda   #$18        Kommando 'Track suchen'
89c0   20  4e  88    jsr   $884e       an Controller
89c3   20  61  88    jsr   $8861       Ausführung abwarten
89c6   a2  00       ldx   #$00        Zähler für Zahl der
89c8   a0  80       ldy   #$80        Versuche auf 32768 setzen
89ca   ad  00  20    lda   $2000       Controller-Status lesen
89cd   29  02       and   #$02        Indexloch-lED isolieren
89cf   85  6f       sta   $6f         und merken
89d1   ad  00  20    lda   $2000       Controller-Status lesen
89d4   29  02       and   #$02        Indexloch-lED-Bit
```

```
89d6   c5  6f       cmp   $6f         auf Änderung prüfen
89d8   f0  04       beq   $89de       verzweige, wenn keine Änderung
89da   28           plp               Prozessorstatus zurückholen
89db   4c  e7  89   jmp   $89e7       Flag für ok setzen; Ende
89de   ca           dex               Zähler vermindern
89df   d0  f0       bne   $89d1       und weitersuchen
89e1   88           dey               Zähler Hi vermindern
89e2   d0  ed       bne   $89d1       und 32768 mal versuchen
89e4   28           plp               Prozessorstatus zurückholen
89e5   38           sec               Flag für Fehler setzen
89e6   24           .byte $24         nächsten Befehl überspringen
89e7   18           clc               Flag für ok setzen
89e8   68           pla               Zwischenspeicher wieder holen
89e9   85  6f       sta   $6f         und setzen
89eb   60           rts               Ende
-------------------------------------------------------------------------
89ec                                  Sprung zur RESET-Routine.
89ec   4c  a0  ea    jmp   $eaa0
-------------------------------------------------------------------------
89ef                                  Track 0 an Diskcontroller und Kopf auf Spur 0 positionieren
                                      (BUMP).
89ef   a9  b4       lda   #$b4        Zahl der Steps für weites Rückfahren
89f1   85  64       sta   $64         setzen
89f3   a9  00       lda   #$00        Track 0
89f5   8d  01  20   sta   $2001       an MFM-Controller
89f8   85  67       sta   $67         und für Kopfpositionierung abspeichern
89fa   4c  ba  87   jmp   $87ba       Kopf positionieren; Ende
-------------------------------------------------------------------------
89fd                                  Schreibschutzlichtschranke abfragen.
89fd   ad  00  1c   lda   $1c00       Steuerport des Diskcontrollers
8a00   29  10       and   #$10        Schreibschutzlichtschranke prüfen
8a02   60           rts               Ende
-------------------------------------------------------------------------
8a03                                  Parameter für Kopfpositionierung setzen; X muß die Anzahl
                                      der Steps und Y den Zieltrack ent- halten.
8a03   84  67       sty   $67         Zieltrack setzen
8a05   86  64       stx   $64         Anzahl der Steps setzen
8a07   60           rts               Ende
-------------------------------------------------------------------------
8a08                                  Programmrest.
8a08   60           rts               Ende
-------------------------------------------------------------------------
8a09                                  Header des nächsten Sektors für INQUIRE DISK lesen.
8a09   20  ef  89   jsr   $89ef       Kopf auf Track 0 positionieren
8a0c   20  b3  89   jsr   $89b3       Controller setzen; Indexloch abwarten
8a0f   b0  0f       bcs   $8a20       verzweige, wenn Fehler
8a11   20  27  8a   jsr   $8a27       Blockheader lesen
8a14   bd  7e  8a   lda   $8a7e,x     Anzahl der Sektoren pro Track holen
8a17   85  97       sta   $97         und setzen
8a19   85  61       sta   $61         und setzen als größte Sektornummer
8a1b   a9  01       lda   #$01        kleinste Sektornummer auf 1
8a1d   85  60       sta   $60         festlegen
8a1f   60           rts               Ende
8a20   a9  0d       lda   #$0d        Nummer für 'INDEX NOT FOUND'
8a22   8d  b0  01   sta   $01b0       als Rückmeldung
8a25   d0  3e       bne   $8a65       unbedingter Sprung; Ende
```

```
--------------------------------------------------------------------------
8a27                                    MFM-Sektorheader lesen.
8a27   a9  00       lda   #$00          Anzahl der Bytes pro Sektor
8a29   8d  71  02   sta   $0271         vorbesetzen
8a2c   85  44       sta   $44           und Anzahl der Blöcke pro Sektor setzen
8a2e   a9  c8       lda   #$c8          Kommando Header lesen
8a30   20  4e  88   jsr   $884e         an MFM-Controller
8a33   a2  00       ldx   #$00          Pufferindex löschen
8a35   a0  06       ldy   #$06          Zahl der Bytes im Sektorheader
8a37   ad  00  20   lda   $2000         Controller-Status lesen
8a3a   29  03       and   #$03          und Flags isolieren
8a3c   4a           lsr                 Busy-Flag ins Carry
8a3d   90  0b       bcc   $8a4a         verzweige, wenn Kommando beendet
8a3f   f0  f6       beq   $8a37         sonst verzweige, wenn Indexloch unterbrochen
8a41   ad  03  20   lda   $2003         Byte vom Controller holen
8a44   95  24       sta   $24,x         und abspeichern
8a46   e8           inx                 nächstes Byte
8a47   88           dey                 Anzahl minus 1
8a48   d0  ed       bne   $8a37         und weitermachen
8a4a   20  61  88   jsr   $8861         warten, bis Kommando beendet
8a4d   20  3c  88   jsr   $883c         Rückmeldung holen
8a50   a5  24       lda   $24           Tracknummer des Headers holen
8a52   0a           asl                 mal 2
8a53   85  64       sta   $64           als Stepperwert
8a55   a5  27       lda   $27           Kennzeichen für Sektor länge holen
8a57   29  03       and   #$03          auf 0 bis 3 begrenzen
8a59   aa           tax                 und als Index
8a5a   bd  72  8a    lda   $8a72,x      zugehörige Zahl der Bytes pro Sektorblock
8a5d   8d  71  02    sta   $0271        setzen
8a60   bd  76  8a    lda   $8a76,x      Anzahl der Blöcke pro Sektor
8a63   85  44        sta   $44          setzen
8a65   a5  5e        lda   $5e          Burst-Status
8a67   29  80        and   #$80         löschen bis auf das MFM-Flag
8a69   0d  b0  01    ora   $01b0        neuen Status bilden
8a6c   1d  7a  8a    ora   $8a7a,x      und Sektorgröße ergänzen
8a6f   85  5e        sta   $5e          Status setzen
8a71   60            rts                Ende
--------------------------------------------------------------------------
8a72                                    Anzahl der Bytes pro Sektor-Block.
8a72 7f                                  128 Bytes/Sektor
8a73 ff                                  256 Bytes/Sektor
8a74 ff                                  512 Bytes/Sektor
8a75 ff                                 1024 Bytes/Sektor
--------------------------------------------------------------------------
8a76                                    Anzahl der Blöcke pro MFM-Sektor.
8a76 01                                  128 Bytes = 1 Block  mit 128 Bytes pro Sektor
8a77 01                                  256 Bytes = 1 Block  mit 256 Bytes pro Sektor
8a78 02                                  512 Bytes = 2 Blöcke mit 256 Bytes pro Sektor
8a79 04                                 1024 Bytes = 4 Blöcke mit 256 Bytes pro Sektor
--------------------------------------------------------------------------
8a7a                                    Kennzeichen der Sektorlänge für Burst-Status.
8a7a 00                                  128 Bytes pro Sektor
8a7b 10                                  256 Bytes pro Sektor
8a7c 20                                  512 Bytes pro Sektor
8a7d 30                                 1024 Bytes pro Sektor
```

```
------------------------------------------------------------------------
8a7e                                    Anzahl der Sektoren eines Tracks.
8a7e 1a                                 26 mal  128 Byte Sektoren
8a7f 10                                 16 mal  256 Byte Sektoren
8a80 09                                  9 mal  512 Byte Sektoren
8a81 05                                  5 mal 1024 Byte Sektoren
------------------------------------------------------------------------
8a82                                    Nummern für den MFM-Fehlerstatus.
8a82 01                                 'OK'
8a83 09                                 'HEADER BLOCK CHECKSUM ERRCR'
8a84 02                                 'SECTOR NOT FOUND'
8a85 03                                 'NO ADDRESS MARK'
------------------------------------------------------------------------
8a86                                    Routine formatiert eine ganze Spur der Diskette im MFM-
                                        Format mit den gegebenen Parametern.
8a86   a9  f8       lda   #$f8         Kommando 'Track schreiben'
8a88   20  4e  88    jsr   $884e        an den MFM-Controller übergeben
8a8b   24  3b       bit   $3b          Index-Adressmarke geschrieben erden?
8a8d   50  62       bvc   $8af1        verzweige, wenn nein
8a8f   a2  50       ldx   #$50         sonst Anzahl der Bytes für Index-Start setzen
8a91   ad  00  20    lda   $2000        Controller-Status holen
8a94   29  03       and   #$03         Flags prüfen
8a96   4a           lsr                Busy-Flag testen
8a97   90  60       bcc   $8af9        verzweige, enn Kommando schon beendet
8a99   f0  f6       beq   $8a91        verzweige, enn Controller noch nicht bereit
8a9b   a9  4e       lda   #$4e         Wert für Pre-Index 1
8a9d   8d  03  20    sta   $2003        auf Diskette schreiben
8aa0   ca           dex                und zwar
8aa1   d0  ee       bne   $8a91        80 mal
8aa3   a2  0c       ldx   #$0c         Zahl der Lückenbytes setzen (12)
8aa5   ad  00  20    lda   $2000        Controller-Status holen
8aa8   29  03       and   #$03         Flags isolieren
8aaa   4a           lsr                Busy-Flag testen
8aab   90  4c       bcc   $8af9        verzweige, wenn Kommando schon beendet
8aad   f0  f6       beq   $8aa5        verzweige, wenn Controller noch nicht bereit
8aaf   a9  00       lda   #$00         Byte für Pre-Index 2
8ab1   8d  03  20    sta   $2003        auf Diskette schreiben
8ab4   ca           dex                und zwar
8ab5   d0  ee       bne   $8aa5        12 mal
8ab7   a2  03       ldx   #$03         Zähler für 3 Bytes setzen
8ab9   ad  00  20    lda   $2000        Controller-Status holen
8abc   29  03       and   #$03         Flags isolieren
8abe   4a           lsr                Busy-Flag testen
8abf   90  38       bcc   $8af9        verzweige, wenn Kommando schon beendet
8ac1   f0  f6       beq   $8ab9        verzweige, wenn Controller noch nicht bereit
8ac3   a9  f6       lda   #$f6         Taktbyte $c2
8ac5   8d  03  20    sta   $2003        auf Diskette schreiben
8ac8   ca           dex                und zwar
8ac9   d0  ee       bne   $8ab9        3 mal
8acb   ad  00  20    lda   $2000        Controller-Status holen
8ace   29  03       and   #$03         Flags i sol i eren
8ad0   4a           lsr                Busy-Flag testen
8ad1   90  26       bcc   $8af9        verzweige, wenn Kommando schon beendet
8ad3   f0  f6       beq   $8acb        verzweige, wenn Controller noch nicht bereit
8ad5   a9  fc       lda   #$fc         Byte für Index-Adressmarke
8ad7   8d  03  20    sta   $2003        auf Diskette schreiben
8ada   a2  32       ldx   #$32         und
8adc   ea           nop                mit 2 Taktzyklen Verzögerug
```

```
8add   ad  00  20    lda   $2000       Controller-Status holen
8ae0   29  03        and   #$03        Flags i sol i eren
8ae2   4a             lsr              Busy-Flag prüfen
8ae3   90  14        bcc   $8af9       verzweige, wenn Kommando schon beendet
8ae5   f0  f6        beq   $8add       verzweige, wenn Controller noch nicht bereit
8ae7   a9  4e        lda   #$4e        Byte für Post-Index
8ae9   8d  03  20    sta   $2003       auf Diskette schreiben
8aec   ca             dex              und zwar
8aed   d0  ee        bne   $8add       50 mal
8aef   f0  14        beq   $8b05       unbedingter Sprung

8af1   a2  3c        ldx   #$3c        Zähler auf 60
8af3   ad  00  20    lda   $2000       Controller-Status holen
8af6   29  03        and   #$03        Flags isolieren
8af8   4a             lsr              Busy-Flag testen
8af9   90  28        bcc   $8b23       verzweige, wenn Kommando schon beendet
8afb   f0  f6        beq   $8af3       verzweige, wenn Controller noch nicht bereit
8afd   a9  4e        lda   #$4e        Füllbyte für Trackvorspann
8aff   8d  03  20    sta   $2003       auf Diskette schreiben
8b02   ca             dex              und zwar
8b03   d0  ee        bne   $8af3       60 mal
8b05   a0  01        ldy   #$01        Zähler für Sektoren setzen

8b07   a2  0c        ldx   #$0c        Zähler auf 12
8b09   ad  00  20    lda   $2000       Controller-Status holen
8b0c   29  03        and   #$03        Flags isolieren
8b0e   4a             lsr              Busy-Flag isolieren
8b0f   90  12        bcc   $8b23       verzweige, wenn Kommando schon beendet
8b11   f0  f6        beq   $8b09       verzweige, wenn Controller noch nicht bereit
8b13   a9  00        lda   #$00        Byte für Sektorvorspann
8b15   8d  03  20    sta   $2003       auf Diskette schreiben
8b18   ca             dex              und zwar
8b19   d0  ee        bne   $8b09       12 mal
8b1b   a2  03        ldx   #$03        Zähler auf 3
8b1d   ad  00  20    lda   $2000       Controller-Status
8b20   29  03        and   #$03        Flags isolieren
8b22   4a             lsr              Busy-Flag testen
8b23   90  57        bcc   $8b7c       verzweige, wenn Kommando schon beendet
8b25   f0  f6        beq   $8b1d       verzweige, wenn Controller noch nicht bereit
8b27   a9  f5        lda   #$f5        Taktbyte $a1
8b29   8d  03  20    sta   $2003       auf Diskette schreiben
8b2c   ca             dex              und zwar
8b2d   d0  ee        bne   $8b1d       3 mal
8b2f   ad  00  20    lda   $2000       Controller-Status holen
8b32   29  03        and   #$03        Flags isolieren
8b34   4a             lsr              Busy-Flag testen
8b35   90  45        bcc   $8b7c       verzweige, wenn Kommando schon beendet
8b37   f0  f6        beq   $8b2f       verzweige, wenn Controller noch nicht bereit
8b39   a9  fe        lda   #$fe        ID Adressmarke
8b3b   8d  03  20    sta   $2003       auf Diskette schreiben
8b3e   ad  00  20    lda   $2000       Controller-Status holen
8b41   29  03        and   #$03        FlagS isolieren
8b43   4a             lsr              Busy-Flag testen
8b44   90  36        bcc   $8b7c       verzweige, wenn Kommando schon beendet
8b46   f0  f6        beq   $8b3e       verzweige, wenn Controller noch nicht bereit
8b48   ad  b0  01    lda   $01b0       Tracknummer bei Formatierung
8b4b   8d  03  20    sta   $2003       auf Diskette schreiben
8b4e   ad  00  20    lda   $2000       Controller-Status holen
```

```
8b51   29  03       and   #$03        Flags isolieren
8b53   4a           lsr               Busy-Flag testen
8b54   90  26       bcc   $8b7c       verzweige, wenn KOITITndo schon beendet
8b56   f0  f6       beq   $8b4e       verzweige, wenn Controller noch nicht bereit
8b58   a5  3b       lda   $3b         Befehlsbyte
8b5a   29  10       and   #$10        Seitenkennzeichen isolieren
8b5c   d0  03       bne   $8b61       verzweige, wenn Seite 1
8b5e   a9  00       lda   #$00        Kennzeichen für Diskettensei:e 0
8b60   2c           .byte $2c         nächsten Befehl überspringen
8b61   a9  01       lda   #$01        Kennzeichen für Diskettenseite i
8b63   8d  03  20    sta   $2003       auf Diskette schreiben
8b66   ad  00  20    lda   $2000       Controller-Status holen
8b69   29  03       and   #$03        Flags isolieren
8b6b   4a           lsr               Busy-Flag testen
8b6c   90  0e       bcc   $8b7c       verzweige, wenn Kommando schon beendet
8b6e   f0  f6       beq   $8b66       verzweige, wenn Contrller noch nicht bereit
8b70   b9  0a  02    lda   $020a,y     Sektornummer aus Tabelle holen
8b73   8d  03  20    sta   $2003       und auf Diskette schreiben
8b76   ad  00  20    lda   $2000       Controller-Status holen
8b79   29  03       and   #$03        Flags isolieren
8b7b   4a           lsr               Busy-Flag testen
8b7c   90  33       bcc   $8bb1       verzweige, wenn Kommando schon beendet
8b7e   f0  f6       beq   $8b76       verzweige, wenn Controller noch nicht bereit
8b80   ad  05  02    lda   $0205       Kennzeichen für Länge des Sektors
8b83   8d  03  20    sta   $2003       auf Diskette schreiben
8b86   ad  00  20    lda   $2000       Controller-Status holen
8b89   29  03       and   #$03        Flags isolieren
8b8b   4a           lsr               Busy-Flag testen
8b8c   90  23       bcc   $8bb1       verzweige, wenn Kommando schon beendet
8b8e   f0  f6       beq   $8b86       verzweige, wenn Controller noch nicht bereit
8b90   a9  f7       lda   #$f7        2 CRC-Bytes
8b92   8d  03  20    sta   $2003       auf Diskette schreiben (simultan)
8b95   a2  16       ldx   #$16        Zähler auf 22
8b97   ad  00  20    lda   $2000       Controller-Status holen
8b9a   29  03       and   #$03        Flags isolieren
8b9c   4a           lsr               Busy-Flag testen
8b9d   90  12       bcc   $8bb1       verzweige, wenn Kommando schon beendet
8b9f   f0  f6       beq   $8b97       verzweige, wenn Controller noch nicht bereit
8ba1   a9  4e       lda   #$4e        Lückenbyte
8ba3   8d  03  20    sta   $2003       auf Diskette schreiben
8ba6   ca           dex               und zwar
8ba7   d0  ee       bne   $8b97       22 mal
8ba9   a2  0c       ldx   #$0c        Zähler auf 12
8bab   ad  00  20    lda   $2000       Controller-Status holen
8bae   29  03       and   #$03        Flags isolieren
8bb0   4a           lsr               Busy-Flag testen
8bb1   90  38       bcc   $8beb       verzweige, wenn Kommando schon beendet
8bb3   f0  f6       beq   $8bab       verzweige, wenn Controller noch nicht bereit
8bb5   a9  00       lda   #$00        Lückenbyte
8bb7   8d  03  20    sta   $2003       auf Diskette schreiben
8bba   ca           dex               und zwar
8bbb   d0  ee       bne   $8bab       12 mal
8bbd   a2  03       ldx   #$03        Zähler auf 3
8bbf   ad  00  20    lda   $2000       Controller-Status holen
8bc2   29  03       and   #$03        Flags isolieren
8bc4   4a           lsr               Busy-Flag testen
8bc5   90  24       bcc   $8beb       verzweige, wenn Kommando schon beendet
8bc7   f0  f6       beq   $8bbf       verzweige, wenn Controller noch nicht bereit
```

```
8bc9   a9  f5       lda   #$f5        Taktbyte $a1
8bcb   8d  03  20   sta   $2003       auf Diskette schreiben
8bce   ca           dex               und zwar
8bcf   d0  ee       bne   $8bbf       3 mal
8bd1   ad  00  20   lda   $2000       Controller-Status holen
8bd4   29  03       and   #$03        Flags isolieren
8bd6   4a           lsr               Busy-Flag testen
8bd7   90  12       bcc   $8beb       verzweige, wenn Kommando schon beendet
8bd9   f0  f6       beq   $8bd1       verzweige, wenn Controller noch nicht bereit
8bdb   a9  fb       lda   #$fb        Data Address Mark
8bdd   8d  03  20   sta   $2003       auf Diskette schreiben
8be0   84  6f       sty   $6f         Zeiger in Sektortabelle merken
8be2   a4  44       ldy   $44         Anzahl der 256-Byte-Blöcke pro Sektor
8be4   ea           nop               Pause
8be5   ad  00  20   lda   $2000       Controller-Status holen
8be8   29  03       and   #$03        Flags isolieren
8bea   4a           lsr               Busy-Flag testen
8beb   90  60       bcc   $8c4d       verzweige, wenn Kommando schon beendet
8bed   f0  f6       beq   $8be5       verzweige, wenn Controller noch nicht bereit
8bef   ad  0a  02   lda   $020a       Füllbyte für Sektor
8bf2   8d  03  20   sta   $2003       auf Diskette schreiben
8bf5   ec  71  02   cpx   $0271       und zwar 128, 256, 512 oder 1024 mal
8bf8   f0  04       beq   $8bfe       je nachdem, was als
8bfa   e8           inx               Sektor länge
8bfb   4c  e5  8b   jmp   $8be5       angegeben wurde
8bfe   e8           inx               nächster 256-Byte-Block?
8bff   88           dey               weitermachen,
8c00   d0  e3       bne   $8be5       wenn ja
8c02   ad  00  20   lda   $2000       Controller-Status holen
8c05   29  03       and   #$03        Flags isolieren
8c07   4a           lsr               Busy-Flag testen
8c08   90  43       bcc   $8c4d       verzweige, wenn Kommando schon beendet
8c0a   f0  f6       beq   $8c02       verzweige, wenn Controller noch nicht bereit
8c0c   a9  f7       lda   #$f7        2 CRC-Bytes
8c0e   8d  03  20   sta   $2003       auf Diskette schreiben (simultan)
8c11   ac  05  02   ldy   $0205       Kennzeichen für Größe der Sektoren
8c14   b9  4f  8c   lda   $8c4f,y     entsprechende Anzahl von Lückenbytes holen
8c17   a4  6f       ldy   $6f         Zeiger in Sektortabelle holen
8c19   aa           tax               Anzahl der Lückenbytes
8c1a   ad  00  20   lda   $2000       Controller-Status holen
8c1d   29  03       and   #$03        Flags isolieren
8c1f   4a           lsr               Busy-Flag testen
8c20   90  2b       bcc   $8c4d       verzweige, wenn Kommando schon bbendet
8c22   f0  f6       beq   $8c1a       verzweige, wenn Controller noch nicht bereit
8c24   a9  4e       lda   #$4e        Lückenbyte
8c26   8d  03  20   sta   $2003       auf Diskette schreiben
8c29   ca           dex               und zwar
8c2a   d0  ee       bne   $8c1a       der Größe der Lücke entsprechend oft
8c2c   cc  07  02   cpy   $0207       Anzahl der Sektoren des Tracks erreicht?
8c2f   f0  04       beq   $8c35       verzweige, wenn ja
8c31   c8           iny               sonst Sektornummer erhöhen
8c32   4c  07  8b   jmp   $8b07       und nächsten Sektor schreiben
8c35   ad  00  20   lda   $2000       Controller-Status holen
8c38   29  03       and   #$03        Flags isolieren
8c3a   4a           lsr               Busy-Flag testen
8c3b   90  0b       bcc   $8c48       verzweige, wenn Kommando schon beendet
8c3d   f0  f6       beq   $8c35       verzweige, wenn Controller noch nicht bereit
8c3f   18           clc               Füllbyte für
```

```
8c40   a9  4e       lda   #$4e       die Lücke
8c42   8d  03  20    sta   $2003      auf Diskette schreiben,
8c45   4c  35  8c    jmp   $8c35      bis Spur vollgeschrieben ist
8c48   20  61  88    jsr   $8861      Ausführung des K0m7ndos aarten
8c4b   18            clc              Flag für ok setzen
8c4c   24            .byte $24        nächsten Befehl überspringer
8c4d   38            sec              Flag für Fehler setzen
8c4e   60            rts              Ende
-------------------------------------------------------------------------
8c4f                                  Anzahl der Lückenbytes zwischen MFM-Sektoren.
8c4f 07                               bei  128 Byte Sektoren
8c50 0c                               bei  256 Byte Sektoren
8c51 17                               bei  512 Byte Sektoren
8c52 2c                               bei 1024 Byte Sektoren
-------------------------------------------------------------------------
8c53                                  Anzahl der MFM-Sektoren pro Trac(.
8c53 1°                               bei  128 Byte Sektoren
8c54 10                               bei  256 Byte Sektoren
8c55 09                               bei  512 Byte Sektoren
8c56 05                               bei 1024 Byte sektoren
-------------------------------------------------------------------------
8c57                                  Gesamte Diskette im MFM-Format formatieren.
8c57   a5  3b       lda   $3b        Befehlsbyte
8c59   29  08       and   #$08       ist die Diskette schreibgeschützt?
8c5b   f0  07       beq   $8c64      verzeige, enn nein
8c5d   a6  46       ldx   $46        sonst Fehlernummer holen
8c5f   8e  b0  01   stx   $01b0      und als Rückmeldung setzen
8c62   38           sec              Fehlerflag setzen
8c63   60           rts              Ende
8c64   20  07  d3   jsr   $d307      alLe KanäLe Löschen
8c67   ad  74  02   lda   $0274      Länge des BefehLsstrings
8c6a   38           sec              minus
8c6b   e9  04       sbc   #$04       4 Bytes für Standardbefehl
8c6d   a8           tay              als Index
8c6e   f0  20       beq   $8c90      verzweige, wenn gleich Null
8c70   88           dey              minus 1
8c71   f0  22       beq   $8c95      verzweige, wenn gleich Null
8c73   a9  00       lda   #$00       Track 0
8c75   8d  b0  01   sta   $01b0      als erste Spur für Formatierung setzen
8c78   ad  05  02   lda   $0205      Kennzeichen für Sektor länge holen
8c7b   20  57  8a   jsr   $8a57      Parameter für Sektorgröße setzen
8c7e   88           dey              je nach
8c7f   f0  21       beq   $8ca2      Anzahl der
8c81   88           dey              angegebenen Parameter
8c82   f0  23       beq   $8ca7      erfolgt der
8c84   88           dey              Einsprung zum Setzen
8c85   f0  26       beq   $8cad      der Defaultwerte für
8c87   88           dey              die Formatierung
8c88   f0  2b       beq   $8cb5      früher oder später, da
8c8a   88           dey              die Angabe der Parameter
8c8b   f0  2d       beq   $8cba      optional ist!
8c8d   4c  bf  8c   jmp   $8cbf      alle Parameter angegeben; keine Defaults setzen
-------------------------------------------------------------------------
8c90                                  Parameter setzen und Diskette formatieren.
8c90   a9  00       lda   #$00       0 als
8c92   8d  04  02   sta   $0204      Sektor-Interleave (Sektorabstand)
8c95   a9  00       lda   #$00       0 als
8c97   8d  b0  01   sta   $01b0      erste Tracknummer für Formatierung
```

```
8c9a    a9  01       lda   #$01        1 als
8c9c    8d  05  02   sta   $0205       Kennzeichen für 256-Byte-Sektoren
8c9f    20  57  8a   jsr   $8a57       Parameter entsprechend dieser Länge setzen
8ca2    a9  27       lda   #$27        39 als
8ca4    8d  06  02   sta   $0206       höchste Tracknummer
8ca7    bd  53  8c   lda   $8c53,x     Anzahl der Sektoren pro Track holen
8caa    8d  07  02   sta   $0207       und setzen
8cad    a9  00       lda   #$00        0 als
8caf    8d  08  02   sta   $0208       logischer Anfangstrack
8cb2    8d  01  20   sta   $2001       Tracknummer für MFM-Controller setzen
8cb5    a9  00       lda   #$00        0 als
8cb7    8d  09  02   sta   $0209       Offset für Track 0 setzen
8cba    a9  e5       lda   #$e5        $e5 als
8cbc    8d  0a  02   sta   $020a       Füllbyte der Sektoren setzen
8cbf    20  de  8c   jsr   $8cde       Diskettenseite formatieren
8cc2    ad  b0  01   lda   $01b0       Rückmeldung holen
8cc5    e0  02       cpx   #$02        Fehler?
8cc7    b0  12       bcs   $8cdb       verzweige, wenn ja
8cc9    a5  3b       lda   $3b         Befehlsbyte
8ccb    29  20       and   #$20        soll zweiseitig formatiert werden?
8ccd    f0  0c       beq   $8cdb       verzweige, wenn nein; Ende
8ccf    a5  3b       lda   $3b         Nummer für Diskettenseite
8cd1    09  10       ora   #$10        auf 1
8cd3    85  3b       sta   $3b         setzen
8cd5    20  54  89   jsr   $8954       Hardware für entsprechende Seite setzen
8cd8    20  de  8c   jsr   $8cde       Diskettenseite entsprechend formatieren
8cdb    4c  ef  89   jmp   $89ef       Kopf auf Track 0 positionieren; Ende
--------------------------------------------------------------------------------
8cde                                   Eine Diskettenseite im MFM-Format formatieren.
8cde    20  b3  89   jsr   $89b3       Controller initialisieren; Indexloch abwarten
8ce1    b0  7c       bcs   $8d5f       verzweige, wenn Fehler
8ce3    a9  01       lda   #$01        Flag für Diskettenwechsel
8ce5    8d  0d  18   sta   $180d       löschen
8ce8    20  ef  89   jsr   $89ef       Kopf auf Track 0 positionieren
8ceb    ad  08  02   lda   $0208       neuntes Zeichen aus Befehsstring
8cee    8d  b0  01   sta   $01b0       als Starttrack merken
8cf1    8d  01  20   sta   $2001       und an MFM-Controller
8cf4    2c  03  02   bit   $0203       Flag für Sektortabelle gesetzt?
8cf7    70  05       bvs   $8cfe       verzweige, wenn ja
8cf9    20  88  88   jsr   $8888       sonst Sektortabelle herstellen
8cfc    b0  61       bcs   $8d5f       verzweige, wenn Fehler
8cfe    ad  09  02   lda   $0209       zehntes Zeichen aus Befehlsstring
8d01    29  7f       and   #$7f        Offset für Track 0
8d03    f0  08       beq   $8d0d       verzweige, wenn Null
8d05    18           clc               sonst
8d06    65  67       adc   $67         zum Stepperwert addieren und
8d08    85  67       sta   $67         Trackverschiebung neu setzen
8d0a    20  ba  87   jsr   $87ba       Kopf positionieren
8d0d    78           sei               Diskcontroller inaktivieren
8d0e    ad  0d  18   lda   $180d       auf Wechseln der Diskette
8d11    4a           lsr               prüfen
8d12    b0  4b       bcs   $8d5f       verzweige, wenn Diskette gewechselt wurde
8d14    20  86  8a   jsr   $8a86       Track formatieren
8d17    b0  46       bcs   $8d5f       verzweige, wenn Fehler
8d19    ad  0d  18   lda   $180d       auf Wechsel der Diskette
8d1c    4a           lsr               prüfen
8d1d    b0  40       bcs   $8d5f       verzweige, wenn Diskette gewechselt wurde
8d1f    20  f0  88   jsr   $88f0       sonst Verify durchführen
```

```
8d22   b0  3b       bcs   $8d5f       verzweige, wenn Fehler
8d24   ad  0d  18    lda   $180d       auf Yechsel der Dis<ette
8d27   4a            lsr               prüfen
8d28   b0  35       bcs   $8d5f       verzweige, wenn Dis<ette se_echselt wurde
8d2a   ad  b0  01    lda   $01b0       Tracknummer für Formotier_r9
8d2d   cd  06  02    cmp   $0206       mit maximaler Trackn_r ver;leichen
8d30   f0  0e       beq   $8d40       verzweige, wenn erreicht
8d32   e6  67       inc   $67         sonst Stepperzeiger auf nac_sten Track
8d34   ee  01  20    inc   $2001       und MFM-Controller entspreche')<::: setzen
8d37   ee  b0  01    inc   $01b0       Track für Formatierung setze')
8d3a   20  ba  87    jsr   $87ba       Kopf positionieren
8d3d   4c  0d  8d    jmp   $8d0d       nächsten Track formatieren
8d40   24  3b       bit   $3b         Befehlsbyte; Diskette nur teilNeise formatiert?
8d42   10  18       bpl   $8d5c       verzweige, wenn nein
8d44   38            sec               sonst
8d45   ad  06  02    lda   $0206       letzten Track
8d48   ed  08  02    sbc   $0208       minus ersten Track rechnen
8d4b   c9  27       cmp   #$27        Ergebnis größer gleich 39?
8d4d   b0  0d       bcs   $8d5c       verzweige, wenn ja; Diskette vollständig
8d4f   e6  67       inc   $67         sonst Trackzähler erhöhen
8d51   20  ba  87    jsr   $87ba       Kopf positionieren
8d54   a2  1c       ldx   #$1c        7168 mal
8d56   20  63  9d    jsr   $9d63       $55 auf Diskette schreiben
8d59   20  00  fe    jsr   $fe00       auf Lesen umschalten
8d5c   a2  00       ldx   #$00        Nummer für alles ok
8d5e   2c           .byte $2c         nächsten Befehl überspringen
8d5f   a2  06       ldx   #$06        Nummer für 'FORMAT ERROR'
8d61   8e  b0  01    stx   $01b0       setzen
8d64   4c  e9  85    jmp   $85e9       und zur Ausgabe; Ende
-------------------------------------------------------------------------
8d67                                   MFM-Sektor von der Diskette lesen und ggf. an den Computer
                                       senden.
8d67   a5  3b       lda   $3b         Befehlsbyte
8d69   29  20       and   #$20        nur Pufferinhalt übertragen?
8d6b   d0  59       bne   $8dc6       verzweige, wenn ja
8d6d   a9  03       lda   #$03        Pufferadresse Hi
8d6f   85  31       sta   $31         setzen
8d71   a0  00       ldy   #$00        Pufferadresse Lo
8d73   84  30       sty   $30         setzen; ergibt $0300
8d75   a6  44       ldx   $44         Anzahl der 256-Byte-Blöcke pro Sektor
8d77   ad  03  02    lda   $0203       Tracknummer holen
8d7a   8d  01  20    sta   $2001       und an MFM-Controller übergeben
8d7d   ad  04  02    lda   $0204       Sektornummer holen
8d80   8d  02  20    sta   $2002       und an MFM-Controller übergeben
8d83   a9  88       lda   #$88        Befehl für Sektor lesen
8d85   20  4e  88    jsr   $884e       an MFM-Controller übergeben
8d88   ea            nop               Pause (2 Taktzyklen)
8d89   ad  00  20    lda   $2000       Controller-Status holen
8d8c   29  03       and   #$03        Flags isolieren
8d8e   4a            lsr               Busy-Flag testen
8d8f   90  1a       bcc   $8dab       verzweige, wenn Kommando schon beendet
8d91   29  01       and   #$01        sonst: Datenbyte schon eingelesen?
8d93   f0  f4       beq   $8d89       verzweige, wenn nein
8d95   ad  03  20    lda   $2003       Datenbyte holen
8d98   91  30       sta   ($30),y     und in Puffer schreiben
8d9a   cc  71  02    cpy   $0271       schon alle Bytes eines Blocks geholt?
8d9d   f0  03       beq   $8da2       verzweige, wenn ja
8d9f   c8            iny               sonst Zeiger auf nächstes Byte
```

```
8da0   d0  e7       bne   $8d89       weitermachen
8da2   c8            iny              Zeiger auf nächstes Byte
8da3   ca            dex              Anzahl der Blöcke vermindern
8da4   f0  05       beq   $8dab       verzweige, wenn schon alle Blöcke gelesen
8da6   e6  31       inc   $31         sonst Pufferadresse Hi erhöhen
8da8   4c  89  8d    jmp   $8d89       und weitermachen, bis ganzer Sektor eingelesen
8dab   20  61  88    jsr   $8861       Kommando beenden; LED am Laufwerk ausschalten
8dae   20  3c  88    jsr   $883c       Controller-Status holen; Fehlerflags isolieren
8db1   20  e9  85    jsr   $85e9       Burst-Status für Ausgabe vorbereiten
8db4   24  3b       bit   $3b         Befehlsbyte
8db6   70  07       bvs   $8dbf       verzweige, wenn keine Fehlerprüfung
8db8   e0  02       cpx   #$02        ist Fehler aufgetreten?
8dba   90  03       bcc   $8dbf       verzweige, wenn nein
8dbc   4c  84  83    jmp   $8384       Fehlermeldung an Computer; Ende
8dbf   20  f9  85    jsr   $85f9       Burst-Status ausgeben
8dc2   a5  3b       lda   $3b         Befehlsbyte
8dc4   30  22       bmi   $8de8       verzweige, wenn keine Datenübertragung
8dc6   a9  03       lda   #$03        sonst Pufferadresse Hi
8dc8   85  31       sta   $31         setzen
8dca   a0  00       ldy   #$00        Pufferadresse Lo und Index
8dcc   84  30       sty   $30         setzen
8dce   a6  44       ldx   $44         Anzahl der 256-Byte-Blöcke pro Sektor
8dd0   b1  30       lda   ($30),y     Byte aus Puffer holen
8dd2   85  46       sta   $46         und für Ausgabe setzen
8dd4   20  f9  85    jsr   $85f9       Byte zum Computer
8dd7   cc  71  02    cpy   $0271       schon ein Block gesendet?
8dda   f0  03       beq   $8ddf       verzweige, wenn ja
8ddc   c8            iny              sonst Pufferzeiger erhöhen
8ddd   d0  f1       bne   $8dd0       und weitermachen
8ddf   c8            iny              Pufferindex auf nächstes Byte
8de0   ca            dex              Anzahl der Blöcke des Sektors vermindern
8de1   f0  05       beq   $8de8       und Ende, wenn alle gesendet
8de3   e6  31       inc   $31         sonst Pufferadresse Hi erhöhen
8de5   4c  d0  8d    jmp   $8dd0       und die nächsten Bytes senden
8de8   ce  05  02    dec   $0205       Anzahl der Sektoren vermindern
8deb   f0  06       beq   $8df3       Ende, wenn alle übertragen
8ded   20  6c  88    jsr   $886c       sonst nächste Sektornummer holen
8df0   4c  67  8d    jmp   $8d67       und Sektor lesen
8df3   4c  1b  89    jmp   $891b       Kopf positionieren; Ende
--------------------------------------------------------------------------
8df6                                   Routine für BURST-WRITE. MFM-Sektor auf Diskette schreiben,
                                       der entweder im Puffer steht, oder erst vom Computer geholt
                                       werden muß. Es kann auch nur der Puffer vollgeschrieben
                                       werden, ohne den Sektor auf Diskette zu schreiben.
8df6   a9  03       lda   #$03        Pufferadresse Hi
8df8   85  31       sta   $31         setzen
8dfa   a0  00       ldy   #$00        Pufferadresse Lo und Index
8dfc   84  30       sty   $30         setzen
8dfe   a6  44       ldx   $44         Anzahl der 256-Byte-Blöcke pro Sektor
8e00   a5  3b       lda   $3b         Befehlsbyte
8e02   30  30       bmi   $8e34       verzweige, wenn Sektor schon im Puffer
8e04   ad  00  18    lda   $1800       sonst Daten vom Rechner holen
8e07   49  08       eor   #$08        CLOCK OUT-Leitung invertieren
8e09   2c  0d  40    bit   $400d       IRQ-Status löschen
8e0c   8d  00  18    sta   $1800       Byte auf Bus
8e0f   ad  00  18    lda   $1800       Bus lesen
8e12   10  03       bpl   $8e17       verzweige, wenn kein ATN
```

```
8e14   20  59  ea    jsr   $ea59       sonst auf ATN-Signal testen
8e17   ad  0d  40    lda   $400d       Status für Schiebereg;ster
8e1a   29  08        and   #$08        testen
8e1c   f0  f1        beq   $8e0f       warten, wenn noch kein Byte _=e"tragen
8e1e   ad  0c  40    lda   $400c       Byte holen
8e21   91  30        sta   ($30),y     und in Puffer schreiben
8e23   cc  71  02    cpy   $0271       schon alle Bytes des Blcc(s ge.esen?
8e26   f0  03        beq   $8e2b       verzweige, wenn ja
8e28   c8            iny               Zeiger auf nächstes Byte
8e29   d0  d9        bne   $8e04       und weitermachen
8e2b   c8            iny               Zeiger auf nächstes Byte
8e2c   ca            dex               und Anzahl der Blöcke vermindern
8e2d   f0  05        beq   $8e34       verzweige, wenn kein Block mehr
8e2f   e6  31        inc   $31         sonst Pufferadresse Hi erhöhen
8e31   4c  04  8e    jmp   $8e04       und weitermachen
8e34   a5  3b        lda   $3b         Befehlsbyte
8e36   29  20        and   #$20        soll Pufferinhalt auf Diskette?
8e38   d0  7d        bne   $8eb7       verzweige, wenn nein
8e3a   a5  3b        lda   $3b         Befehlsbyte
8e3c   29  08        and   #$08        Schreibschutz auf der Diskette?
8e3e   f0  05        beq   $8e45       verzweige, wenn nein
8e40   a6  46        ldx   $46         sonst Fehlerstatus holen
8e42   4c  81  83    jmp   $8381       $8381 und an Computer ausgeben
8e45   a9  03        lda   #$03        Pufferadresse Hi
8e47   85  31        sta   $31         setzen
8e49   a0  00        ldy   #$00        Pufferadresse Lo und Index
8e4b   84  30        sty   $30         setzen
8e4d   a6  44        ldx   $44         Anzahl der 256-Byte-Blöcke des Sektors
8e4f   ad  03  02    lda   $0203       Tracknummer holen
8e52   8d  01  20    sta   $2001       und an MFM-Controller übergeben
8e55   ad  04  02    lda   $0204       Sektornummer aus Befehlsstring holen
8e58   8d  02  20    sta   $2002       und an MFM-Controller übergeben
8e5b   ad  0d  18    lda   $180d       Steuerport lesen
8e5e   4a            lsr               wurde die Diskette gewechselt?
8e5f   b0  32        bcs   $8e93       verzweige, wenn ja
8e61   a9  a8        lda   #$a8        Befehl Sektor schreiben
8e63   20  4e  88    jsr   $884e       an MFM-Controller übergeben
8e66   ad  00  20    lda   $2000       Controller-Status holen
8e69   29  03        and   #$03        Flags isolieren
8e6b   4a            lsr               Busy-Flag testen
8e6c   90  25        bcc   $8e93       verzweige, wenn Kommando schon beendet
8e6e   29  01        and   #$01        Controller bereit?
8e70   f0  f4        beq   $8e66       verzweige, wenn nein
8e72   b1  30        lda   ($30),y     sonst Byte aus Puffer holen
8e74   8d  03  20    sta   $2003       und auf Diskette schreiben
8e77   cc  71  02    cpy   $0271       schon alle Bytes des Blocks?
8e7a   f0  03        beq   $8e7f       verzweige, wenn ja
8e7c   c8            iny               sonst Zeiger auf nächstes Byte
8e7d   d0  e7        bne   $8e66       und weitermachen
8e7f   c8            iny               Zeiger auf nächstes Byte
8e80   ca            dex               und Anzahl der Blöcke vermindern
8e81   f0  05        beq   $8e88       verzweige, wenn alle Blöcke geschrieben
8e83   e6  31        inc   $31         Pufferadresse Hi erhöhen
8e85   4c  66  8e    jmp   $8e66       und weitermachen
8e88   ad  0d  18    lda   $180d       Steuerport lesen
8e8b   4a            lsr               wurde die Diskette gewechselt?
8e8c   b0  05        bcs   $8e93       verzweige, wenn ja
8e8e   20  c6  8e    jsr   $8ec6       Verify für Sektor durchführen
```

```
8e91   90 07       bcc   $8e9a       verzweige, wenn Sektor in Ordnung
8e93   20 ce 81    jsr   $81ce       seriellen Bus auf Ausgang
8e96   a2 07       ldx   #$07        Nummer für 'VERIFY ERROR'
8e98   d0 06       bne   $8ea0       unbedingter Sprung zur Fehlerausgabe
8e9a   20 ce 81    jsr   $81ce       seriellen Bus auf Ausgang
8e9d   20 3c 88    jsr   $883c       Controller-Status holen
8ea0   8e b0 01    stx   $01b0       und merken
8ea3   20 e9 85    jsr   $85e9       Burst-Status setzen
8ea6   20 f9 85    jsr   $85f9       und zum Computer senden
8ea9   20 a0 86    jsr   $86a0       auf Reaktion des Computers warten
8eac   20 b2 81    jsr   $81b2       und seriellen Bus auf Eingang
8eaf   24 3b       bit   $3b         Befehlsbyte
8eb1   70 04       bvs   $8eb7       verzweige, wenn keine Fehlerprüfung gewünscht
8eb3   e0 02       cpx   #$02        Fehler?
8eb5   b0 0e       bcs   $8ec5       verzweige, wenn ja
8eb7   ce 05 02    dec   $0205       Anzahl der Sektoren vermindern
8eba   f0 06       beq   $8ec2       und Ende, wenn alle geschrieben
8ebc   20 6c 88    jsr   $886c       Nummer des nächsten Sektors holen
8ebf   4c f6 8d    jmp   $8df6       und weitermachen
8ec2   4c 1b 89    jmp   $891b       Kopf auf nächsten Track; Ende
8ec5   60          rts               Ende
--------------------------------------------------------------------------
8ec6                                 Verify eines MFM-Sektors mit dem Inhalt des Pufferspeichers
                                     ab Puffer 0 bis maximal Puffer 3. Tritt ein Fehler auf, so
                                     wird vor der Rückkehr das Carry-Flag gesetzt.
8ec6   a9 03       lda   #$03        Pufferadresse Hi
8ec8   85 31       sta   $31         setzen
8eca   a0 00       ldy   #$00        Pufferadresse Lo und Index
8ecc   84 30       sty   $30         setzen
8ece   a6 44       ldx   $44         Anzahl der 256-Byte-Blöcke des Sektors
8ed0   ad 03 02    lda   $0203       Tracknummer aus Befehlsstring
8ed3   8d 01 20    sta   $2001       an MFM-Controller übergeben
8ed6   ad 04 02    lda   $0204       Sektornummer aus Befehlsstring
8ed9   8d 02 20    sta   $2002       an MFM-Controller übergeben
8edc   a9 88       lda   #$88        Kommando für Sektor lesen
8ede   20 4e 88    jsr   $884e       an Controller übergeben
8ee1   ad 00 20    lda   $2000       Controller.Status holen
8ee4   29 03       and   #$03        Flags isolieren
8ee6   4a          lsr               Busy-Flag testen
8ee7   90 1c       bcc   $8f05       verzweige, wenn Kommando schon ausgeführt
8ee9   29 01       and   #$01        sonst testen, ob Byte schon eingelesen
8eeb   f0 f4       beq   $8ee1       warten, wenn nein
8eed   ad 03 20    lda   $2003       Byte holen und
8ef0   d1 30       cmp   ($30),y     mit Pufferinhalt vergleichen
8ef2   d0 11       bne   $8f05       verzweige, wenn ungleich
8ef4   cc 71 02    cpy   $0271       schon ein Block verglichen?
8ef7   f0 03       beq   $8efc       verzweige, wenn ja
8ef9   c8          iny               Zeiger auf nächstes Byte
8efa   d0 e5       bne   $8ee1       und weiterprüfen
8efc   c8          iny               Zeiger auf nächstes Byte
8efd   ca          dex               Anzahl der Blöcke des Sektors vermindern
8efe   f0 10       beq   $8f10       verzweige, wenn fertig
8f00   e6 31       inc   $31         sonst Pufferadresse Hi erhöhen
8f02   4c e1 8e    jmp   $8ee1       und nächsten Block bearbeiten
8f05   a9 d0       lda   #$d0        Kommando für Abbruch der Arbeit
8f07   8d 00 20    sta   $2000       an den MFM-Controller übergeben, da Fehler
8f0a   20 83 a4    jsr   $a483       ca. 70 Taktzyklen warten
```

```
8f0d   a2  07       ldx   #$07       Nummer für 'VERIFY ERROR'
8f0f   2c           .byte $2c        nächsten Befehl übersprirgen
8f10   a2  00       ldx   #$00       Nummer für 'OK'
8f12   8e  b0  01   stx   $01b0      Status merken
8f15   4c  61  88   jmp   $8861      Kommando Ende abwarten; Ende
-------------------------------------------------------------------------
8f18                                 Neuformatierten MFM-Sektor mit Verify auf korrekten
                                     Leerinhalt überprüfen
8f18   a9  03       lda   #$03       Pufferadresse Hi
8f1a   85  31       sta   $31        setzen
8f1c   a0  00       ldy   #$00       Pufferadresse Lo und Index
8f1e   84  30       sty   $30        setzen
8f20   a6  44       ldx   $44        Anzahl der 256-Byte-Blöcke des Sektors holen
8f22   ac  71  02   ldy   $0271      Anzahl der Bytes eines Blocks (128/256)
8f25   a9  88       lda   #$88       Kommando für Sektor lesen
8f27   20  4e  88   jsr   $884e      an den MFM-Controller übergeben
8f2a   ad  00  20   lda   $2000      Controller-Status holen
8f2d   29  03       and   #$03       Flags isolieren
8f2f   4a           lsr              auf Busy-Flag testen
8f30   90  1a       bcc   $8f4c      verzweige, wenn Kommando schon beendet
8f32   29  01       and   #$01       sonst testen, ob Controller bereit
8f34   f0  f4       beq   $8f2a      verzweige, wenn nein
8f36   ad  03  20   lda   $2003      Datenbyte von Diskette holen
8f39   cd  0a  02   cmp   $020a      und mit aktuellem Füllbyte vergleichen
8f3c   d0  0e       bne   $8f4c      verzweige, wenn ungleich
8f3e   88           dey              Zähler auf nächstes Byte
8f3f   10  e9       bpl   $8f2a      Fehler! (muß bne $8f2a heißen); weitermachen
8f41   ca           dex              nächster Block des Sektors
8f42   f0  13       beq   $8f57      verzweige, wenn schon ganzer Sektor verglichen
8f44   ac  71  02   ldy   $0271      Anzahl der Bytes pro Block (128/256)
8f47   e6  31       inc   $31        Pufferadresse Hi erhöhen
8f49   4c  2a  8f   jmp   $8f2a      und weitermachen
8f4c   a9  d0       lda   #$d0       Kommando für Befehl abbrechen
8f4e   8d  00  20   sta   $2000      an MFM-Controller übergeben
8f51   20  83  a4   jsr   $a483      ca. 70 Taktzyklen warten
8f54   a2  07       ldx   #$07       Nummer für 'VERIFY ERROR'
8f56   2c           .byte $2c        nächsten Befehl überspringen
8f57   a2  00       ldx   #$00       Nummer für 'OK'
8f59   8e  b0  01   stx   $01b0      Status merken
8f5c   4c  61  88   jmp   $8861      Ende des Kommandos abwarten; Ende
-------------------------------------------------------------------------
8f5f                                 Routine, die die Sektorfolge für den QUERY- DISK-Befehl
                                     feststellt.
8f5f   08           php              Prozessorstatus retten
8f60   78           sei              Diskcontroller inaktivieren
8f61   20  ef  89   jsr   $89ef      Kopf auf Track 0 positionieren
8f64   24  3b       bit   $3b        soll eine andere Spur als 0 analysiert werden?
8f66   10  08       bpl   $8f70      verzweige, wenn nein
8f68   ad  03  02   lda   $0203      Sonst Tracknummer aus Befehlsstring holen
8f6b   85  67       sta   $67        und setzen
8f6d   20  ba  87   jsr   $87ba      Kopf auf anderen Track positionieren
8f70   a9  00       lda   #$00       Anzahl der Sektoren
8f72   85  97       sta   $97        löschen
8f74   20  27  8a   jsr   $8a27      Sektorheader lesen
8f77   ae  b0  01   ldx   $01b0      und Fehlerstatus holen
8f7a   e0  02       cpx   #$02       Fehler aufgetreten?
8f7c   b0  1f       bcs   $8f9d      verzweige, wenn ja
8f7e   a5  26       lda   $26        sonst Sektornummer holen
```

```
8f80    85  96       sta   $96         und merken
8f82    20  27  8a   jsr   $8a27       nächsten Sektorheader einlesen
8f85    a5  26       lda   $26         Sektornummer holen
8f87    a4  97       ldy   $97         erste Sektornummer als Zähler
8f89    99  0b  02   sta   $020b,y     neue Sektornummer in Befehlsstring
8f8c    e6  97       inc   $97         Zähler erhöhen
8f8e    c0  1f       cpy   #$1f        schon maximale Anzahl erreicht?
8f90    b0  0b       bcs   $8f9d       verzweige, wenn ja
8f92    c5  96       cmp   $96         sonst aktuelle Sektornummer mit erster
8f94    d0  ec       bne   $8f82       Sektornummer vergleichen; weiter, wenn ungleich
8f96    a5  24       lda   $24         sonst Tracknummer holen
8f98    85  67       sta   $67         und setzen
8f9a    a2  00       ldx   #$00        Nummer für 'OK'
8f9c    2c           .byte $2c         nächsten Befehl überspringen
8f9d    a2  02       ldx   #$02        Nummer für 'SECTOR NOT FOUND'
8f9f    8e  b0  01   stx   $01b0       Status setzen
8fa2    28           plp               Prozessorstatus wieder zurückholen
8fa3    60           rts               Ende
-------------------------------------------------------------------------
8fa4                                   UTILITY-Befehl 'S'. Abstand der Sektoren beim Schreiben und
                                       Lesen neu setzen ('sector interleave').
8fa4    ad  04  02   lda   $0204       Wert für Sektorabstand aus Befehlsstring
8fa7    85  69       sta   $69         setzen
8fa9    60           rts               Ende
-------------------------------------------------------------------------
8faa                                   UTILITY-Befehl 'R'. Anzahl der Leseversuche bei Auftreten
                                       eines Fehlers setzen.
8faa    ad  04  02   lda   $0204       Anzahl der Versuche aus Befehlsstring
8fad    85  6a       sta   $6a         setzen
8faf    60           rts               Ende
-------------------------------------------------------------------------
8fb0                                   UTILITY-Befehl 'T'. ROM-Prüfsumme testen.
8fb0    4c  4e  92   jmp   $924e       zur Berechnung der Prüfsumme; Ende
-------------------------------------------------------------------------
8fb3                                   UTILITY-Befehl 'H'. Diskettenseite wählen, wenn sich die
                                       Floppy im 1541-Modus befindet.
8fb3    78           sei               Diskcontroller inaktivieren
8fb4    ad  0f  18   lda   $180f       Steuerregister lesen
8fb7    29  20       and   #$20         Betriebsmodus testen
8fb9    d0  66       bne   $9021       verzweige, wenn 1571-Modus
8fbb    ad  04  02   lda   $0204       sonst Nummer der Diskettenseite aus Befehl
8fbe    c9  31       cmp   #$31        mit '1' vergleichen
8fc0    f0  12       beq   $8fd4       verzweige, wenn Seite 1 gewünscht
8fc2    c9  30       cmp   #$30        mit '0' vergleichen
8fc4    d0  5b       bne   $9021       Fehler, wenn ungleich 0
8fc6    ad  0f  18   lda   $180f       Steuerregister lesen
8fc9    29  fb       and   #$fb        Elektronik auf Seite 0 stellen
8fcb    8d  0f  18   sta   $180f       und setzen
8fce    58           cli               Diskcontroller wieder aktivieren
8fcf    24  3b       bit   $3b         testen, ob Diskette initialisiert werden soll
8fd1    10  0e       bpl   $8fe1       verzweige, wenn ja
8fd3    60           rts               sonst Ende
*1 8fd4 ad  0f  18   lda   $180f       Steuerregister lesen
*0 8fd4 4c  21  90   jmp   $9021       '31, SYNTAX ERROR' ausgeben; Ende
8fd7    09  04       ora   #$04        Elektronik auf Seite 1 stellen
8fd9    8d  0f  18   sta   $180f       und setzen
8fdc    58           cli               Diskcontroller wieder aktivieren
```

```
8fdd   24  3b       bit   $3b        soll Diskette initialisiert werden?
8fdf   30  03       bmi   $8fe4      verzweige, wenn nein
8fe1   4c  42  d0    jmp   $d042      Diskette initialisieren
8fe4   60           rts               Ende
-------------------------------------------------------------------------
8fe5                                  Steuerroutine der UTILITY-Befehle.
8fe5   ae  74  02    ldx   $0274      Länge des Befehlsstrings
8fe8   e0  04       cpx   #$04       kleiner als 4?
8fea   90  35       bcc   $9021      verzweige, wenn ja
8fec   ad  03  02    lda   $0203      sonst viertes Zeichen holen
8fef   c9  53       cmp   #$53       mit 'S' vergleichen
8ff1   f0  b1       beq   $8fa4      verzweige, wenn ja
8ff3   c9  52       cmp   #$52       mit 'R' vergleichen
8ff5   f0  b3       beq   $8faa      verzweige, wenn ja
8ff7   c9  54       cmp   #$54       mit 'T' vergleichen
8ff9   f0  b5       beq   $8fb0      verzweige, wenn ja
8ffb   c9  4d       cmp   #$4d       mit 'M' vergleichen
8ffd   f0  27       beq   $9026      verzweige, wenn ja
8fff   c9  48       cmp   #$48       mit 'H' vergleichen
9001   f0  b0       beq   $8fb3      verzweige, wenn ja
9003   a8           tay              sonst Wert in Y
9004   c0  04       cpy   #$04       mit 4 vergleichen (kleinste Gerätenummer)
9006   90  19       bcc   $9021      verzweige, wenn kleiner
9008   c0  1f       cpy   #$1f       mit 31 vergleichen (größte Gerätenummer)
900a   b0  15       bcs   $9021      verzweige, wenn größer
900c   a9  40       lda   #$40       Gerätenummer für TALK
900e   85  78       sta   $78        vorbereiten
9010   a9  20       lda   #$20       Gerätenummer für LISTEN
9012   85  77       sta   $77        vorbereiten
9014   98           tya              Nummer holen
9015   18           clc              und
9016   65  78       adc   $78        für TALK
9018   85  78       sta   $78        setzen
901a   98           tya              Nummer holen
901b   18           clc              und
901c   65  77       adc   $77        für LISTEN
901e   85  77       sta   $77        setzen
9020   60           rts              Ende

9021   a9  31       lda   #$31       Nummer der Fehlermeldung
9023   4c  c8  c1    jmp   $c1c8      '31, SYNTAX ERROR' ausgeben; Ende
-------------------------------------------------------------------------
9026                                  UTILITY-Befehl 'M'. Schaltet zwischen 1541- und 1571-Modus
                                      um.
9026   78           sei              Diskcontroller inaktivieren
9027   ad  04  02    lda   $0204      fünftes Zeichen aus Befehlsstring
902a   c9  31       cmp   #$31       mit '1' vergleichen
902c   f0  20       beq   $904e      verzweige, wenn 'M1'-Befehl
902e   c9  30       cmp   #$30       mit '0' vergleichen
9030   d0  ef       bne   $9021      verzweige, wenn ungleich '0'
9032   ad  0f  18    lda   $180f      sonst Steuerregister lesen
9035   29  df       and   #$df       und die Elektronik
9037   8d  0f  18    sta   $180f      auf 1 MHz schalten
903a   20  83  a4    jsr   $a483      ca. 70 Taktzyklen Verzögerung
903d   20  82  ff    jsr   $ff82      Flags für 1541-Modus setzen
9040   ad  af  02    lda   $02af      Flag für Umschaltung der
9043   09  80       ora   #$80       IRQ-Adresse auf 1541-Modus
9045   8d  af  02    sta   $02af      setzen
```

```
9048   58          cli             Diskcontroller wieder aktivieren
9049   24  3b      bit   $3b       soll die Diskette initialisiert werden?
904b   10  2f      bpl   $907c     verzweige, wenn ja
904d   60          rts             Ende
904e   ad  0f  18  lda   $180f     Steuerregister lesen
9051   09  20      ora   #$20      und die Elektronik
9053   8d  0f  18  sta   $180f     auf 2 MHz schalten
9056   20  83  a4  jsr   $a483     ca. 70 Taktzyklen Verzögerung
9059   a9  de      lda   #$de      Adresse der Jobschleife
905b   8d  a9  02  sta   $02a9     in $02a9/02aa auf die
905e   a9  9d      lda   #$9d      Adresse der 1571-Jobschleife $9dde
9060   8d  aa  02  sta   $02aa     setzen
9063   a9  40      lda   #$40      Timer auf 40 ms IRQ
9065   8d  07  1c  sta   $1c07     setzen und
9068   8d  05  1c  sta   $1c05     starten
906b   ad  af  02  lda   $02af     Flag für Umschaltung der
906e   29  7f      and   #$7f      IRQ-Adresse auf 1571-Modus
9070   8d  af  02  sta   $02af     setzen
9073   a9  00      lda   #$00      Adresse der aktuellen Stepperroutine
9075   85  62      sta   $62       löschen
9077   58          cli             Diskcontroller aktivieren
9078   24  3b      bit   $3b       soll die Diskette initialisiert werden?
907a   30  03      bmi   $907f     verzweige, wenn nein
907c   4c  42  d0  jmp   $d042     Diskette initialisieren; Ende
907f   60          rts             Ende
--------------------------------------------------------------------------------
9080                               FAST-LOAD-UTILITY. Routine ermöglicht das schnelle Laden
                                   einer Datei von Diskette.
9080   20  ce  81  jsr   $81ce     seriellen Bus auf Ausgang schalten
9083   20  ea  91  jsr   $91ea     Filenamen aufbereiten
9086   b0  5f      bcs   $90e7     verzweige, wenn Fehler
9088   20  3d  c6  jsr   $c63d     Drive ggf. initialisieren
908b   a5  ff      lda   $ff       Drivestatus holen
908d   d0  58      bne   $90e7     verzweige, wenn Fehler
908f   a5  37      lda   $37       zur Vorbereitung Flag für 2-MHz-Busbetrieb
9091   09  81      ora   #$81      und Flag für File hat nur 1 Block
9093   85  37      sta   $37       setzen
9095   20  ca  91  jsr   $91ca     Lesekana l öffnen
9098   ad  00  02  lda   $0200     erstes Zeichen aus Befehlsstring
909b   c9  2a      cmp   #$2a      mit '*' vergleichen
909d   d0  0f      bne   $90ae     verzweige, wenn ungleich
909f   a5  7e      lda   $7e       letzte Tracknummer holen
90a1   f0  0b      beq   $90ae     verzweige, wenn nicht vorhanden
90a3   48          pha             sonst Nummer merken
90a4   ad  6f  02  lda   $026f     letzte Sektornummer holen
90a7   8d  85  02  sta   $0285     und setzen
90aa   68          pla             Tracknummer zurückholen
90ab   4c  ec  90  jmp   $90ec     und Fi le laden
90ae   a9  00      lda   #$00      Flags und Werte löschen:
90b0   a8          tay             Y-Register löschen
90b1   aa          tax             X-Register löschen
90b2   8d  8e  02  sta   $028e     letzte Drivenummer
90b5   8d  7a  02  sta   $027a     Zeiger auf Filetabelle
90b8   20  12  c3  jsr   $c312     Drivenummer holen und setzen
90bb   ad  78  02  lda   $0278     Anzahl der Filenamen
90be   48          pha             merken
90bf   a9  01      lda   #$01      und jetzige Anzahl auf 1
90c1   8d  78  02  sta   $0278     setzen
```

```
90c4    a9  ff       lda   #$ff        Zeiger für Directorypuffer
90c6    85  86       sta   $86         setzen
90c8    20  4f  c4   jsr   $c44f       Datei im Directory suchen
90cb    68           pla               Anzahl der Filenamen zurückholen
90cc    8d  78  02   sta   $0278       und wieder merken
90cf    a5  37       lda   $37         Flag für 1571-Bus-Modus
90d1    29  7f       and   #$7f        löschen und
90d3    85  37       sta   $37         Flags wieder merken
90d5    24  3b       bit   $3b         muß gesuchtes File ein PRG-File sein?
90d7    30  06       bmi   $90df       verzweige, wenn nein
*1 90d9 a5  e7       lda   $e7         sonst Filetyp holen
*1 90db c9  02       cmp   #$02        und auf PRG-File testen
*0 90d9 20  5b  aa   jsr   $aa5b       Filetyp prüfen
*0 90dc ea           nop
90dd    d0  05       bne   $90e4       verzweige, wenn ungleich; Fehler
90df    ad  80  02   lda   $0280       Tracknummer des ersten Blocks im File
90e2    d0  08       bne   $90ec       verzweige, wenn vorhanden
90e4    a2  02       ldx   #$02        sonst Nummer für 'FILE NOT FOUND'
90e6    2c           .byte $2c         nächsten Befehl überspringen
90e7    a2  0f       ldx   #$0f        Nummer für 'DRIVE NOT READY'
90e9    4c  ad  91   jmp   $91ad       Fehlermeldung ausgeben; Ende
90ec    85  7e       sta   $7e         erste Tracknummer merken
90ee    48           pha               und merken
90ef    20  da  91   jsr   $91da       Pufferzeiger setzen
90f2    68           pla               Tracknummer zurückholen
90f3    ae  b0  02   ldx   $02b0       Zeiger in Jobspeicher
90f6    95  06       sta   $06,x       Tracknummer in Jobspeicher
90f8    ad  85  02   lda   $0285       Sektornummer holen
90fb    8d  6f  02   sta   $026f       und ebenfalls setzen
90fe    95  07       sta   $07,x       und in Jobspeicher
9100    a9  80       lda   #$80        Jobcode für Block lesen
9102    8d  02  02   sta   $0202       in Befehlsstring für Ausführung
9105    85  5f       sta   $5f         und als Jobzeiger
9107    58           cli               Diskcontroller aktivieren
9108    a6  f9       ldx   $f9         Puffernummer
910a    a5  5f       lda   $5f         Jobzeiger
910c    95  00       sta   $00,x       und Jobcode an Diskcontroller übergeben
910e    20  4b  86   jsr   $864b       Job ausführen und Ende abwarten
9111    e0  02       cpx   #$02        Fehler?
9113    90  03       bcc   $9118       verzweige, wenn nein
9115    4c  99  91   jmp   $9199       sonst zur Fehlerausgabe
9118    78           sei               Diskcontroller inaktivieren
9119    a0  00       ldy   #$00        Zeiger auf Tracknummer
911b    b1  94       lda   ($94),y     Byte aus Puffer holen
911d    f0  2f       beq   $914e       verzweige, wenn Tracknummer gleich 0
911f    a5  37       lda   $37         sonst Flag für File hat nur 1 Block
9121    29  fe       and   #$fe        löschen und
9123    85  37       sta   $37         Flags wieder merken
9125    20  28  92   jsr   $9228       aktuellen Burst-Status zum Computer
9128    a0  02       ldy   #$02        Zeiger auf erstes Datenbyte
912a    b1  94       lda   ($94),y     Byte holen
912c    aa           tax               und für Ausgabe setzen
912d    20  28  92   jsr   $9228       Byte zum Computer
9130    c8           iny               Zeiger auf nächstes Byte
9131    d0  f7       bne   $912a       und weitermachen
9133    ae  b0  02   ldx   $02b0       Zeiger in Jobspeicher
9136    b1  94       lda   ($94),y     Tracknummer aus Puffer
9138    d5  06       cmp   $06,x       gleich der aktuellen Tracknummer des Jobs?
913a    f0  03       beq   $913f       verzweige, wenn ja
```

```
913c   a0  80       ldy   #$80        sonst Jobcode für Block lesen
913e   2c           .byte $2c         nächsten Befehl überspringen
913f   a0  88       ldy   #$88        Jobcode für Block auf gleichem Track lesen
9141   84  5f       sty   $5f         setzen
9143   95  06       sta   $06,x       und Tracknummer setzen
9145   a0  01       ldy   #$01        Zeiger auf Sektornummer
9147   b1  94       lda   ($94),y     Byte aus Puffer holen
9149   95  07       sta   $07,x       und Sektornummer setzen
914b   4c  07  91    jmp   $9107       weitermachen; nächsten Sektor holen
914e   a2  1f       ldx   #$1f        Flag für letzten Block
9150   20  28  92    jsr   $9228       zum Computer
9153   a9  01       lda   #$01        Flag für File hat nur 1 Block
9155   24  37       bit   $37         prüfen
9157   f0  1e       beq   $9177       verzweige, wenn nicht gesetzt
9159   a8           tay               Zeiger auf Anzahl der Bytes im Block
915a   b1  94       lda   ($94),y     Byte holen
915c   38           sec               und
915d   e9  03       sbc   #$03        minus drei für Anzahl
915f   85  46       sta   $46         setzen
9161   aa           tax               Anzahl für Ausgabe setzen
9162   20  28  92    jsr   $9228       und zum Computer
9165   c8           iny               Zeiger auf nächstes Datenbyte
9166   b1  94       lda   ($94),y     Byte aus Puffer
9168   aa           tax               und
9169   20  28  92    jsr   $9228       zum Computer
916c   c8           iny               Zeiger auf nächstes Datenbyte
916d   b1  94       lda   ($94),y     Byte aus Puffer
916f   aa           tax               und
9170   20  28  92    jsr   $9228       zum Computer
9173   a0  04       ldy   #$04        Zeiger auf erstes Datenbyte bei 1-Block-File
9175   d0  0d       bne   $9184       unbedingter Sprung
9177   a0  01       ldy   #$01        Zeiger auf Anzahl der Datenbytes
9179   b1  94       lda   ($94),y     Anzahl holen
917b   aa           tax               und setzen
917c   ca           dex               minus 1
917d   86  46       stx   $46         merken
917f   20  28  92    jsr   $9228       Wert zum Computer
9182   a0  02       ldy   #$02        Zeiger auf erstes Datenbyte
9184   b1  94       lda   ($94),y     Byte holen
9186   aa           tax               und
9187   20  28  92    jsr   $9228       zum Computer
918a   c8           iny               Zeiger auf nächstes Byte
918b   c6  46       dec   $46         Zähler vermindern
918d   d0  f5       bne   $9184       und weitermachen
918f   a9  00       lda   #$00        interne Sekundäradresse für LOAD
9191   85  83       sta   $83         setzen und
9193   20  c0  da    jsr   $dac0       File schließen
9196   4c  94  c1    jmp   $c194       Diskstatus bereitstellen; Ende
-------------------------------------------------------------------------
9199                                  Routine sendet das Burst-Statusbyte zum Compu- ter und
                                      stellt eine entsprechende Fehlermeldung im Klartext bereit.
9199   78           sei               Diskcontroller inaktivieren
919a   86  46       stx   $46         Fehlernummer merken
919c   20  28  92    jsr   $9228       Burst-Status zum Computer
919f   a9  00       lda   #$00        interne Sekundäradresse für LOAD
91a1   85  83       sta   $83         setzen
91a3   20  c0  da    jsr   $dac0       File schließen
```

```
91a6   a6  f9       ldx   $f9        Puffernummer holen
91a8   a5  46       lda   $46        Fehlernummer holen
91aa   4c  0a  e6   jmp   $e60a      und Fehlermeldung bereitstellen; Ende
-------------------------------------------------------------------------
91ad                                 Routine zur Fehlerbeharlurg bei der Suche im Directory.
91ad   78           sei              Diskcontroller inaktivieren
91ae   86  46       stx   $46        Fehlernummer merken
91b0   a2  02       ldx   #$02       Nummer für 'FILE NOT FOU)'
91b2   20  28  92   jsr   $9228      als Burst-Status zum Computer
91b5   a9  00       lda   #$00       interne Sekundäradresse für L:A)
91b7   85  83       sta   $83        setzen und
91b9   20  c0  da   jsr   $dac0      File schließen
91bc   a5  46       lda   $46        Fehlernummer holen
91be   c9  02       cmp   #$02       mit 2 ('FILE NOT FOUND') vergleichen
91c0   f0  03       beq   $91c5      verzweige, wenn gleich
91c2   a9  74       lda   #$74       sonst Nummer für '74, DRIVE NOT READY'
91c4   2c           .byte $2c        nächsten Befehl überspringen
91c5   a9  62       lda   #$62       Nummer für '62, FILE NOT FOUND'
91c7   4c  c8  c1   jmp   $c1c8      Fehlermeldung ausgeben; Ende
-------------------------------------------------------------------------
91ca                                 Internen Lesekanal holen und belegen.
91ca   a9  00       lda   #$00       Sekundäradresse für LOAD
91cc   85  83       sta   $83        setzen
91ce   a9  01       lda   #$01       einen lesekanal
91d0   20  e2  d1   jsr   $d1e2      suchen und belegen
91d3   aa           tax              Pufternummer
91d4   bd  e0  fe   lda   $fee0,x    Pufferadresse Hi holen
91d7   85  95       sta   $95        und setzen
91d9   60           rts              Ende
-------------------------------------------------------------------------
91da                                 Pufferzeiger für Jobspeicher ermitteln.
91da   a5  95       lda   $95        Pufferadresse Hi
91dc   38           sec              minus 3
91dd   e9  03       sbc   #$03       ergibt
91df   85  f9       sta   $f9        die aktuelle Puffernummer
91e1   0a           asl              mal 2
91e2   8d  b0  02   sta   $02b0      ergibt den Zeiger in den Jobspeicher
91e5   a9  00       lda   #$00       Pufteradresse Lo
91e7   85  94       sta   $94        setzen
91e9   60           rts              Ende
-------------------------------------------------------------------------
91ea                                 Eingabezeile auf Drivenummer und ':' prüfen und
                                     überflüssige Angaben durch Verschieben des Filenamens an
                                     den Anfang löschen.
91ea   a0  03       ldy   #$03       Zeiger auf Beginn des Filenamens
91ec   ad  74  02   lda   $0274      Anzahl der Zeichen im Befehlsstring
91ef   38           sec              minus 3
91f0   e9  03       sbc   #$03       ergibt
91f2   8d  74  02   sta   $0274      die Länge des Filenamens
91f5   ad  04  02   lda   $0204      fünftes Zeichen aus Befehlsstring
91f8   c9  3a       cmp   #$3a       mit ':' vergleichen
91fa   d0  0e       bne   $920a      verzweige, wenn ungleich
91fc   ad  03  02   lda   $0203      viertes Zeichen aus Befehlsstring
91ff   aa           tax              merken
9200   29  30       and   #$30       eventuell vorhandene Nummer isolieren
9202   c9  30       cmp   #$30       und prüfen
9204   d0  04       bne   $920a      verzweige, wenn ungleich Drivenummer 0
```

```
9206   e0  31       cpx   #$31        sonst auf Drivenummer 1 testen
9208   f0  1c       beq   $9226       Fehler, wenn ja
920a   ad  03  02    lda   $0203       viertes Zeichen aus Befehlsstring
920d   c9  3a        cmp   #$3a        mit ':' vergleichen
920f   d0  04        bne   $9215       verzweige, wenn ungleich
9211   ce  74  02    dec   $0274       Länge des Filenamen vermindern
9214   c8            iny               Zeiger in Befehlsstring erhöhen
9215   a2  00        ldx   #$00        zweiten Index setzen
9217   b9  00  02    lda   $0200,y     und
921a   9d  00  02    sta   $0200,x     Filename an den Anfang verschieben
921d   c8            iny               Zeiger
921e   e8            inx               auf nächstes Zeichen
921f   ec  74  02    cpx   $0274       schon alle Zeichen verschoben?
9222   d0  f3        bne   $9217       weitermachen, wenn nein
9224   18            clc               Flag für alles ok
9225   24            .byte $24         nächsten Befehl überspringen
9226   38            sec               Flag für Fehler bei Drivenummer
9227   60            rts               Ende
--------------------------------------------------------------------------
9228                                   Inhalt des X-Registers im Fast-Modus zum Computer schicken.
9228   ad  00  18    lda   $1800       Bus lesen
922b   cd  00  18    cmp   $1800       und konstanten Wert
922e   d0  f8        bne   $9228       abwarten
9230   29  ff        and   #$ff        auf ATN-Signal testen
9232   30  17        bmi   $924b       verzweige, wenn gesetzt
9234   45  37        eor   $37         Flags für Busbetrieb
9236   29  04        and   #$04         CLOCK-Zustand prüfen
9238   f0  ee        beq   $9228       warten, wenn nicht gegeben
923a   8e  0c  40    stx   $400c       sonst Byte ins serielle Schieberegister
923d   a5  37        lda   $37         Flags für Busbetrieb holen
923f   49  04        eor   #$04        CLOCK-Zustand invertieren
9241   85  37        sta   $37         und für nächsten Zugriff setzen
9243   a9  08        lda   #$08        Flag für Schieberegister
9245   2c  0d  40    bit   $400d       warten, bis
9248   f0  fb        beq   $9245       Byte zum Computer gesendet ist
924a   60            rts               Ende
924b   4c  b3  a7    jmp   $a7b3       zur Bedienung bei ATN; Ende
--------------------------------------------------------------------------
924e                                   Routine berechnet die Prüfsumme des ROM als Folge des 'T'-
                                       Kommandos. Bei Übereinstimmung mit dem Wert bei $8000/8001
                                       erfolgt der Rück, sprung mit RTS; sonst ein Sprung nach
                                       $ea71 (Fehlerblinken bei Hardwaredefekten).
924e   08            php               Prozessorstatus retten
924f   78            sei               Diskcontroller inaktivieren
9250   a2  00        ldx   #$00        Jobspeicher
9252   86  00        stx   $00         löschen
9254   86  01        stx   $01         löschen
9256   a9  03        lda   #$03        ROM-Startadresse Lo
9258   85  75        sta   $75         setzen
925a   a8            tay               Zeiger merken
925b   a9  80        lda   #$80        ROM-Startadresse Hi
925d   85  76        sta   $76         setzen
925f   b1  75        lda   ($75),y     Byte aus ROM holen
9261   85  02        sta   $02         und merken
9263   a2  08        ldx   #$08        Bitzähler setzen (8 Bits pro Byte)
9265   a5  02        lda   $02         Byte aus ROM holen
```

```
9267   29  01       and   #$01       und Bit isolieren
9269   85  03       sta   $03        Bit merken
926b   a5  01       lda   $01        Bit 15 (Bit 7 des H;-3ye)
926d   10  02       bpl   $9271      zu dem geholten Byte
926f   e6  03       inc   $03        addieren
9271   6a           ror               Bit 8 (Bit 0 des Hi-8yte)
9272   90  02       bcc   $9276      ebenfalls zum geholte Bye
9274   e6  03       inc   $03        addieren
9276   6a           ror               jetzt das
9277   6a           ror               Bit 11 des Zwischenspeic1ers
9278   6a           ror               ebenfalls
9279   90  02       bcc   $927d      zu dem geholten Byte
927b   e6  03       inc   $03        addieren
927d   a5  00       lda   $00        Lo-Byte des Prüfsummenzisc,e-re'sters
927f   2a           rol               Bit 6
9280   2a           rol               isolieren und
9281   90  02       bcc   $9285      zum Bytewert aus dem R
9283   e6  03       inc   $03        addieren
9285   66  03       ror   $03        Bit 0 des Bytewertes aus de R
9287   26  00       rol   $00        über das Carry-Bit ins Bit 0 des
9289   26  01       rol   $01        Zwischenspeichers rollen
928b   66  02       ror   $02        nächstes Bit des ROM-Bytes
928d   ca           dex               Zähler auf nächstes Bit des ROM-Bytes
928e   d0  d5       bne   $9265      und weitermachen
9290   e6  75       inc   $75        Adresse Lo erhöhen
9292   d0  cb       bne   $925f      und nächstes Byte holen
9294   e6  76       inc   $76        ROM-Adresse Hi erhöhen
9296   d0  c7       bne   $925f      und nächstes Byte holen
9298   88           dey               Zeiger von ert 3
9299   88           dey               auf den ert 0
929a   88           dey               vermindern
929b   a5  00       lda   $00        Lo-Byte der ROM-Summe
929d   cd  00  80    cmp   $8000      mit abgespeicherter Prüfsumme vergleichen
92a0   d0  11       bne   $92b3      verzweige, wenn ungleich
92a2   a5  01       lda   $01        gleich, dann Hi-Byte ebenfalls
92a4   cd  01  80    cmp   $8001      testen
92a7   d0  0a       bne   $92b3      Fehler, wenn ungleich
92a9   84  00       sty   $00        sonst Zwischenspeicher
92ab   84  01       sty   $01        wieder
92ad   84  02       sty   $02        löschen und für
92af   84  03       sty   $03        Jobs in definierten Zustand bringen
92b1   28           plp               Prozessorstatus wieder zurückholen
92b2   60           rts               und Ende; alles ok
92b3   a2  03       ldx   #$03       Nummer des Fehlers
92b5   86  6f       stx   $6f        setzen
92b7   4c  71  ea    jmp   $ea71      zum Hardwareblinken der LED
```

```
--------------------------------------------------------------------------
92ba                                    Diskcontroller-Routine (Jobschleife) für den 1571-Modus.
                                        Diese Routine befindet sich analog für den 1541-Modus bei
                                        $f2b0.
92ba   ba          tsx                  Stackpointer
92bb   86  49      stx   $49            merken
92bd   2c  04  1c  bit   $1c04          IRQ-Flag durch lesen löschen
92c0   ad  0c  1c  lda   $1c0c          BYTE-READY-leitung durch
92c3   09  0e      ora   #$0e           setzen der Bits 1, 2 und 3
92c5   8d  0c  1c  sta   $1c0c          initialisieren
92c8   a0  05      ldy   #$05           Index in Jobspeicher setzen
92ca   b9  00  00  lda   $0000,y        liegt Job für Puffer an?
92cd   30  06      bmi   $92d5          verzweige, wenn" ja
92cf   88          dey                  Index auf nächsten Puffer
92d0   10  f8      bpl   $92ca          weitermachen
92d2   4c  ca  99  jmp   $99ca          zur Routine für Motor- und Steppersteuerung
92d5   c9  88      cmp   #$88           Kommando für Sektor auf gleichem Track lesen?
92d7   d0  03      bne   $92dc          verzweige, wenn nein
92d9   4c  0d  96  jmp   $960d          sonst Kommando ausführen
92dc   c9  d0      cmp   #$d0           Kommando für Programm im Puffer ausführen?
92de   d0  03      bne   $92e3          verzweige, wenn nein
92e0   4c  a2  93  jmp   $93a2          sonst Kommando ausführen
92e3   29  01      and   #$01           Drivenummer aus Jobcode isolieren
92e5   f0  07      beq   $92ee          verzweige, wenn Drive 0
92e7   84  3f      sty   $3f            sonst Puffernummer setzen und
92e9   a9  0f      lda   #$0f           Nummer für Fehlermeldung setzen
92eb   4c  b5  99  jmp   $99b5          '74, DRIVE NOT READY' ausgeben
92ee   aa          tax                  'Drivenummer merken
92ef   c5  3e      cmp   $3e            und mit aktivem Drive vergleichen
92f1   f0  08      beq   $92fb          verzweige, wenn gleich
92f3   85  3e      sta   $3e            sonst Nummer neu setzen
92f5   20  7e  f9  jsr   $f97e          und Drivemotor einschalten
92f8   4c  ca  99  jmp   $99ca          weiter zur Steppermotorsteuerung
92fb   a5  20      lda   $20            Flags für Drivestatus
92fd   30  03      bmi   $9302          verzweige, wenn Drive nicht bereit für Zugriff
92ff   0a          asl                  sonst: Steppermotor in Aktion?
9300   10  03      bpl   $9305          verzweige, wenn nein
9302   4c  ca  99  jmp   $99ca          ja; zur Jobschleife für Steppermotor
9305   a9  20      lda   #$20           Flag für Drive an und bereit
9307   85  20      sta   $20            setzen
9309   a0  05      ldy   #$05           Index in Jobspeicher
930b   84  3f      sty   $3f            setzen
930d   20  d1  93  jsr   $93d1          Pufferadresse setzen; Jobcode holen
9310   30  1a      bmi   $932c          verzweige, wenn Job anliegt
9312   c6  3f      dec   $3f            sonst Zeiger auf nächsten Puffer
9314   10  f7      bpl   $930d          und weiterprüfen
9316   a4  41      ldy   $41            Puffernummer für nächsten Job
9318   20  d3  93  jsr   $93d3          Pufferadresse für Job setzen
931b   a5  42      lda   $42            Trackdifferenz zu letztem Job
931d   85  4a      sta   $4a            setzen
931f   06  4a      asl   $4a            mal 2 ergibt Anzahl der Schritte
9321   a9  60      lda   #$60           Flags für Drive bereit und Stepper in Aktion
9323   85  20      sta   $20            setzen
9325   b1  32      lda   ($32),y        Tracknummer holen
9327   85  22      sta   $22            und für Job setzen
9329   4c  ca  99  jmp   $99ca          zur Jobschleife für Kopfpositionierung
932c   29  01      and   #$01           Drivenummer für Job
932e   c5  3e      cmp   $3e            mit Nummer des letzten Jobs vergleichen
```

```
9330   d0  e0       bne   $9312       verzweige, wenn ungleich
9332   a5  22       lda   $22         aktuelle Tracknummer des Jobs
9334   f0  32       beq   $9368       verzweige, wenn nicht gesetzt
9336   a5  22       lda   $22         sonst Nummer holen
9338   c9  24       cmp   #$24        und mit Maximum (36) vergleichen
933a   08           php               Ergebnis merken
933b   b1  32       lda   ($32),y     Tracknummer für Job
933d   c9  24       cmp   #$24        mit Maximum (36) vergleichen
933f   6a           ror               und Ergebnis in Akku (Bit 7)
9340   28           plp               letztes Ergebnis
9341   29  80       and   #$80        Ergebnis prüfen
9343   90  0b       bcc   $9350       verzweige, wenn letztes Ergebnis kleiner 36
9345   30  11       bmi   $9358       verzweige, wenn jetziges Ergebnis größer 36
9347   a5  22       lda   $22         Tracknummer für Job
9349   e9  23       sbc   #$23        minus 35 (Wert für Seite 1 der Diskette)
934b   85  22       sta   $22         setzen
934d   4c  58  93    jmp   $9358       und Tracknummer weiter beerbe;ten
9350   10  06       bpl   $9358       verzweige, wenn neuer Track auf Seite 0
9352   a5  22       lda   $22         aktuelle Tracknummer
9354   69  23       adc   #$23        plus 35 für
9356   85  22       sta   $22         Seite 1 setzen
9358   38           sec               und
9359   b1  32       lda   ($32),y     Tracknummer
935b   e5  22       sbc   $22         minus Differenzwert; gleicher Track für Kopf?
935d   f0  09       beq   $9368       verzweige, wenn ja
935f   85  42       sta   $42         sonst Differenz für Positionierung merken
9361   a5  3f       lda   $3f         Puffernummer holen und
9363   85  41       sta   $41         für Job setzen
9365   4c  12  93    jmp   $9312       weitermachen; für nächsten Job
9368   a2  04       ldx   #$04        (überflüssiger Befehl)
936a   b1  32       lda   ($32),y     Tracknummer für Job holen
936c   85  40       sta   $40         und setzen
936e   c9  24       cmp   #$24        mit Maximum (36) vergleichen
9370   a8           tay               und Tracknummer merken
9371   20  f3  93    jsr   $93f3       Hardware für entsprechende Diskseite setzen
9374   98           tya               Tracknummer
9375   90  02       bcc   $9379       verzweige, wenn kleiner 36
9377   e9  23       sbc   #$23        sonst minus 35 für Stepper
9379   aa           tax               und merken
937a   bd  08  94    lda   $9408,x     Wert für Timersteuerung des Tracks holen
937d   85  43       sta   $43         und setzen
937f   ad  00  1c    lda   $1c00       Steuerport für Diskcontroller lesen
9382   29  9f       and   #$9f        Timerbits löschen
9384   05  43       ora   $43         und neuen Wert für Track setzen
9386   8d  00  1c    sta   $1c00       an Diskcontroller übergeben
9389   bd  2b  94    lda   $942b,x     Anzahl der Sektoren des Tracks holen
938c   85  43       sta   $43         und setzen
938e   a5  45       lda   $45         Jobcode
9390   c9  40       cmp   #$40        mit Wert für 'BUMP' vergleichen
9392   f0  1c       beq   $93b0       ja; BUMP ausführen; Kopf auf Track 0 setzen
9394   c9  60       cmp   #$60        mit Wert für Jobprogamm ausführen vergleichen
9396   f0  0a       beq   $93a2       ja; Programm ausführen
9398   c9  70       cmp   #$70        mit Wert für Formatieren vergleichen
939a   f0  03       beq   $939f       ja; Diskette formatieren
939c   4c  4f  94    jmp   $944f       sonst SEEK ausführen; Blockheader lesen
939f   4c  29  9b    jmp   $9b29       Diskette formatieren
```

```
--------------------------------------------------------------------------
93a2                                    Routine setzt die Pufferadresse und startet ein Benutzer-
                                        Jobprogramm im Puffer. Analog zu $f36e.
93a2   a5  3f       lda   $3f           Puffernummer
93a4   18           clc                 holen und
93a5   69  03       adc   #$03          3 addieren
93a7   85  31       sta   $31           ergibt Pufferadresse Hi
93a9   a9  00       lda   #$00          Pufferadresse Lo gleich 0
93ab   85  30       sta   $30           setzen
93ad   6c  30  00   jmp   ($0030)       Programm starten
--------------------------------------------------------------------------
93b0                                    Routine führt einen BUMP aus, d.h. der Kopf wird auf Track
                                        0 zurückgefahren. Analog zu $f37c.
93b0   a9  60       lda   #$60          Flags für Drive und Stepper in Aktion
93b2   85  20       sta   $20           setzen
93b4   ad  00  1c   lda   $1c00         Steuerport für Diskcontroller
93b7   29  fc       and   #$fc          Bits für Steppermotor löschen
93b9   8d  00  1c   sta   $1c00         und neuen Status setzen
93bc   a9  a4       lda   #$a4          Anzahl der Schritte
93be   85  4a       sta   $4a           zum Zurückfahren des Kopfes (-45 Tracks)
93c0   ad  b1  01   lda   $01b1         Diskettenseite holen
93c3   30  03       bmi   $93c8         verzweige, wenn Seite 1
93c5   a9  01       lda   #$01          sonst Tracknummer 1
93c7   2c           .byte $2c           nächsten Befehl überspringen
93c8   a9  24       lda   #$24          Tracknummer 36 für Seite 1
93ca   85  22       sta   $22           setzen
93cc   a9  01       lda   #$01          Nummer für 'OK'
93ce   4c  b5  99   jmp   $99b5         und Jobschleife beenden
--------------------------------------------------------------------------
93d1                                    Pufferadresse und Pufferzeiger für jeden Job setzen; in
                                        $30/31 und $32. Analog zu $f393.
93d1   a4  3f       ldy   $3f          Puffernummer des Jobs
93d3   b9  00  00   lda   $0000,y      Jobcode holen
93d6   48           pha                 und merken
93d7   10  14       bpl   $93ed         verzweige, wenn kein Job anliegt
93d9   29  78       and   #$78          sonst Bits für Jobcode isolieren
93db   85  45       sta   $45           und Code merken
93dd   98           tya                 Puffernummer
93de   0a           asl                 mal 2 und
93df   69  06       adc   #$06          plus 6 ergibt
93e1   85  32       sta   $32           Adresse Hi in Jobspeicher für Track und Sektor
93e3   a9  00       lda   #$00          Adresse Lo
93e5   85  33       sta   $33           für Track und Sektornummer
93e7   98           tya                 Puffernummer
93e8   18           clc                 plus 3
93e9   69  03       adc   #$03          ergibt
93eb   85  31       sta   $31           aktuelle Pufferadresse Hi für Job
93ed   a0  00       ldy   #$00          Pufferadresse Lo gleich 0
93ef   84  30       sty   $30           setzen
93f1   68           pla                 Jobcode zurückholen
93f2   60           rts                 Ende
```

```
--------------------------------------------------------------------------
93f3                                    Kontrollregister entsprechend Diskettensei- te 0 oder 1
                                        setzen. um damit die Hardware der Schreib-/Leseelektronik
                                        zu initialisieren. Das Carry-Flag gibt dabei die
                                        Diskettenseite an (Carry = 1 heißt Seite 1).
93f3   b0  03       bcs   $93f8       verzweige, wenn Seite 1
93f5   a9  00       lda   #$00        Wert für Seite 0
93f7   2c           .byte $2c         nächsten Befehl überspringen
93f8   a9  84       lda   #$84        Wert für Seite 1
93fa   8d  b1  01   sta   $01b1       setzen
93fd   ad  0f  18   lda   $180f       Steuerregister lesen
9400   29  fb       and   #$fb        Bits für Hardwareorganisation löschen
9402   0d  b1  01   ora   $01b1       und mit Stellwert verknüpfen
9405   8d  0f  18   sta   $180f       Hardware setzen
9408   60           rts               Ende
--------------------------------------------------------------------------
9409                                    Werte für die Timersteuerung beim Schreiben und Lesen der 4
                                        unterschiedlichen Spurbereiche einer Diskette im GCR-
                                        Format.
9409 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60 60
9419 60 40 40 40 40 40 40 40 20 20 20 20 20 20 00 00
9429 00 00 00
--------------------------------------------------------------------------
942c                                    Anzahl der Sektoren pro Spur.
942c 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15
943c 15 13 13 13 13 13 13 13 12 12 12 12 12 12 11 11
944c 11 11 11
--------------------------------------------------------------------------
944f                                    Routine zum Suchen eines Blockheaders auf einem beliebigen
                                        Track ('SEEK'). Diese Routine ist analog bei $f3b1 für die
                                        1541 vorhanden.
944f   a9  5a       lda   #$5a        90; Anzahl der Versuche
9451   85  4b       sta   $4b         setzen
9453   20  54  97   jsr   $9754       SYNC-Signal abwarten
9456   2c  0f  18   bit   $180f       warten, bis
9459   30  fb       bmi   $9456       Byte eingelesen
945b   ad  01  1c   lda   $1c01       Byte vom Diskcontroller holen
945e   c9  52       cmp   #$52        und mit Kennzeichen für Blockheader vergleichen
9460   d0  3e       bne   $94a0       verzweige, wenn kein Blockheader
9462   99  24  00   sta   $0024,y     sonst Byte merken
9465   c8           iny               und Zeiger auf nächstes GCR-Byte setzen
9466   2c  0f  18   bit   $180f       warten, bis
9469   30  fb       bmi   $9466       Byte eingelesen
946b   ad  01  1c   lda   $1c01       Byte vom Diskcontroller holen
946e   99  24  00   sta   $0024,y     und abspei ehern
9471   c8           iny               nächstes Byte
9472   c0  08       cpy   #$08        bis alle 8 Bytes eines Blockheaders
9474   d0  f0       bne   $9466       gelesen worden sind
9476   20  2f  95   jsr   $952f       Blockheader von GCR in Binär umwandeln
9479   a0  04       ldy   #$04        Index in Blockheader setzen
947b   a9  00       lda   #$00        Anfangswert für Prüfsumme
947d   59  16  00   eor   $0016,y     und Prüfsumme über Blockheader berechnen
9480   88           dey               nächstes Byte
9481   10  fa       bpl   $947d       und weitermachen
9483   c9  00       cmp   #$00        hebt sich die Prüfsumme mit der echten auf?
9485   d0  30       bne   $94b7       Fehler, wenn nein
9487   a5  18       lda   $18         sonst Tracknummer des Headers
```

```
9489   85  22       sta   $22        übernehmen
948b   a5  45       lda   $45        und Jobcode
948d   c9  30       cmp   #$30       auf Kommando für 'SEEK' prüfen
948f   f0  18       beq   $94a9      Ende, wenn ja
9491   a5  12       lda   $12        sonst ID 1
9493   c5  16       cmp   $16        mit ID 1 des Blockheaders vergleichen
9495   d0  1d       bne   $94b4      Fehler, wenn ungleich
9497   a5  13       lda   $13        ID 2 mit
9499   c5  17       cmp   $17        ID 2 des Blockheaders vergleichen
949b   d0  17       bne   $94b4      Fehler, wenn ungleich
949d   4c  bc  94    jmp   $94bc      nächsten Job für Diskette holen
94a0   c6  4b       dec   $4b        Zähler für Leseversuche vermindern
94a2   d0  af       bne   $9453      weitermachen, wenn noch nicht Null
94a4   a9  02       lda   #$02       Nummer für '20, READ ERROR'
94a6   20  b5  99    jsr   $99b5      Fehler ausgeben
94a9   a5  16       lda   $16        ID 1 aus Blockheader
94ab   85  12       sta   $12        übernehmen
94ad   a5  17       lda   $17        ID 2 aus Blockheader
94af   85  13       sta   $13        übernehmen
94b1   a9  01       lda   #$01       Nummer für 'OK'
94b3   2c           .byte $2c        nächsten Befehl überspringen
94b4   a9  0b       lda   #$0b       Nummer für '29, DISK ID MISMATCH'
94b6   2c           .byte $2c        nächsten Befehl überspringen
94b7   a9  09       lda   #$09       Nummer für '27, READ ERROR'
94b9   4c  b5  99    jmp   $99b5      Meldungen ausgeben; Ende der Jobschleife
------------------------------------------------------------------------------
94bc                                 Sucht den Job, der mit der geringsten Bewegung des Schreib-
                                     /Lesekopfes und damit mit dem geringsten Aufwand bearbeitet
                                     werden kann. Analog zu $f423 der 1541.
94bc   a9  7f       lda   #$7f       Differenzwert für Sektornummer
94be   85  4c       sta   $4c        vorbesetzen
94c0   a5  19       lda   $19        Sektornummer aus Blockheader
94c2   18           clc              holen und
94c3   69  02       adc   #$02       2 addieren
94c5   c5  43       cmp   $43        danach mit Maximum vergleichen
94c7   90  02       bcc   $94cb      verzweige, wenn kleiner
94c9   e5  43       sbc   $43        sonst maximale Sektornummer abziehen, um
94cb   85  4d       sta   $4d        den Nulldurchgang zu bekommen
94cd   a2  05       ldx   #$05       Maximale Puffernummer
94cf   86  3f       stx   $3f        setzen
94d1   a2  ff       ldx   #$ff       zugehörige Pufferadresse
94d3   20  d1  93    jsr   $93d1      setzen
94d6   10  43       bpl   $951b      verzweige, wenn kein Jobcode vorhanden
94d8   29  01       and   #$01       Drivenummer isolieren
94da   c5  3e       cmp   $3e        und mit aktueller Nummer vergleichen
94dc   d0  3d       bne   $951b      verzweige, wenn ungleich
94de   a0  00       ldy   #$00       sonst Zeiger auf Tracknummer
94e0   b1  32       lda   ($32),y    und Tracknummer für Puffer holen
94e2   c5  40       cmp   $40        und mit aktueller Spur vergleichen
94e4   d0  35       bne   $951b      verzweige, wenn ungleich
94e6   a5  45       lda   $45        Jobcode holen
94e8   c9  60       cmp   #$60       und auf Job für Programm ausführen testen
94ea   f0  0c       beq   $94f8      verzweige, wenn ja
94ec   a0  01       ldy   #$01       sonst Zeiger auf Sektornummer des Jobs
94ee   38           sec              und
94ef   b1  32       lda   ($32),y    Sektornummer für Job
94f1   e5  4d       sbc   $4d        auf optimalen Wert prüfen
```

```
94f3   10  03       bpl   $94f8       verzweige, wenn ja
94f5   18            clc              sonst
94f6   65  43       adc   $43         Maximalanzahl der Sektoren abziehen
94f8   c5  4c       cmp   $4c         und mit Differenz vergleichen
94fa   b0  1f       bcs   $951b       verzweige, wenn größer
94fc   48            pha              Ergebnis merken
94fd   a5  45       lda   $45         Jobcode holen
94ff   f0  15       beq   $9516       verzweige, wenn Job fur Sektor lesen
9501   68            pla              sonst Ergebnis zurückholen
9502   c9  09       cmp   #$09        und mit 9 vergleichen
9504   90  15       bcc   $951b       verzweige, wenn kleiner
9506   c9  0c       cmp   #$0c        mit 12 vergleichen
9508   b0  11       bcs   $951b       verzweige, wenn größer
950a   85  4c       sta   $4c         neue Differenz merken
950c   a5  3f       lda   $3f         Puffernummer
950e   aa            tax              als Index nehmen
950f   18            clc              und
9510   69  03       adc   #$03        Pufferadresse Hi erre:hen
9512   85  31       sta   $31         und setzen
9514   d0  05       bne   $951b       unbedingter Sprung
9516   68            pla              Differenz zurückholen
9517   c9  06       cmp   #$06        mit 6 vergleichen
9519   90  ef       bcc   $950a       verzweige, wenn kleiner
951b   c6  3f       dec   $3f         sonst Puffernummer vermindern
951d   10  b4       bpl   $94d3       und weitermachen, wenn noch Puffer vorhanden
951f   8a            txa              Puffernummer
9520   10  03       bpl   $9525       verzweige, wenn günstiger Job für diesen Puffer
9522   4c  ca  99    jmp   $99ca       sonst zur Jobschleife für Stepperbedienung
9525   86  3f       stx   $3f         Puffernummer setzen
9527   20  d1  93    jsr   $93d1       und Pufferadresse setzen; Jobcode holen
952a   a5  45       lda   $45         Jobcode
952c   4c  06  96    jmp   $9606       Jobcode prüfen; ggf. ausführen; Ende
-------------------------------------------------------------------------------
952f                                   Blockheader von GCR-(5 Bit)-Format ins Binär-(4 Bit)-Format
                                       umrechnen. Analog zu $f497. Der GCR-codierte Header steht
                                       dabei ab $24; das Ergebnis wird ab $16 bis $1a abgelegt und
                                       zwar:
                                       $16 - ID 1 der Diskette
                                       $17 - ID 2 der Diskette
                                       $18 - Tracknummer des Blocks
                                       $19 - Sektornummer des Blocks
                                       $1a - Prüfsumme über den Blockheader
952f   a5  30       lda   $30         Pufferadresse Lo
9531   48            pha              merken
9532   a5  31       lda   $31         Pufferadresse Hi
9534   48            pha              merken
9535   a9  24       lda   #$24        Pufferadresse Lo auf $24
9537   85  30       sta   $30         setzen
9539   a9  00       lda   #$00        Pufferadresse Hi auf $00
953b   85  31       sta   $31         setzen; ergibt $0024
953d   a9  00       lda   #$00        Pufferzeiger
953f   85  34       sta   $34         setzen
9541   20  d9  98    jsr   $98d9       ersten Teil des Blockheaders konvertieren
9544   a5  55       lda   $55         Tracknummer
9546   85  18       sta   $18         holen und setzen
9548   a5  54       lda   $54         Sektornummer
954a   85  19       sta   $19         holen und setzen
954c   a5  53       lda   $53         Prüfsumme
```

```
954e   85  1a       sta   $1a         holen und setzen
9550   20  d9  98   jsr   $98d9       zweiten Teil des Blockheaders konvertieren
9553   a5  52       lda   $52         10 2 holen
9555   85  17       sta   $17         und setzen
9557   a5  53       lda   $53         ID 1 holen
9559   85  16       sta   $16         und setzen
955b   68           pla               Pufferadresse Hi zurückholen
955c   85  31       sta   $31         und wieder setzen
955e   68           pla               Pufferadresse Lo zurückholen
955f   85  30       sta   $30         und wieder setzen
9561   60           rts               Ende
--------------------------------------------------------------------------
9562 ff ...                           unbenutzter
95ff  ... ff                          Leerbereich
--------------------------------------------------------------------------
9600                                  Sucht nach einem Blockheader und wartet das SYNC-Signal des
                                      folgenden Datenblocks ab. Analog zu $f50a der 1541.
9600   20  0f  97   jsr   $970f       Blockheader suchen
9603   4c  54  97   jmp   $9754       SYNC-Signal abwarten
--------------------------------------------------------------------------
9606                                  Prüft auf Jobcode für Lesen und liest ggf. den gewünschten
                                      GCR-Datenblock ein. Analog zu $f4ca
9606   c9  00       cmp   #$00        Jobcode für Block lesen?
9608   f0  03       beq   $960d       verzweige, wenn ja
960a   4c  6e  97   jmp   $976e       sonst weiter prüfen
960d   20  00  96   jsr   $9600       gewünschten Blockheader suchen
9610   2c  0f  18   bit   $180f       warten, bi s
9613   30  fb       bmi   $9610       Byte eingelesen
9615   ad  01  1c   lda   $1c01       Byte vom Diskcontroller holen
9618   aa           tax               und merken
9619   bd  0d  a0   lda   $a00d,x     Binäräquivalent zu GCR-Nibble holen
961c   85  52       sta   $52         und merken
961e   8a           txa               GCR-Byte zurückholen
961f   29  07       and   #$07        Teil des zweiten GCR-Nibbles isolieren
9621   85  53       sta   $53         und merken
9623   2c  0f  18   bit   $180f       warten, bis
9626   30  fb       bmi   $9623       Byte eingelesen
9628   ad  01  1c   lda   $1c01       Byte vom Diskcontroller holen
962b   85  54       sta   $54         und merken
962d   29  c0       and   #$c0        obere 2 Bits isolieren
962f   05  53       ora   $53         und zu unteren 3 Bits summieren
9631   aa           tax               Wert merken und
9632   bd  0d  9f   lda   $9f0d,x     Binäräquivalent holen
9635   05  52       ora   $52         mit erstem Wert verknüpfen
9637   48           pha               und merken (ergibt Datenblockkennzeichen)
9638   4c  67  96   jmp   $9667       restlichen Datenblock einlesen

963b   2c  0f  18   bit   $180f       warten, bis
963e   30  fb       bmi   $963b       Byte eingelesen
9640   ad  01  1c   lda   $1c01       Byte vom Diskcontroller holen
9643   aa           tax               und merken
9644   bd  0d  a0   lda   $a00d,x     Binäräquivalent holen
9647   85  52       sta   $52         und merken
9649   8a           txa               Byte zurückholen
964a   29  07       and   #$07        Lo-Teil isolieren; für 2. GCR-Nibble
964c   85  53       sta   $53         und merken
964e   2c  0f  18   bit   $180f       warten, bis
```

```
9651   30  fb       bmi   $964e       Byte eingelesen
9653   ad  01  1c   lda   $1c01       Byte vom Diskcontroller holen
9656   85  54       sta   $54         und merken
9658   29  c0       and   #$c0        obere 2 Bits isoliere.
965a   05  53       ora   $53         2. GCR-Nibble bilce
965c   aa           tax               und
965d   bd  0d  9f   lda   $9f0d,x     Binäräquivalent holen
9660   05  52       ora   $52         mit erstem Teil verknüpfen
9662   91  30       sta   ($30),y     vollständiges Byte im Puffer ablegen
9664   c8           iny               Zeiger auf nächstes Byte
9665   f0  70       beq   $96d7       verzweige, wenn fertig
9667   a5  54       lda   $54         sonst Byte holen
9669   aa           tax               und merken
966a   bd  0d  a1   lda   $a10d,x     Binäräquivalent holen
966d   85  52       sta   $52         und merken
966f   8a           txa               Byte zurückholen
9670   29  01       and   #$01        und Bit 0 isolieren
9672   85  54       sta   $54         Bit merken
9674   2c  0f  18   bit   $180f       warten, bis
9677   30  fb       bmi   $9674       Byte eingelesen
9679   ad  01  1c   lda   $1c01       Byte vom Diskcontroller holen
967c   85  55       sta   $55         und merken
967e   29  f0       and   #$f0        obere 4 Bits isolieren,
9680   05  54       ora   $54         3. GCR-Nibble bilden
9682   aa           tax               und
9683   bd  0f  9f   lda   $9f0f,x     Binäräquivalent holen
9686   05  52       ora   $52         mit vorherigem Wert verknüpfen
9688   91  30       sta   ($30),y     vollständiges Byte in Puffer schreiben
968a   c8           iny               Zeiger auf nächstes Byte setzen
968b   a5  55       lda   $55         Byte holen,
968d   29  0f       and   #$0f        untere 4 Bits isolieren
968f   85  55       sta   $55         und merken
9691   2c  0f  18   bit   $180f       warten, bis
9694   30  fb       bmi   $9691       Byte eingelesen
9696   ad  01  1c   lda   $1c01       Byte vom Diskcontroller holen
9699   85  3a       sta   $3a         merken
969b   29  80       and   #$80        oberes Bit isolieren
969d   05  55       ora   $55         und 4. GCR-Nibble bilden
969f   aa           tax               Wert merken
96a0   bd  1d  9f   lda   $9f1d,x     und Binäräquivalent holen
96a3   85  52       sta   $52         und merken
96a5   a5  3a       lda   $3a         GCR-Rest holen
96a7   aa           tax               und merken
96a8   bd  0d  a2   lda   $a20d,x     Binäräquivalent holen
96ab   05  52       ora   $52         und mit vorigem Wert verknüpfen
96ad   91  30       sta   ($30),y     vollständiges Byte in Puffer schreiben
96af   c8           iny               Zeiger auf nächstes Byte
96b0   8a           txa               Wert zurückholen
96b1   29  03       and   #$03        und untere 2 Bits isolieren
96b3   85  3a       sta   $3a         Wert merken
96b5   2c  0f  18   bit   $180f       warten, bis
96b8   30  fb       bmi   $96b5       Byte eingelesen
96ba   ad  01  1c   lda   $1c01       Byte vom Diskcontroller holen
96bd   85  53       sta   $53         und merken
96bf   29  e0       and   #$e0        obere 3 Bits isolieren
96c1   05  3a       ora   $3a         und mit vorigem Wert zu GCR-Nibble summieren
96c3   aa           tax               Wert merken
96c4   bd  2a  9f   lda   $9f2a,x     Binäräquivalent holen
```

```
96c7   85  52       sta   $52         und merken
96c9   a5  53       lda   $53         Wert zurückholen
96cb   aa           tax               und merken
96cc   bd  0d  a3   lda   $a30d,x     Binäräquivalent holen
96cf   05  52       ora   $52         und mit vorigem Teil verknüpfen
96d1   91  30       sta   ($30),y     Byte in Puffer schreiben
96d3   c8           iny               Zeiger auf nächstes Byte
96d4   4c  3b  96   jmp   $963b       und weitermachen
96d7   a5  54       lda   $54         letztes GCR-Byte holen
96d9   aa           tax               und merken
96da   bd  0d  a1   lda   $a10d,x     Binäräquivalent holen
96dd   85  52       sta   $52         und merken
96df   8a           txa               Wert zurückholen
96e0   29  01       and   #$01        und Bit 0 isolieren
96e2   85  54       sta   $54         Bit merken
96e4   2c  0f  18   bit   $180f       warten, bis
96e7   30  fb       bmi   $96e4       Byte eingelesen
96e9   ad  01  1c   lda   $1c01       Byte vom Diskcontroller holen,
96ec   29  f0       and   #$f0        obere 4 Bits isolieren
96ee   05  54       ora   $54         und GCR-Nibble bilden
96f0   aa           tax               Wert merken
96f1   bd  0f  9f   lda   $9f0f,x     Binäräquivalent holen,
96f4   05  52       ora   $52         mit vorigem Byte verknüpfen
96f6   85  53       sta   $53         und Wert merken
96f8   68           pla               Datenblockkennzeichen zurückholen
96f9   c5  47       cmp   $47         mit richtigem Wert (normal 7) vergleichen
96fb   d0  0a       bne   $9707       verzweige, wenn falsch
96fd   20  e9  f5   jsr   $f5e9       sonst Prüfsumme über Datenblock berechnen
9700   c5  53       cmp   $53         und mit abgespeichertem Wert vergleichen
9702   f0  06       beq   $970a       verzweige, wenn richtig
9704   a9  05       lda   #$05        sonst Nummer für '23, READ ERROR'
9706   2c           .byte $2c         nächsten Befehl überspringen
9707   a9  04       lda   #$04        Nummer für '22, READ ERROR'
9709   2c           .byte $2c         nächsten Befehl überspringen
970a   a9  01       lda   #$01        Nummer für 'OK'
970c   4c  b5  99   jmp   $99b5       (Fehler)Meldung ausgeben; Ende
-------------------------------------------------------------------------
970f                                  Blockheader mit gegebenen Parametern suchen. Diese Routine
                                      steht analog bei $f510 für 1541.
970f   a5  12       lda   $12         ID 1 holen
9711   85  16       sta   $16         und für Suche setzen
9713   a5  13       lda   $13         ID 2 holen
9715   85  17       sta   $17         und für Suche setzen
9717   a0  00       ldy   #$00        Zeiger in Jobspeicher setzen
9719   b1  32       lda   ($32),y     Tracknummer des Jobs holen
971b   85  18       sta   $18         und übernehmen
971d   c8           iny               Zeiger erhöhen
971e   b1  32       lda   ($32),y     Sektornummer des Jobs holen
9720   85  19       sta   $19         und übernehmen
9722   a9  00       lda   #$00        Startwert für Prüfsumme
9724   45  16       eor   $16         mit ID 1 verknüpfen
9726   45  17       eor   $17         mit ID 2 verknüpfen
9728   45  18       eor   $18         mit Tracknummer verknüpfen
972a   45  19       eor   $19         mit Sektornummer verknüpfen
972c   85  1a       sta   $1a         und Prüfsumme merken
972e   20  34  f9   jsr   $f934       Blockheader in GCR umrechnen
9731   a9  5a       lda   #$5a        90 Leseversuche maximal
9733   85  4b       sta   $4b         setzen
```

```
9735   20  54  97    jsr   $9754       SYNC-Signal abwarten
9738   b9  24  00    lda   $0024,y     Byte des Blockheaders
973b   2c  0f  18    bit   $180f       warten, bis
973e   30  fb        bmi   $973b       Byte von Diskette eingelesen
9740   cd  01  1c    cmp   $1c01       Byte vom Diskcontroller vergleichen
9743   d0  06        bne   $974b       verzweige, wenn ungeicr
9745   c8            iny               sonst Zeiger erhöhen
9746   c0  08        cpy   #$08        und mit 8 (letztes Headerbyte) vergleichen
9748   d0  ee        bne   $9738       weitermachen, wenn noch nicht fertig
974a   60            rts               sonst Ende; alles ok
974b   c6  4b        dec   $4b         Zähler für Leseversuche minus 1
974d   d0  e6        bne   $9735       und weitermachen, wenn ungleich Null
974f   a9  02        lda   #$02        Nummmer der Fehlermeldung
9751   4c  b5  99    jmp   $99b5       '20, READ ERROR' ausgeben
-------------------------------------------------------------------------
9754                                   Wartet ein SYNC-Signal auf Diskette ab Diese Routine stehht
                                       analog bei $f556.
9754   a2  0f        ldx   #$0f        Zähler für Leseversuche Hi
9756   a0  00        ldy   #$00        Zähler für Leseversucne Lo (3840 Versuche)
9758   2c  00  1c    bit   $1c00       SYNC-Signal aufgetreten?
975b   10  0b        bpl   $9768       verzweige, wenn ja
975d   88            dey               sonst Zähler Lo vermindern
975e   d0  f8        bne   $9758       und weitersuchen
9760   ca            dex               Zähler Hi vermindern
9761   d0  f5        bne   $9758       und weitersuchen
9763   a9  03        lda   #$03        Nummer der Fehlermeldung
9765   4c  b5  99    jmp   $99b5       '21, READ ERROR' ausgeben
9768   ad  01  1c    lda   $1c01       Byte lesen; Diskcontroller freimachen
976b   a0  00        ldy   #$00        Index löschen
976d   60            rts               Ende
-------------------------------------------------------------------------
976e                                   Prüft auf Jobcode für Schreiben und schreibt ggf. den
                                       aktuellen Block als GCR-Sektor auf die Diskette. Analog zu
                                       $f56e bei der 1541.
976e   c9  10        cmp   #$10        Jobcode für Schreiben?
9770   f0  03        beq   $9775       verzweige, wenn ja
9772   4c  98  98    jmp   $9898       sonst Jobcode weiter prüfen
9775   20  e9  f5    jsr   $f5e9       Prüfsumme über Datenblock berechnen
9778   85  3a        sta   $3a         und abspeichern
977a   ad  00  1c    lda   $1c00       Steuerport des Diskcontrollers lesen und
977d   29  10        and   #$10        auf Schreibschutz testen
977f   d0  05        bne   $9786       verzweige, wenn kein Schreibschutz
9781   a9  08        lda   #$08        sonst Nummer der Fehlermeldung
9783   4c  b5  99    jmp   $99b5       '26, WRITE PROTECT' ausgeben; Ende
9786   20  8f  f7    jsr   $f78f       Pufferinhalt in GCR umwandeln
9789   20  0f  97    jsr   $970f       Blockheader suchen
978c   a0  09        ldy   #$09        Anzahl der Bytes,
978e   2c  0f  18    bit   $180f       die überlesen
9791   30  fb        bmi   $978e       werden müssen, um
9793   2c  00  1c    bit   $1c00       hinter den
9796   88            dey               Blockheader zu
9797   d0  f5        bne   $978e       kommen
9799   a9  ff        lda   #$ff        Schreib-/Lesekopf auf Ausgang
979b   8d  03  1c    sta   $1c03       schalten
979e   ad  0c  1c    lda   $1c0c       PCR laden und
97a1   29  1f        and   #$1f        Elektronik auf
97a3   09  c0        ora   #$c0        Schreiben
97a5   8d  0c  1c    sta   $1c0c       umschalten
```

```
97a8   a9  ff       lda   #$ff        Bytewert für SYNC
97aa   a0  05       ldy   #$05        Byte muß fünfmal geschrieben werden
97ac   8d  01  1c   sta   $1c01       Byte an Diskcontroller
97af   2c  0f  18   bit   $180f       und
97b2   30  fb       bmi   $97af       Schreibvorgang abwarten
97b4   2c  00  1c   bit   $1c00       Flag für BYTE READY löschen
97b7   88           dey               Anzahl der Bytes vermindern
97b8   d0  f5       bne   $97af       und weitermachen
97ba   a0  bb       ldy   #$bb        Zeiger auf Ausweichpuffer
97bc   b9  00  01   lda   $0100,y     Byte holen
97bf   2c  0f  18   bit   $180f       warten, bis Diskcontroller
97c2   30  fb       bmi   $97bf       bereit für Schreiben
97c4   8d  01  1c   sta   $1c01       Byte an Diskcontroller übergeben
97c7   c8           iny               Zeiger auf nächstes Byte
97c8   d0  f2       bne   $97bc       und weitermachen
97ca   b1  30       lda   ($30),y     Byte aus Puffer holen
97cc   2c  0f  18   bit   $180f       warten, bis Diskcontroller bereit
97cf   30  fb       bmi   $97cc       zum Schreiben
97d1   8d  01  1c   sta   $1c01       Byte an Diskcontroller übergeben
97d4   c8           iny               Zeiger auf nächstes Byte
97d5   d0  f3       bne   $97ca       und weitermachen
97d7   2c  0f  18   bit   $180f       warten, bis
97da   30  fb       bmi   $97d7       Diskcontroller bereit
97dc   ad  0c  1c   lda   $1c0c       und PCR lesen,
97df   09  e0       ora   #$e0        um die Elektronik auf Lesebetrieb
97e1   8d  0c  1c   sta   $1c0c       umzuschalten
97e4   a9  00       lda   #$00        Schreib-/Lesekopf auf Eingang
97e6   8d  03  1c   sta   $1c03       schalten
97e9   20  f9  97   jsr   $97f9       Puffer von GCR in Binär umwandeln
97ec   a4  3f       ldy   $3f         Puffernummer
97ee   b9  00  00   lda   $0000,y     Jobcode für Puffer
97f1   49  30       eor   #$30        in Jobcode für Verify umwandeln
97f3   99  00  00   sta   $0000,y     und wieder übergeben
97f6   4c  4f  94   jmp   $944f       Verify ausführen; Ende
-------------------------------------------------------------------------
97f9                                  Routine wandelt den Inhalt des Ausweichpuffers und des
                                      aktuellen Puffers vom GCR-(5 Bit)-Format wieder in das
                                      Binär-(4 Bit)-Format um. Analog zu $f5f2 bei der 1541.
97f9   a9  00       lda   #$00        Pufferadresse Lo
97fb   85  2e       sta   $2e         setzen
97fd   85  30       sta   $30         setzen
97ff   85  4f       sta   $4f         und Pufferzeiger setzen
9801   a5  31       lda   $31         Pufferadresse Hi
9803   85  4e       sta   $4e         setzen
9805   a9  01       lda   #$01        Pufferadresse Hi für Ausweichpuffer
9807   85  31       sta   $31         setzen
9809   85  2f       sta   $2f         setzen
980b   a9  bb       lda   #$bb        Pufferadresse Lo für Ausweichpuffer
980d   85  34       sta   $34         als Pufferzeiger setzen
980f   85  36       sta   $36         setzen
9811   20  d9  98   jsr   $98d9       5 GCR-Bytes in 4 Binärbytes konvertieren
9814   a5  52       lda   $52         erstes Binärbyte
9816   85  38       sta   $38         ist Datenblockkennzeichen
9818   a4  36       ldy   $36         Pufferzeiger holen
981a   a5  53       lda   $53         zweites Binärbyte
981c   91  2e       sta   ($2e),y     in Puffer schreiben
981e   c8           iny               Zeiger auf nächstes Byte
```

```
981f   a5  54       lda   $54         drittes Binärbyte
9821   91  2e       sta   ($2e),y     in Puffer schreiben
9823   c8           iny               Zeiger auf nächstes Byte
9824   a5  55       lda   $55         viertes Binärbyte
9826   91  2e       sta   ($2e),y     in Puffer schreiben
9828   c8           iny               Zeiger auf nächstes S;.te
9829   84  36       sty   $36         merken
982b   20  d9  98   jsr   $98d9       5 GCR-Bytes in 4 Binärbytes konvertieren
982e   a4  36       ldy   $36         Pufferzeiger holen
9830   a5  52       lda   $52         erstes Binärbyte
9832   91  2e       sta   ($2e),y     in Puffer schreiben
9834   c8           iny               Zeiger auf nächstes Byte
9835   a5  53       lda   $53         zweites Binärbyte
9837   91  2e       sta   ($2e),y     in Puffer schreiben
9839   c8           iny               Zeiger auf nächstes Byte
983a   f0  0e       beq   $984a       verzweige, wenn Ende erreicht
983c   a5  54       lda   $54         drittes Binärbyte
983e   91  2e       sta   ($2e),y     in Puffer schreiben
9840   c8           iny               Zeiger auf nächstes Byte
9841   a5  55       lda   $55         viertes Binärbyte
9843   91  2e       sta   ($2e),y     in Puffer schreiben
9845   c8           iny               Zeiger auf nächstes Byte
9846   84  36       sty   $36         merken
9848   d0  e1       bne   $982b       und weitermachen, bis Ende
984a   a5  54       lda   $54         drittes Binärbyte
984c   91  30       sta   ($30),y     in aktuellen Puffer schreiben
984e   c8           iny               Zeiger auf nächstes Byte
984f   a5  55       lda   $55         viertes Byte
9851   91  30       sta   ($30),y     in aktuellen Puffer schreiben
9853   c8           iny               Zeiger auf nächstes Byte
9854   84  36       sty   $36         merken
9856   20  d9  98   jsr   $98d9       5 GCR-Bytes in 4 Binärbytes konvertieren
9859   a4  36       ldy   $36         Pufferzeiger holen
985b   a5  52       lda   $52         erstes Binärbyte holen
985d   91  30       sta   ($30),y     und in aktuellen Puffer schreiben
985f   c8           iny               Zeiger auf nächstes Zeichen
9860   a5  53       lda   $53         zweites Binärbyte
9862   91  30       sta   ($30),y     in aktuellen Puffer schreiben
9864   c8           iny               Zeiger auf nächstes Byte
9865   a5  54       lda   $54         drittes Binärbyte
9867   91  30       sta   ($30),y     in aktuellen Puffer schreiben
9869   c8           iny               Zeiger auf nächstes Byte
986a   a5  55       lda   $55         viertes Binärbyte
986c   91  30       sta   ($30),y     in aktuellen Puffer schreiben
986e   c8           iny               Zeiger auf nächstes Byte
986f   84  36       sty   $36         merken
9871   c0  bb       cpy   #$bb        schon Ende erreicht?
9873   90  e1       bcc   $9856       verzweige, wenn nein
9875   a9  45       lda   #$45        Pufferadresse Lo
9877   85  2e       sta   $2e         neu setzen; zweiter Teil
9879   a5  31       lda   $31         Pufferadresse Hi
987b   85  2f       sta   $2f         übernehmen
987d   a0  ba       ldy   #$ba        Zeiger in Puffer
987f   b1  30       lda   ($30),y     und Pufferinhalt
9881   91  2e       sta   ($2e),y     nach hinten verschieben
9883   88           dey               weitermachen
9884   d0  f9       bne   $987f       bis alles verschoben
9886   b1  30       lda   ($30),y     letztes Byte
```

```
9888   91  2e       sta   ($2e),y     ebenfalls verschieben
988a   a2  bb       ldx   #$bb        Zeiger in Ausweichpuffer
988c   bd  00  01    lda   $0100,x     Zeichen aus Ausweichpuffer holen
988f   91  30       sta   ($30),y     und in aktuellen Puffer verschieben
9891   c8           iny               Zeiger auf nächstes Zeichen
9892   e8           inx               setzen
9893   d0  f7       bne   $988c       und weitermachen
9895   86  50       stx   $50         0; Flag für Puffer im Binärformat setzen
9897   60           rts               Ende
-------------------------------------------------------------------------
9898                                  Prüft auf Jobcode für Verify und vergleicht ggf. die Daten
                                      im Puffer mit denen auf der Diskette. Analog dazu auch
                                      $f691 für 1541.
9898   c9  20       cmp   #$20        Jobcode für Verify?
989a   f0  02       beq   $989e       verzweige, wenn ja
989c   d0  30       bne   $98ce       unbedingter Sprung
989e   20  e9  f5    jsr   $f5e9       Prüfsumme über Datenblock berechnen
98a1   85  3a       sta   $3a         und merken
98a3   20  8f  f7    jsr   $f78f       Datenblock in GCR-Format umwandeln
98a6   20  00  96    jsr   $9600       Blockheader suchen
98a9   a0  bb       ldy   #$bb        Zeiger in Ausweichpuffer
98ab   b9  00  01    lda   $0100,y     Zeichen holen
98ae   2c  0f  18    bit   $180f       Lesevorgang des Diskcontrollers
98b1   30  fb       bmi   $98ae       abwarten
98b3   4d  01  1c    eor   $1c01       und Byte vergleichen
98b6   d0  1c       bne   $98d4       verzweige, wenn ungleich
98b8   c8           iny               Zeiger auf nächstes Byte
98b9   d0  f0       bne   $98ab       und weitermachen
98bb   b1  30       lda   ($30),y     Byte aus aktuellem Puffer
98bd   2c  0f  18    bit   $180f       Lesevorgang des Diskcontrollers
98c0   30  fb       bmi   $98bd       abwarten
98c2   4d  01  1c    eor   $1c01       und Byte vergleichen
98c5   d0  0d       bne   $98d4       verzweige, wenn ungleich
98c7   c8           iny               Zeiger auf nächstes Byte
98c8   c0  fd       cpy   #$fd        Ende erreicht?
98ca   d0  ef       bne   $98bb       verzweige, wenn nein
98cc   f0  03       beq   $98d1       unbedingter Sprung
98ce   20  0f  97    jsr   $970f       nächsten Blockheader suchen
98d1   a9  01       lda   #$01        Nummer für 'OK'
98d3   2c           .byte $2c         nächsten Befehl überspringen
98d4   a9  07       lda   #$07        Nummer für '25, WRITE ERROR'
98d6   4c  b5  99    jmp   $99b5       (Fehler)Meldung ausgeben; Ende
-------------------------------------------------------------------------
98d9                                  Wandelt 5 GCR-(5 Bit)-codierte Werte aus dem Puffer in 4
                                      Binär-(4 Bit)-Bytes um und legt diese dann nach $56 bis $59
                                      ab.
98d9   a4  34       ldy   $34         Pufferzeiger
98db   b1  30       lda   ($30),y     GCR-Byte aus Puffer holen
98dd   85  56       sta   $56         und merken
98df   29  07       and   #$07        Bits 0 bis 2 isolieren
98e1   85  57       sta   $57         und merken
98e3   c8           iny               Zeiger auf nächstes Byte
98e4   d0  06       bne   $98ec       verzweige, wenn ungleich Null
98e6   a5  4e       lda   $4e         Pufferadresse Hi des aktuellen Puffers
98e8   85  31       sta   $31         setzen
98ea   a4  4f       ldy   $4f         Pufferadresse Lo als Zeiger
98ec   b1  30       lda   ($30),y     GCR-Byte aus Puffer holen
98ee   85  58       sta   $58         und merken
```

```
98f0   29  c0       and   #$c0        Bits 6 und 7 isolieren
98f2   05  57       ora   $57         und mit Bits 0 bis 2 des ersten ertes verbinden
98f4   85  57       sta   $57         1. GCR-Byte merken
98f6   a5  58       lda   $58         GCR-Byte holen
98f8   29  01       and   #$01        und Bit 0 isolieren
98fa   85  59       sta   $59         Wert merken
98fc   c8           iny               Zeiger auf nächstes GCR-alte
98fd   b1  30       lda   ($30),y     GCR-Byte aus Puffer hc e
98ff   aa           tax               und merken
9900   29  f0       and   #$f0        Bits 4 bis 7 isolieren
9902   05  59       ora   $59         und mit vorigem Bit 0 verknüpfen
9904   85  59       sta   $59         2. GCR-Byte merken
9906   8a           txa               Wert zurückholen
9907   29  0f       and   #$0f        und Bits 0 bis 3 isolieren
9909   85  5a       sta   $5a         und merken
990b   c8           iny               Zeiger auf nächstes Byte
990c   b1  30       lda   ($30),y     GCR-Byte aus Puffer holen
990e   85  5b       sta   $5b         und merken
9910   29  80       and   #$80        Bit 7 isolieren,
9912   05  5a       ora   $5a         mit vorigen Bits 0 bis 3 verknpfen
9914   85  5a       sta   $5a         und 3. GCR-Byte merken
9916   a5  5b       lda   $5b         Wert zurückholen
9918   29  03       and   #$03        Bits 0 und 1 isolieren
991a   85  5c       sta   $5c         und merken
991c   c8           iny               Zeiger auf nächstes Byte
991d   d0  08       bne   $9927       verzweige, wenn ungleich 0
991f   a5  4e       lda   $4e         sonst aktuelle Pufferadresse Hi holen
9921   85  31       sta   $31         und übernehmen
9923   a4  4f       ldy   $4f         Pufferadresse Lo als Zeiger
9925   84  30       sty   $30         merken
9927   b1  30       lda   ($30),y     GCR-Byte aus Puffer holen
9929   85  5d       sta   $5d         und merken
992b   29  e0       and   #$e0        Bits 5 bis 7 isolieren
992d   05  5c       ora   $5c         und mit vorigen Bits 0 und 1 verknüpfen
992f   85  5c       sta   $5c         4. GCR-Byte merken
9931   c8           iny               Zeiger auf nächstes Byte
9932   84  34       sty   $34         merken
9934   a6  56       ldx   $56         1. GCR-Nibble
9936   bd  0d  a0   lda   $a00d,x     Binäräquivalent holen
9939   a6  57       ldx   $57         2. GCR-Nibble
993b   1d  0d  9f   ora   $9f0d,x     Binäräquivalent holen und
993e   85  52       sta   $52         erstes Byte merken
9940   a6  58       ldx   $58         4. GCR-Nibble
9942   bd  0d  a1   lda   $a10d,x     Binäräquivalent holen
9945   a6  59       ldx   $59         5. GCR-Nibble
9947   1d  0f  9f   ora   $9f0f,x     Binäräquivalent holen und
994a   85  53       sta   $53         zweites Byte merken
994c   a6  5a       ldx   $5a         7. GCR-Nibble
994e   bd  1d  9f   lda   $9f1d,x     Binäräquivalent holen
9951   a6  5b       ldx   $5b         8. GCR-Nibble
9953   1d  0d  a2   ora   $a20d,x     Binäräquivalent holen und
9956   85  54       sta   $54         drittes Byte merken
9958   a6  5c       ldx   $5c         9. GCR-Nibble
995a   bd  2a  9f   lda   $9f2a,x     Binäräquivalent holen
995d   a6  5d       ldx   $5d         10. GCR-Nibble
995f   1d  0d  a3   ora   $a30d,x     Binäräquivalent holen und
9962   85  55       sta   $55         viertes Byte merken
9964   60           rts               Ende
```

```
------------------------------------------------------------------------
9965                                    Ausweichpuffer $01bb bis $01ff von GCR nach Binär
                                        umwandeln. Bytes werden dann im aktuellen Puffer abgelegt.
                                        Analog zu $f8eO.
9965    a9  00      lda   #$00          Pufferzeiger auf 0
9967    85  34      sta   $34           setzen
9969    85  2e      sta   $2e           Pufferadresse Lo
996b    85  36      sta   $36           Pufferzeiger
996d    a9  01      lda   #$01          Pufferadresse Hi
996f    85  4e      sta   $4e           setzen
9971    a9  ba      lda   #$ba          Puffer adresse Lo
9973    85  4f      sta   $4f           setzen; ergibt $01ba
9975    a5  31      lda   $31           Pufferadresse Hi
9977    85  2f      sta   $2f           übernehmen
9979    20  d9  98  jsr   $98d9        5 GCR-Bytes in 4 Binärbytes konvertieren
997c    a5  52      lda   $52           erstes Byte
997e    85  38      sta   $38           als Kennzeichen des Datenblocks nehmen
9980    a4  36      ldy   $36           Pufferzeiger holen
9982    a5  53      lda   $53           zweites Byte
9984    91  2e      sta   ($2e),y       in Puffer schreiben
9986    c8          iny                 Zeiger auf nächstes Byte
9987    a5  54      lda   $54           drittes Byte holen
9989    91  2e      sta   ($2e),y       in Puffer schreiben
998b    c8          iny                 Zeiger auf nächstes Byte
998c    a5  55      lda   $55           viertes Byte holen
998e    91  2e      sta   ($2e),y       in Puffer schreiben
9990    c8          iny                 Zeiger auf nächstes Byte
9991    84  36      sty   $36           merken
9993    20  d9  98  jsr   $98d9        5 GCR-Bytes in 4 Binärbytes konvertieren
9996    a4  36      ldy   $36           Pufferzeiger holen
9998    a5  52      lda   $52           erstes Byte der Umwandlung
999a    91  2e      sta   ($2e),y       in Puffer schreiben
999c    c8          iny                 Zeiger auf nächstes Byte
999d    f0  11      beq   $99b0        verzweige, wenn Null
999f    a5  53      lda   $53           zweites Byte der Umwandlung
99a1    91  2e      sta   ($2e),y       in Puffer schreiben
99a3    c8          iny                 Zeiger auf nächstes Byte
99a4    a5  54      lda   $54           drittes Byte der Umwandlung
99a6    91  2e      sta   ($2e),y       in Puffer schreiben
99a8    c8          iny                 Zeiger auf nächstes Byte
99a9    a5  55      lda   $55           viertes Byte der Umwandlung
99ab    91  2e      sta   ($2e),y       in Puffer schreiben
99ad    c8          iny                 Zeiger auf nächstes Byte
99ae    d0  e1      bne   $9991        und weitermachen
99b0    a5  2f      lda   $2f           Pufferadresse Hi
99b2    85  31      sta   $31           wieder setzen
99b4    60          rts                 Ende
------------------------------------------------------------------------
99b5                                    Ausgang der 1571-Jobschleife mit Übergabe der Fehlernummer
                                        in A. Analog zu $f969.
99b5    a4  3f      ldy   $3f           Puffernummer
99b7    99  00  00  sta   $0000,y       Rückmeldung in Jobspeicher
99ba    a5  50      lda   $50           Flag für Puffer enthält GCR-Bytes
99bc    f0  03      beq   $99c1        verzweige, wenn nicht gesetzt; alles ok
99be    20  f9  97  jsr   $97f9        sonst Puffer nach Binär umwandeln
99c1    20  8f  f9  jsr   $f98f        Zähler für Ausschalten des Drivemotors setzen
99c4    a6  49      ldx   $49           Stackpointer zurückholen
```

```
99c6   9a           txs             und wieder setzen
99c7   4c  c8  92    jmp   $92c8     zurück zur Jobabfrage;
-------------------------------------------------------------------------
99ca                                 Kontrollroutine für Diskcontroller des DOS. Hier erfolgt
                                     die Steuerung des Steppermotors und des Drivemotors. Analog
                                     zu $f99c der 1541.
99ca   ad  07  1c    lda   $1c07     Timer neu setzen; IRQ-Status
99cd   8d  05  1c    sta   $1c05     löschen
99d0   ad  00  1c    lda   $1c00     auf Diskettenwechsel
99d3   29  10        and   #$10      testen
99d5   c5  1e        cmp   $1e       wurde die Diskette gewechselt?
99d7   85  1e        sta   $1e       Status merken
99d9   d0  07        bne   $99e2     verzweige, wenn ja
99db   ad  ab  02    lda   $02ab     Zählbyte für Motorverzögerung
99de   d0  10        bne   $99f0     verzweige, wenn ungleich Null
99e0   f0  1c        beq   $99fe     unbedingter Sprung; Motor ist aus
99e2   a9  ff        lda   #$ff      Zähler für automatisches Anlaufen des Motors
99e4   8d  ab  02    sta   $02ab     bei Diskettenwechsel setzen
99e7   20  64  87    jsr   $8764     Drivemotor einschalten
99ea   a9  01        lda   #$01      Flag für Diskettenwechsel
99ec   85  1c        sta   $1c       setzen
99ee   d0  0e        bne   $99fe     unbedingter Sprung
99f0   ce  ab  02    dec   $02ab     Zähler für Motorverzögerung vermindern
99f3   d0  09        bne   $99fe     verzweige, wenn ungleich Null
99f5   a5  20        lda   $20       sonst Flags für Drivestatus holen
99f7   c9  00        cmp   #$00      Motor aus?
99f9   d0  03        bne   $99fe     verzweige, wenn nein
99fb   20  70  87    jsr   $8770     sonst Drivemotor ausschalten
99fe   ad  fe  02    lda   $02fe     Byte für Kopfdejustierung bei Leseproblemen
9a01   f0  15        beq   $9a18     verzweige, wenn Kopf genau auf Track steht
9a03   c9  02        cmp   #$02      wurde Kopf gerade wieder auf Track gestellt?
9a05   d0  07        bne   $9a0e     verzweige, wenn nein
9a07   a9  00        lda   #$00      Flag für Kopf steht wieder auf Track
9a09   8d  fe  02    sta   $02fe     setzen
9a0c   f0  0a        beq   $9a18     unbedingter Sprung
9a0e   85  4a        sta   $4a       Kopf steht auf einer Halbspur; also muß
9a10   a9  02        lda   #$02      der Kopf eine Halbspur bewegt werden, um wieder
9a12   8d  fe  02    sta   $02fe     richtig zu stehen
9a15   4c  56  9a    jmp   $9a56     Kopfbewegung ausführen
9a18   a6  3e        ldx   $3e       Drivenummer für Jobschleife
9a1a   30  07        bmi   $9a23     verzweige, wenn Drive inaktiv
9a1c   a5  20        lda   $20       sonst Flags für Drivestatus lesen
9a1e   a8            tay             und merken
9a1f   c9  20        cmp   #$20      Motor an und Drive bereit?
9a21   d0  03        bne   $9a26     verzweige, wenn nein
9a23   4c  c9  9a    jmp   $9ac9     sonst zurück aus IRQ-Programm
9a26   c6  48        dec   $48       Verzögerungszähler für Drivemotor
9a28   d0  1c        bne   $9a46     verzweige, wenn ungleich Null
9a2a   98            tya             Flags für Drivestatus holen
9a2b   10  04        bpl   $9a31     verzweige, wenn Drive bereit
9a2d   29  7f        and   #$7f      sonst Flag für Drive bereit
9a2f   85  20        sta   $20       setzen
9a31   29  10        and   #$10      und Flag für Motor in Ausschaltphase
9a33   f0  11        beq   $9a46     verzweige, wenn nicht gesetzt
9a35   c6  35        dec   $35       sonst Ausschaltverzögerung minus 1
9a37   d0  0d        bne   $9a46     und weiter, wenn ungleich Null
9a39   20  70  87    jsr   $8770     Motor ausschalten
9a3c   a9  ff        lda   #$ff      Flag für aktives Drive
```

```
9a3e   85  3e       sta   $3e        löschen
9a40   a9  00       lda   #$00       Drivestatus auf 'inaktiv'
9a42   85  20       sta   $20        setzen
9a44   f0  dd       beq   $9a23      unbedingter Sprung
9a46   98           tya               Flags für Drivestatus holen
9a47   29  40       and   #$40       Steppermotor in Aktion?
9a49   d0  03       bne   $9a4e      verzweige, wenn ja
9a4b   4c  c9  9a    jmp   $9ac9      $9ac9 sonst zurück aus IRQ-Programm
9a4e   a5  62       lda   $62        aktuelle Stepperroutine in Aktion?
9a50   d0  50       bne   $9aa2      verzweige, wenn ja
9a52   a5  4a       lda   $4a        Anzahl der Schritte für Steppermotor
9a54   f0  43       beq   $9a99      verzweige, wenn keine Kopfbewegung
-------------------------------------------------------------------------
9a56                                 Steuerung der Kopfbewegung.
9a56   a5  4a       lda   $4a        Anzahl der Schritte
9a58   10  59       bpl   $9ab3      verzweige, wenn Bewegung nach innen
9a5a   98           tya               Flags für Drivestatus
9a5b   48           pha               merken
9a5c   a0  63       ldy   #$63       Dauer einer Kopfbewegung in Schleifendurchläufen
9a5e   ad  0f  18    lda   $180f      Steuerport lesen
9a61   6a           ror               Zustand der lichtschranke bei Track 0 ins Carry
9a62   08           php               Zustand merken
9a63   ad  0f  18    lda   $180f      Steuerport lesen
9a66   6a           ror               Zustand der lichtschranke bei Track 0
9a67   6a           ror               ins Bit 7 des Akku schieben
9a68   28           plp               vorigen Zustand zurückholen
9a69   29  80       and   #$80       und auf Nullanschlag testen
9a6b   90  04       bcc   $9a71      verzweige, wenn Anschlag beim 1. Test
9a6d   10  1d       bpl   $9a8c      verzweige, wenn Anschlag beim 2. Test
9a6f   30  02       bmi   $9a73      unbedingter Sprung; kein Anschlag
9a71   30  19       bmi   $9a8c      verzweige, wenn kein Anschlag im 2. Test
9a73   88           dey               Zähler vermindern; Kopf noch in Bewegung?
9a74   d0  e8       bne   $9a5e      verzweige, wenn ja
9a76   b0  14       bcs   $9a8c      sonst verzweige, wenn kein Anschlag
9a78   a5  7b       lda   $7b        Kopf n:ben Track.positioniert?
9a7a   d0  10       bne   $9a8c      verzwelge, wenn Ja
9a7c   ad  00  1c    lda   $1c00      Steuerport des Diskcontrollers
9a7f   29  03       and   #$03       Steppermotor eingeschaltet?
9a81   d0  09       bne   $9a8c      verzweige, wenn ja
9a83   68           pla               sonst Flags für Drivestatus holen
9a84   a8           tay               und merken
9a85   a9  00       lda   #$00       Anzahl der Schritte
9a87   85  4a       sta   $4a        zurücksetzen; keine Bewegung mehr
9a89   4c  c9  9a    jmp   $9ac9      zurück aus IRQ-Programm; Ende
9a8c   68           pla               Flags für Drivestatus zurückholen
9a8d   a8           tay               und merken
9a8e   e6  4a       inc   $4a        Schrittzähler vermindern (Invers, da negativ)
9a90   ad  00  1c    lda   $1c00      Steuerport lesen
9a93   38           sec               und
9a94   e9  01       sbc   #$01       neue Ansteuerung für Steppermotorspulen
9a96   4c  bb  9a    jmp   $9abb      setzen
9a99   a9  02       lda   #$02       Verzögerungszähler
9a9b   85  48       sta   $48        für weitere Bewegung setzen
9a9d   85  62       sta   $62        Flag für langsamen Steppermodus setzen
9a9f   4c  c9  9a    jmp   $9ac9      zurück aus IRQ-Programm; Ende
9aa2   c6  48       dec   $48        Zähler für Bremsen des Steppermotors
9aa4   d0  23       bne   $9ac9      verzweige, wenn ungleich Null
9aa6   a5  20       lda   $20        Flags für Drivestatus
```

```
9aa8   29  bf       and   #$bf       Steppermotor auf inaktiv
9aaa   85  20       sta   $20        setzen
9aac   a9  00       lda   #$00       Flag für Stepper
9aae   85  62       sta   $62        inaktiv setzen
9ab0   4c  c9  9a   jmp   $9ac9      zurück aus IRQ-Programm; Ende
-------------------------------------------------------------------------
9ab3                                 Kopf eine Halbspur nach innen bewegen.
9ab3   c6  4a       dec   $4a        Schrittzähler minus 1
9ab5   ad  00  1c   lda   $1c00      Steuerport des Diskcontrollers lesen
9ab8   18           clc              und
9ab9   69  01       adc   #$01       Stepperspulen für Bewegung neu setzen
-------------------------------------------------------------------------
9abb                                 Steuerbits für die Steppermotorspulen je nach Kopfbewegung
                                     setzen. Die Bits 0 und 1 bestim- men dabei die
                                     Bewegungsrichtung. Nehmen sie die Werte 00/01/10/11/... an,
                                     so bewegt sich der Kopf nach innen; bei der Folge
                                     11/10/01/00/... bewegt sich der Kopf nach außen.
9abb   29  03       and   #$03       Steuerbits isolieren
9abd   85  4b       sta   $4b        und merken
9abf   ad  00  1c   lda   $1c00      Steuerport lesen
9ac2   29  fc       and   #$fc       und Steuerbits isolieren
9ac4   05  4b       ora   $4b        neuen Bitstatus setzen
9ac6   8d  00  1c   sta   $1c00      und Steppermotor ansteuern
9ac9   60           rts              Ende
-------------------------------------------------------------------------
9aca                                 Steuerbytes für das Ausmessen einer Spur beim Formatieren.
                                     Jedem Zählwert für die Spurgröße ist dabei ein Wert für die
                                     Anzahl der Leerby- tes zwischen den Sektoren zugeordnet:
9aca 21 =  8448 Bytes    9ad3 02 = 02 Leerbytes
9acb 22 =  8704 Bytes    9ad4 02 = 02 Leerbytes
9acc 23 =  8960 Bytes    9ad5 04 = 04 Leerbytes
9acd 24 =  9216 Bytes    9ad6 06 = 06 Leerbytes
9ace 25 =  9472 Bytes    9ad7 08 = 08 Leerbytes
9acf 26 =  9728 Bytes    9ad8 08 = 08 Leerbytes
9ad0 27 =  9984 Bytes    9ad9 0b = 11 Leerbytes
9ad1 28 = 10240 Bytes    9ada 13 = 19 Leerbytes
9ad2 29 = 10496 Bytes    9adb 16 = 22 Leerbytes
-------------------------------------------------------------------------
9adc                                 Track für das Formatieren ausmessen, damit die Sektoren
                                     möglichst gleichmäßig verteilt werden. Hierbei wird die
                                     Größe einer Spur festgestellt.
9adc   a0  00       ldy   #$00       Zähler Lo
9ade   a2  1c       ldx   #$1c       Zähler Hi; ergibt 7168
9ae0   20  63  9d   jsr   $9d63      Track mit 7168 mal $55 vollschreiben
9ae3   20  73  9b   jsr   $9b73      Track mit 5120 mal $ff vollschreiben
9ae6   20  00  fe   jsr   $fe00      auf Lesen umschalten
9ae9   a0  ff       ldy   #$ff       Zähler Hi setzen
9aeb   a2  ff       ldx   #$ff       Zähler Lo setzen
9aed   2c  00  1c   bit   $1c00      auf SYNC testen
9af0   10  0b       bpl   $9afd      verzweige, wenn SYNC-Signal gefunden
9af2   ca           dex              sonst Zähler vermindern
9af3   d0  f8       bne   $9aed      und weitermachen
9af5   88           dey              Zähler Hi vermindern
9af6   d0  f3       bne   $9aeb      und weitermachen
9af8   a9  02       lda   #$02       Nummer der Fehlermeldung
9afa   4c  59  9d   jmp   $9d59      '20, READ ERROR' ausgeben; Ende
```

```
9afd   a0  00       ldy   #$00        Zähler Lo setzen
9aff   a2  00       ldx   #$00        Zähler Hi setzen
9b01   2c  00  1c   bit   $1c00       auf SYNC-Signal prüfen
9b04   10  fb       bpl   $9b01       warten, bis SYNC-Bereich zu Ende
9b06   ad  00  1c   lda   $1c00       Steuerport lesen
9b09   c8           iny               Zähler Lo erhöhen
9b0a   d0  01       bne   $9b0d       verzweige, wenn ungleich 0
9b0c   e8           inx               Zähler Hi erhöhen
9b0d   29  80       and   #$80        und auf SYNC-Signal prüfen
9b0f   d0  f5       bne   $9b06       und auf SYNC-Signal warten
9b11   a9  00       lda   #$00        sonst Zwischenspeicher
9b13   85  3b       sta   $3b         löschen
9b15   8a           txa               Zähler gibt Größe des Feldes ohne SYNC an
9b16   a2  08       ldx   #$08        Index auf größten Bytewert
9b18   dd  ca  9a   cmp   $9aca,x     und Trackgröße feststellen
9b1b   f0  05       beq   $9b22       verzweige, wenn richtigen Tabellenwert gefunden
9b1d   ca           dex               sonst Index auf nächsten Wert
9b1e   10  f8       bpl   $9b18       und weitersuchen
9b20   30  d6       bmi   $9af8       unbedingter Sprung; Fehler
9b22   bd  d3  9a   lda   $9ad3,x     zugehörige Größe der Sektor lücke holen
9b25   8d  26  06   sta   $0626       und abspeichern
9b28   60           rts               Ende
--------------------------------------------------------------------------
9b29                                  GCR-Formatierung einer Diskette. Jobroutine.
9b29   a5  51       lda   $51         Formatierung schon im Gange?
9b2b   10  2b       bpl   $9b58       verzweige, wenn ja
9b2d   a9  60       lda   #$60        sonst Drivestatus auf Beginn eines Zugriffs
9b2f   85  20       sta   $20         stellen
9b31   ad  b2  01   lda   $01b2       Flag für Diskettenseite abfragen
9b34   d0  03       bne   $9b39       verzweige, wenn Seite 1 formatiert werden soll
9b36   a9  01       lda   #$01        sonst Anfangstrack 1 für Seite 0 setzen
9b38   2c           .byte $2c         nächsten Befehl überspringen
9b39   a9  24       lda   #$24        Anfangstrack 36 für Seite 1 setzen
9b3b   85  22       sta   $22         Tracknummer übernehmen
9b3d   85  51       sta   $51         und für Formatierung setzen
9b3f   c9  24       cmp   #$24        mit 36 vergleichen
9b41   20  f3  93   jsr   $93f3       Hardware entsprechend Seite 0 oder 1 setzen
9b44   a9  a4       lda   #$a4        Anzahl der Schritte für Nullanschlag setzen,
9b46   85  4a       sta   $4a         damit Kopf auf Startposition gebracht wird
9b48   ad  00  1c   lda   $1c00       Steuerport lesen
9b4b   29  fc       and   #$fc        Bits für Steppermotor löschen
9b4d   8d  00  1c   sta   $1c00       und an Diskcontroller übergeben
9b50   a9  0a       lda   #$0a        10 Versuche für Formatieren, bevor mit
9b52   8d  20  06   sta   $0620       Fehlermeldung abgebrochen wird, setzen
9b55   4c  ca  99   jmp   $99ca       zur Jobschleife, um Diskcontroller zu starten
9b58   a0  00       ldy   #$00        Zeiger in Jobspeicher
9b5a   b1  32       lda   ($32),y     Tracknummer holen
9b5c   c5  51       cmp   $51         mit Track für Formatierung vergleichen
9b5e   f0  07       beq   $9b67       verzweige, wenn gleich
9b60   a5  51       lda   $51         sonst Track für Formatierung
9b62   91  32       sta   ($32),y     übernehmen
9b64   4c  ca  99   jmp   $99ca       und Kopf positionieren
9b67   ad  00  1c   lda   $1c00       Steuerport lesen
9b6a   29  10       and   #$10        Schreibschutz aktiviert?
9b6c   d0  1b       bne   $9b89       verzweige, wenn nein
9b6e   a9  08       lda   #$08        Nummer der Fehlermeldung
9b70   4c  51  9d   jmp   $9d51       '26, WRITE PROTECT ON'
```

```
------------------------------------------------------------------------
9b73                                    Einen Track mit $ff-Bytes beschreiben.
9b73   a2  14       ldx   #$14         Zähler Hi auf 20 setzen
9b75   a9  ff       lda   #$ff         Bytewert für Schreiben
9b77   2c  0f  18   bit   $180f        warten, bis Diskcontroller
9b7a   30  fb       bmi   $9b77        bereit ist
9b7c   8d  01  1c   sta   $1c01        Byte auf Diskette schreiben
9b7f   2c  00  1c   bit   $1c00        und Rückmeldung für Ende des Schreibens lesen
9b82   88           dey                Zähler Lo vermindern
9b83   d0  f2       bne   $9b77        und weitermachen
9b85   ca           dex                Zähler Hi vermindern
9b86   d0  ef       bne   $9b77        und weitermachen
9b88   60           rts                Ende
------------------------------------------------------------------------
9b89                                    Fortsetzung der Formatierroutine...
9b89   a5  3b       lda   $3b          sollen Sektoren gleichmäßig auf der Spur ver-
9b8b   10  03       bpl   $9b90        teilt werden ('partial format')?
9b8d   20  dc  9a   jsr   $9adc        ja, Track ausmessen; Kapazität feststellen
9b90   ad  26  06   lda   $0626        Ergebnis der Ausmessung (unsinniger Befehl)
9b93   18           clc                (unsinniger Befehl)
9b94   a9  03       lda   #$03         Pufferadresse Hi
9b96   85  33       sta   $33          setzen
9b98   a9  00       lda   #$00         Pufferadresse Lo
9b9a   85  32       sta   $32          setzen
9b9c   8d  28  06   sta   $0628        Nummer des ersten Sektors setzen
9b9f   a0  00       ldy   #$00         Index in Puffer setzen
9ba1   a5  39       lda   $39          Kennzeichen für Blockheader
9ba3   91  32       sta   ($32),y      in Puffer schreiben
9ba5   c8           iny                Zeiger auf nächstes Zeichen
9ba6   a9  00       lda   #$00         Platzhalter für Prüfsumme (kommt später)
9ba8   91  32       sta   ($32),y      in Puffer schreiben
9baa   c8           iny                Zeiger auf nächstes Zeichen
9bab   ad  28  06   lda   $0628        Sektornummer
9bae   91  32       sta   ($32),y      in Puffer schreiben
9bb0   c8           iny                Zeiger auf nächstes Byte
9bb1   a5  51       lda   $51          Tracknummer
9bb3   91  32       sta   ($32),y      in Puffer schreiben
9bb5   c8           iny                Zeiger auf nächstes Byte
9bb6   a5  13       lda   $13          ID 2
9bb8   91  32       sta   ($32),y      in Puffer schreiben
9bba   c8           iny                Zeiger auf nächstes Byte
9bbb   a5  12       lda   $12          ID 1
9bbd   91  32       sta   ($32),y      in Puffer schreiben
9bbf   c8           iny                Zeiger auf nächstes Byte
9bc0   a9  0f       lda   #$0f         Füllbyte
9bc2   91  32       sta   ($32),y      in Puffer schreiben
9bc4   c8           iny                Zeiger auf nächstes Byte
9bc5   91  32       sta   ($32),y      Füllbyte nochmal in Puffer schreiben
9bc7   c8           iny                Zeiger auf nächstes Byte
9bc8   98           tya                und
9bc9   48           pha                merken
9bca   a2  07       ldx   #$07         Zähler für Anzahl der Bytes im Blockheader
9bcc   a9  00       lda   #$00         Startwert für Prüfsumme
9bce   85  3a       sta   $3a          setzen
9bd0   88           dey                . Zeiger auf nächstes Byte im Blockheader
9bd1   b1  32       lda   ($32),y      Byte aus Blockheader holen,
9bd3   45  3a       eor   $3a          Prüfsumme bilden
9bd5   85  3a       sta   $3a          und merken
```

```
9bd7   ca           dex             Zähler vermindern
9bd8   d0  f6       bne   $9bd0     und weitermachen, bis Prüfsumme fertig
9bda   91  32       sta   ($32),y   Prüfsumme in Puffer schreiben
9bdc   68           pla             Pufferzeiger zurückholen
9bdd   a8           tay             und wieder setzen
9bde   ee  28  06    inc   $0628     aktuelle Sektornummer erhöhen
9be1   ad  28  06    lda   $0628     und
9be4   c5  43       cmp   $43       mit der maximalen Nummer vergleichen
9be6   90  b9       bcc   $9ba1     weitermachen, wenn kleiner
9be8   a9  03       lda   #$03      Pufferadresse Hi
9bea   85  31       sta   $31       setzen
9bec   20  30  fe    jsr   $fe30     Blockheader von Binär- in GCR-Werte umwandeln
9bef   a0  ba       ldy   #$ba      Zeiger für Puffer setzen
9bf1   b1  32       lda   ($32),y   Byte aus Puffer holen
9bf3   a2  45       ldx   #$45      Pufferadresse Lo
9bf5   86  32       stx   $32       neu setzen
9bf7   91  32       sta   ($32),y   und Byte an anderer Position wieder in Puffer
9bf9   a2  00       ldx   #$00      Pufferadresse Lo
9bfb   86  32       stx   $32       wieder auf alten Wert
9bfd   88           dey             Pufferzeiger vermindern
9bfe   c0  ff       cpy   #$ff      und alle Bytes verschieben, um
9c00   d0  ef       bne   $9bf1     Platz zu schaffen
9c02   a0  44       ldy   #$44      Pufferzeiger auf ersten, jetzt leeren Pufferteil
9c04   b9  bb  01    lda   $01bb,y   Byte aus Ausweichpuffer
9c07   91  32       sta   ($32),y   in den leeren Pufferteil übernehmen
9c09   88           dey             Zeiger auf nächstes Byte
9c0a   10  f8       bpl   $9c04     und weitermachen, bis alle Bytes übertragen
9c0c   18           clc             (unsinnige Operation)
9c0d   a9  03       lda   #$03      (unsinnige Operation)
9c0f   69  02       adc   #$02      (unsinnige Operation); heißt normal lda #$05
9c11   85  31       sta   $31       Pufferadresse Hi auf $05
9c13   a9  00       lda   #$00      Wert für Leerinhalt der Sektoren
9c15   a8           tay             Index in Puffer setzen
9c16   91  30       sta   ($30),y   Leerbyte in Puffer
9c18   c8           iny             auf nächstes Byte
9c19   d0  fb       bne   $9c16     und ganzen Puffer füllen
9c1b   20  e9  f5    jsr   $f5e9     Prüfsumme über Puffer berechnen
9c1e   85  3a       sta   $3a       und abspeichern
9c20   20  8f  f7    jsr   $f78f     Puffer in GCR umwandeln
9c23   a9  00       lda   #$00      Zeiger auf aktuellen Blockheader
9c25   85  1b       sta   $1b       auf ersten Header setzen
9c27   a2  06       ldx   #$06      1536 mal $55
9c29   20  63  9d    jsr   $9d63     auf Diskette schreiben
9c2c   a0  05       ldy   #$05      Zähler für SYNC-Markierung
9c2e   2c  0f  18    bit   $180f     warten, bis
9c31   30  fb       bmi   $9c2e     Diskcontroller bereit zum Schreiben
9c33   a9  ff       lda   #$ff      Byte für SYNC-Markierung
9c35   8d  01  1c    sta   $1c01     auf Diskette schreiben
9c38   2c  00  1c    bit   $1c00     SYNC-Flag löschen
9c3b   88           dey             und alle 5 Bytes
9c3c   d0  f0       bne   $9c2e     auf Diskette schreiben
9c3e   a2  0a       ldx   #$0a      Zähler für Blockheader
9c40   a4  1b       ldy   $1b       Zeiger auf aktuellen Header
9c42   2c  0f  18    bit   $180f     warten, bis
9c45   30  fb       bmi   $9c42     Diskcontroller bereit zum Schreiben
9c47   b1  32       lda   ($32),y   Headerbyte aus Puffer holen
9c49   8d  01  1c    sta   $1c01     und auf Diskette schreiben
9c4c   2c  00  1c    bit   $1c00     SYNC-Status löschen
```

```
9c4f   c8          iny             Zeiger auf nächstes Headerbyte
9c50   ca          dex             Zähler vermindern
9c51   d0  ef      bne   $9c42     und weitermachen, bis Header geschrieben
9c53   a0  09      ldy   #$09      Zähler für Lücke nach Blockheader
9c55   2c  0f  18  bit   $180f     warten, bis
9c58   30  fb      bmi   $9c55     Diskcontroller bereit für Schreiben
9c5a   a9  55      lda   #$55      Füllbyte für Lücke
9c5c   8d  01  1c  sta   $1c01     auf Diskette schreiben
9c5f   2c  00  1c  bit   $1c00     SYNC-Status löschen
9c62   88          dey             Zähler vermindern
9c63   d0  f0      bne   $9c55     und nächstes Lückenbyte schreiben
9c65   a9  ff      lda   #$ff      Wert für SYNC-Markierung
9c67   a0  05      ldy   #$05      Anzahl der Bytes pro SYNC-Markierung
9c69   2c  0f  18  bit   $180f     warten, bis
9c6c   30  fb      bmi   $9c69     Diskcontroller bereit für Schreiben
9c6e   8d  01  1c  sta   $1c01     SYNC-Byte auf Diskette schreiben
9c71   2c  00  1c  bit   $1c00     SYNC-Status löschen
9c74   88          dey             Zähler vermindern
9c75   d0  f2      bne   $9c69     und weitermachen, bis SYNC fertig
9c77   a0  bb      ldy   #$bb      Zeiger in Ausweichpuffer
9c79   2c  0f  18  bit   $180f     warten, bis
9c7c   30  fb      bmi   $9c79     Diskcontroller bereit für Schreiben
9c7e   b9  00  01  lda   $0100,y   Byte aus Ausweichpuffer
9c81   8d  01  1c  sta   $1c01     als Datenbyte auf Diskette schreiben
9c84   2c  00  1c  bit   $1c00     SYNC-Status löschen
9c87   c8          iny             Zeiger auf nächstes Byte
9c88   d0  ef      bne   $9c79     und gesamten Puffer auf Diskette schreiben
9c8a   2c  0f  18  bit   $180f     warten, bis
9c8d   30  fb      bmi   $9c8a     Diskcontroller bereit für Schreiben
9c8f   b1  30      lda   ($30),y   Byte aus Puffer holen
9c91   8d  01  1c  sta   $1c01     und als Datenbyte auf Diskette schreiben
9c94   2c  00  1c  bit   $1c00     SYNC-Status löschen
9c97   c8          iny             Zeiger auf nächstes Byte
9c98   d0  f0      bne   $9c8a     und gesamten Puffer auf Diskette schreiben
9c9a   a9  55      lda   #$55      Füllbyte für Lücke zwischen den Sektoren
9c9c   ac  26  06  ldy   $0626     Anzahl der Lückenbytes
9c9f   2c  0f  18  bit   $180f     warten, bis
9ca2   30  fb      bmi   $9c9f     Diskcontroller bereit für Schreiben
9ca4   8d  01  1c  sta   $1c01     Füllbyte auf Diskette schreiben
9ca7   2c  00  1c  bit   $1c00     SYNC-Status löschen
9caa   88          dey             Zähler für Lücke vermindern
9cab   d0  f2      bne   $9c9f     und weitermachen
9cad   a5  1b      lda   $1b       Zeiger auf aktuellen Blockheader
9caf   18          clc             plus 10 addieren
9cb0   69  0a      adc   #$0a      damit er auf den Anfang des nächsten Headers
9cb2   85  1b      sta   $1b       zeigt
9cb4   ce  28  06  dec   $0628     Anzahl der Sektoren vermindern
9cb7   f0  03      beq   $9cbc     verzweige, wenn Null
9cb9   4c  2c  9c  jmp   $9c2c     sonst weitermachen; Blöcke schreiben

9cbc   2c  0f  18  bit   $180f     warten, bis
9cbf   30  fb      bmi   $9cbc     Diskcontroller bereit
9cc1   2c  00  1c  bit   $1c00     SYNC-Status löschen
9cc4   2c  0f  18  bit   $180f     warten, bis
9cc7   30  fb      bmi   $9cc4     Diskcontroller bereit
9cc9   2c  00  1c  bit   $1c00     SYNC-Status löschen
9ccc   20  00  fe  jsr   $fe00     auf Lesen umschalten
9ccf   a9  c8      lda   #$c8      Anzahl der Versuche bei Verify
```

```
9cd1   8d 23 06   sta  $0623       setzen (200)
9cd4   a9 00      lda  #$00        Zeiger auf aktuellen Blockheader
9cd6   85 1b      sta  $1b         löschen
9cd8   a5 43      lda  $43         Anzahl der Sektoren des Tracks
9cda   8d 28 06   sta  $0628       als Zähler setzen
9cdd   20 54 97   jsr  $9754       SYNC-Signal abwarten
9ce0   a2 0a      ldx  #$0a        Zähler für Anzahl der Bytes des Blockheaders
9ce2   a4 1b      ldy  $1b         Zeiger auf aktuellen Blockheader
9ce4   b1 32      lda  ($32),y     Byte aus Puffer
9ce6   2c 0f 18   bit  $180f       warten, bis
9ce9   30 fb      bmi  $9ce6       Byte eingelesen
9ceb   cd 01 1c   cmp  $1c01       Byte mit Pufferinhalt vergleichen
9cee   d0 0e      bne  $9cfe       verzweige, wenn ungleich
9cf0   c8         iny              sonst Zeiger in Puffer erhöhen
9cf1   ca         dex              und Zähler für Header vermindern
9cf2   d0 f0      bne  $9ce4       weitermachen, bis ganzer Header verglichen
9cf4   18         clc              Addition vorbereiten
9cf5   a5 1b      lda  $1b         und Zeiger auf aktuellen Blockheader
9cf7   69 0a      adc  #$0a        auf nächsten Blockheader
9cf9   85 1b      sta  $1b         setzen
9cfb   4c 08 9d   jmp  $9d08       weiter; Datenblock vergleichen...
9cfe   ce 23 06   dec  $0623       Zähler für Versuche vermindern
9d01   d0 d1      bne  $9cd4       und weiter probieren, wenn ungleich 0
9d03   a9 06      lda  #$06        sonst Nummer der Fehlermeldung
9d05   4c 51 9d   jmp  $9d51       '24, READ ERROR' ausgeben
9d08   20 54 97   jsr  $9754       SYNC-Signal abwarten
9d0b   a0 bb      ldy  #$bb        Zeiger in Ausweichpuffer setzen
9d0d   b9 00 01   lda  $0100,y     Byte aus Ausweichpuffer
9d10   2c 0f 18   bit  $180f       warten, bis
9d13   30 fb      bmi  $9d10       Byte von Diskette eingelesen
9d15   cd 01 1c   cmp  $1c01       und Byte mit Pufferinhalt vergleichen
9d18   d0 e4      bne  $9cfe       verzweige, wenn ungleich
9d1a   c8         iny              sonst Zeiger in Puffer erhöhen
9d1b   d0 f0      bne  $9d0d       und weiter vergleichen
9d1d   b1 30      lda  ($30),y     Byte aus Puffer
9d1f   2c 0f 18   bit  $180f       warten, bis
9d22   30 fb      bmi  $9d1f       Byte eingelesen
9d24   cd 01 1c   cmp  $1c01       Byte mit Pufferinhalt vergleichen
9d27   d0 d5      bne  $9cfe       verzweige, wenn ungleich
9d29   c8         iny              Zeiger auf nächstes Byte
9d2a   d0 f1      bne  $9d1d       und weitermachen, bis ganzer Puffer verglichen
9d2c   ce 28 06   dec  $0628       Zähler für Sektoren vermindern
9d2f   d0 ac      bne  $9cdd       und weitermachen, bis ganze Spur geprüft
9d31   e6 51      inc  $51         Tracknummer erhöhen
9d33   a5 51      lda  $51         und
9d35   2c b1 01   bit  $01b1       je nach Diskettenseite
9d38   30 03      bmi  $9d3d       mit
9d3a   c9 24      cmp  #$24        dem Maximalwert 36 oder
9d3c   2c         .byte $2c        (nächsten Befehl überspringen)
9d3d   c9 47      cmp  #$47        mit dem Maximalwert 71 vergleichen
9d3f   b0 03      bcs  $9d44       in jedem Fall verzweigen, wenn größer
9d41   4c ca 99   jmp  $99ca       sonst zur Jobschleife; Kopf auf nächsten Track
9d44   a9 ff      lda  #$ff        Flag für laufende Formatierung
9d46   85 51      sta  $51         löschen
9d48   a9 00      lda  #$00        Flag für Pufferinhalt im GCR-Format
9d4a   85 50      sta  $50         löschen
9d4c   a9 01      lda  #$01        Nummer der Meldung
9d4e   4c b5 99   jmp  $99b5       '00, OK' ausgeben; Ende
```

```
------------------------------------------------------------------------
9d51                                    Fehlerausgang beim Formatieren.
9d51    ce  20  06      dec   $0620     Zähler für Versuche vermindern
9d54    f0  03          beq   $9d59     verzweige, wenn kein Versuch mehr
9d56    4c  ca  99      jmp   $99ca     sonst zur Jobschleife und weitermachen
9d59    a0  ff          ldy   #$ff      andernfalls Flag für laufende Formatierung
9d5b    84  51          sty   $51       löschen
9d5d    c8              iny             und
9d5e    84  50          sty   $50       Flag für Puffer im GCR-Format löschen
9d60    4c  b5  99      jmp   $99b5     Fehlernummer in A übergeben; Ende
------------------------------------------------------------------------
9d63                                    Füllbytes $55 auf Diskette schreiben. X bestimmt dabei die
                                        Anzahl der 256-Byte-Blöcke, die geschrieben werden sollen.
9d63    ad  0c  1c      lda   $1c0c     PCR lesen
9d66    29  1f          and   #$1f      und auf
9d68    09  c0          ora   #$c0      Schreiben
9d6a    8d  0c  1c      sta   $1c0c     umschalten
9d6d    a9  ff          lda   #$ff      Port für Schreib-/Lesekopf
9d6f    8d  03  1c      sta   $1c03     auf Ausgang schalten
9d72    a9  55          lda   #$55      Füllbyte
9d74    a0  00          ldy   #$00      Zähler Lo
9d76    2c  0f  18      bit   $180f     warten, bis
9d79    30  fb          bmi   $9d76     Diskcontroller bereit für Schreiben
9d7b    2c  00  1c      bit   $1c00     SYNC-Status löschen
9d7e    8d  01  1c      sta   $1c01     Byte auf Diskette schreiben
9d81    88              dey             Zähler Lo vermindern
9d82    d0  f2          bne   $9d76     und weitermachen
9d84    ca              dex             Zähler Hi vermindern
9d85    d0  ef          bne   $9d76     und weitermachen
9d87    60              rts             Ende
------------------------------------------------------------------------
9d88                                    IRQ-Routine für die Floppy im 1541-Modus.
9d88    48              pha             Akku retten
9d89    8a              txa             X.Register
9d8a    48              pha             retten
9d8b    98              tya             V-Register
9d8c    48              pha             ebenfalls retten
9d8d    ad  0d  40      lda   $400d     Steuerport für seriellen Bus lesen
9d90    29  08          and   #$08      wurde ein Byte eingelesen?
9d92    f0  26          beq   $9dba     verzweige, wenn nein
9d94    2c  af  02      bit   $02af     Flag für 1571-IRQ gesetzt?
9d97    30  21          bmi   $9dba     verzweige, wenn nein
9d99    ad  0f  18      lda   $180f     sonst Steuerregister lesen
9d9c    09  20          ora   #$20      und Elektronik auf 1571-Modus
9d9e    8d  0f  18      sta   $180f     (2 MHz) umschalten
9da1    a9  de          lda   #$de      Adresse Lo für IRQ-Routine
9da3    8d  a9  02      sta   $02a9     setzen
9da6    a9  9d          lda   #$9d      Adresse Hi
9da8    8d  aa  02      sta   $02aa     setzen; $9dde Vektor für 1571-Routine
9dab    a9  40          lda   #$40      Timer
9dad    8d  07  1c      sta   $1c07     auf 8 ms Takt
9db0    8d  05  1c      sta   $1c05     setzen und starten
9db3    a9  00          lda   #$00      Flag für Steppermotor in Aktion
9db5    85  62          sta   $62       löschen
9db7    4c  ea  9d      jmp   $9dea     weiter zur 1571-IRQ-Routine
9dba    ad  0d  18      lda   $180d     Flags für IRQ lesen
9dbd    29  02          and   #$02      und isolieren; IRQ vom seriellen Bus?
```

```
9dbf    f0  03       beq   $9dc4       verzweige, wenn nein
9dc1    20  53  e8   jsr   $e853       zur Behandlung des seriellen Bus
9dc4    ad  0d  1c   lda   $1c0d       Flags für IRQ lesen
9dc7    0a           asl               und isolieren
9dc8    10  03       bpl   $9dcd       verzweige, wenn kein IRQ vom Diskcontroller
9dca    20  b0  f2   jsr   $f2b0       sonst zur Jobschleife
9dcd    ba           tsx               Stackpointer merken
9dce    bd  04  01   lda   $0104,x     Prozessorstatus vom Stack holen
9dd1    29  10       and   #$10        und auf IRQ wegen BRK-Kommando testen
9dd3    f0  03       beq   $9dd8       verzweige, wenn kein BRK
9dd5    20  b0  f2   jsr   $f2b0       sonst zur Jobschleife
9dd8    68           pla               Y-Register zurückholen
9dd9    a8           tay               und setzen
9dda    68           pla               X-Register zurückholen
9ddb    aa           tax               und setzen
9ddc    68           pla               Akku zurückholen
9ddd    40           rti               und Ende des IRQ
-------------------------------------------------------------------------
9dde                                   IRQ-Routine für Floppy im 1571-Modus.
9dde    48           pha               Akku retten
9ddf    8a           txa               X-Register
9de0    48           pha               retten
9de1    98           tya               Y-Register
9de2    48           pha               ebenfalls retten
9de3    ad  0d  40   lda   $400d       Steuerregister für seriellen Bus lesen
9de6    29  08       and   #$08        wurde ein Byte eingelesen?
9de8    f0  08       beq   $9df2       verzweige, wenn nein
9dea    a5  37       lda   $37         sonst Flags für Busbetrieb holen und
9dec    09  40       ora   #$40        Flag für schnellen Busmodus
9dee    85  37       sta   $37         setzen
9df0    d0  22       bne   $9e14       unbedingter Sprung
9df2    ad  0d  18   lda   $180d       IRQ-Flags von Buscontroller holen
9df5    29  02       and   #$02        kam Interrupt vom seriellen Bus?
9df7    f0  07       beq   $9e00       verzweige, wenn nein
9df9    2c  01  18   bit   $1801       sonst IRQ-Flag löschen
9dfc    a9  01       lda   #$01        und Flag für ATN
9dfe    85  7c       sta   $7c         setzen
9e00    ba           tsx               Stackpointer merken
9e01    bd  04  01   lda   $0104,x     Prozessorstatus holen
9e04    29  10       and   #$10        und auf IRQ durch BRK-Kommando prüfen
9e06    f0  03       beq   $9e0b       verzweige, wenn kein BRK
9e08    20  ba  92   jsr   $92ba       sonst Jobschleife aufrufen
9e0b    ad  0d  1c   lda   $1c0d       IRQ-Flags von Diskcontroller holen
9e0e    0a           asl               und auf IRQ vom Diskcontroller testen
9e0f    10  03       bpl   $9e14       verzweige, wenn nicht aufgetreten
9e11    20  ba  92   jsr   $92ba       sonst Jobschleife aufrufen
9e14    68           pla               Y-Register zurückholen
9e15    a8           tay               und setzen
9e16    68           pla               X-Register zurückholen
9e17    aa           tax               und setzen
9e18    68           pla               Akku zurückholen
9e19    40           rti               Ende
-------------------------------------------------------------------------
9e1a ff ...                            unbenutzter
9f0c ... ff                            Leerbereich
```

```
------------------------------------------------------------------------
9f0d                                    Tabelle für die Umwandlung von GCR-(5 Bit)- ibbles in die
                                        entsprechende Binärnibbles. Die Tabelle ist hierbei für die
                                        Umwandlung der Nibbles 2, 4, 5 und 7 zuständig. $ff
                                        bezeichnet dabei die illegalen GCR-Werte.
9f0d 0c 04 05 ff ff 02 03 ff 0f 06 07 ff 09 0a 0b ff
9f1d 0d 0e 80 ff 00 00 10 40 ff 20 c0 60 40 a0 50 e0
9f2d ff ff ff 02 20 08 30 ff ff 00 f0 ff 60 01 70 ff
9f3d ff ff 90 03 a0 0c b0 ff ff 04 d0 ff e0 05 80 ff
9f4d 90 ff 08 0c ff 0f 09 0d 80 02 ff ff ff 03 ff ff
9f5d 00 ff ff 0f ff 0f ff ff 10 06 ff ff ff 07 00 20
9f6d a0 ff ff 06 ff 09 ff ff c0 0a ff ff ff 0b ff ff
9f7d 40 ff ff 07 ff 0d ff ff 50 0e ff ff ff ff 10 30
9fBd b0 ff 00 04 02 06 0a 0e 80 ff ff ff ff ff ff ff
9f9d 20 ff 08 09 80 10 c0 50 30 30 f0 70 90 b0 d0 ff
9fad ff ff 00 0a ff ff ff ff f0 00 ea b5 00 30 fc 60
------------------------------------------------------------------------
9fb6                                    Routine erzwingt die Ausführung eines Jobs durch den BRK-
                                        Befehl, der einen IRQ auslöst. Aus Platzersparnis wurde
                                        diese kleine Routine in die Umwandlungstabelle gelegt, wo
                                        sonst normalerweise $ff-Bytes stehen würden. Nach Aufruf
                                        wird das Ende des Jobs abgewartet und die Rückmeldung in
                                        den Akku übernommen.
9fb6    00          brk                 Kommando zum Auslösen eines IRQ
9fb7    ea          nop                 Rückkehr nach Ende des IRQ nach $9fbB
9fb8    b5  00      lda   $00,x        Rückmeldung aus Jobspeicher
9fba    30  fc      bmi   $9fb8        warten, bis Job beendet
9fbc    60          rts                 Ende
------------------------------------------------------------------------
9fbd                                    Fortsetzung der Umwandlungstabelle...
9fbd 60 ff 01 0b ff ff ff ff 70 ff ff ff ff ff c0 f0
9fcd d0 ff 01 05 03 07 0b ff 90 ff ff ff ff ff ff ff
9fdd a0 ff 0c 0d ff ff ff ff b0 ff ff ff ff ff 40 60
9fed e0 ff 04 0e ff ff ff ff d0 ff ff ff ff ff ff ff
9ffd e0 ff 05 ff ff ff ff ff ff ff ff ff ff ff 50 70
------------------------------------------------------------------------
a00d                                    GCR-Binär-Umwandlungstabelle für 1. GCR-Nibble.
a00d 0c 04 05 ff ff 02 03 ff 0f 06 07 ff 09 0a 0b ff
a01d 0d 0e 80 ff 00 00 10 40 ff 20 c0 60 40 a0 50 e0
a02d ff ff ff 02 20 08 30 30 30 00 f0 ff 60 01 70 ff
a03d ff ff 90 03 a0 0c b0 ff ff 04 d0 ff e0 05 80 ff
a04d 90 ff 08 0c ff 0f 09 0d 80 80 80 80 80 80 80 80
a05d 00 00 00 00 00 00 00 00 10 10 10 10 10 10 10 10
a06d a0 ff ff 06 ff 09 ff ff c0 c0 c0 c0 c0 c0 c0 c0
a07d 40 40 40 40 40 40 40 40 50 50 50 50 50 50 50 50
a08d b0 ff 00 04 02 06 0a 0e 80 80 80 80 80 80 80 80
a09d 20 20 20 20 20 20 20 20 30 30 30 30 30 30 30 30
a0ad ff ff 00 0a 0a 0a 0a 0a f0 f0 f0 f0 f0 f0 f0 f0
a0bd 60 60 60 60 60 60 60 60 70 70 70 70 70 70 70 70
a0cd d0 ff 01 05 03 07 0b ff 90 90 90 90 90 90 90 90
a0dd a0 a0 a0 a0 a0 a0 a0 a0 b0 b0 b0 b0 b0 b0 b0 b0
a0ed e0 ff 04 0e ff ff ff ff d0 d0 d0 d0 d0 d0 d0 d0
a0fd e0 e0 e0 e0 e0 e0 e0 e0 05 05 05 05 05 05 50 70
```

```
------------------------------------------------------------------------
a10d                                  GCR-Binär-Umwandlungstabelle für 3. GCR-Nibble.
a10d ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
a11d ff ff 80 80 00 00 10 10 ff ff c0 c0 40 40 50 50
a12d ff ff ff ff 20 20 30 30 ff ff f0 f0 60 60 70 70
a13d ff ff 90 90 a0 a0 b0 b0 ff ff d0 d0 e0 e0 ff ff
a14d ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
a15d ff ff 80 80 00 00 10 10 ff ff c0 c0 40 40 50 50
a16d ff ff ff ff 20 20 30 30 ff ff f0 f0 60 60 70 70
a17d ff ff 90 90 a0 a0 b0 b0 ff ff d0 d0 e0 e0 ff ff
a18d ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
a19d ff ff 80 80 00 00 10 10 ff ff c0 c0 40 40 50 50
a1ad ff ff ff ff 20 20 30 30 ff ff f0 f0 60 60 70 70
a1bd ff ff 90 90 a0 a0 b0 b0 ff ff d0 d0 e0 e0 ff ff
a1ed ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
a1dd ff ff 80 80 00 00 10 10 ff ff c0 c0 40 40 50 50
a1ed ff ff ff ff 20 20 30 30 ff ff f0 f0 60 60 70 70
a1fd ff ff 90 90 a0 a0 b0 b0 ff ff d0 d0 e0 e0 ff ff
------------------------------------------------------------------------
a20d                                  GCR-Binär-Umwandlungstabelle für 6. GCR-Nibble.
a20d ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
a21d ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
a22d ff ff ff ff 08 08 08 08 00 00 00 00 01 01 01 01
a23d ff ff ff ff 0c 0c 0c 0c 04 04 04 04 05 05 05 05
a24d ff ff ff ff ff ff ff ff 02 02 02 02 03 03 03 03
a25d ff ff ff ff 0f 0f 0f 0f 06 06 06 06 07 07 07 07
a26d ff ff ff ff 09 09 09 09 0a 0a 0a 0a 0b 0b 0b 0b
a27d ff ff ff ff 0d 0d 0d 0d 0e 0e 0e 0e ff ff ff ff
a28d ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
a29d ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
a2ad ff ff ff ff 08 08 08 08 00 00 00 00 01 01 01 01
a2bd ff ff ff ff 0c 0c 0c 0c 04 04 04 04 05 05 05 05
a2ed ff ff ff ff ff ff ff ff 02 02 02 02 03 03 03 03
a2dd ff ff ff ff 0f 0f 0f 0f 06 06 06 06 07 07 07 07
a2ed ff ff ff ff 09 09 09 09 0a 0a 0a 0a 0b 0b 0b 0b
a2fd ff ff ff ff 0d 0d 0d 0d 0e 0e 0e 0e ff ff ff ff
------------------------------------------------------------------------
a30d                                  GCR-Binär-Umwandlungstabelle für 8. GCR-Nibble.
a30d ff ff ff ff ff ff ff ff ff 08 00 01 ff 0c 04 05
a31d ff ff 02 03 ff 0f 06 07 ff 09 0a 0b ff 0d 0e ff
a32d ff ff ff ff ff ff ff ff ff 08 00 01 ff 0c 04 05
a33d ff ff 02 03 ff 0f 06 07 ff 09 0a 0b ff 0d 0e ff
a34d ff ff ff ff ff ff ff ff ff 08 00 01 ff 0c 04 05
a35d ff ff 02 03 ff 0f 06 07 ff 09 0a 0b ff 0d 0e ff
a36d ff ff ff ff ff ff ff ff ff 08 00 01 ff 0c 04 05
a37d ff ff 02 03 ff 0f 06 07 ff 09 0a 0b ff 0d 0e ff
a38d ff ff ff ff ff ff ff ff ff 08 00 01 ff 0c 04 05
a39d ff ff 02 03 ff 0f 06 07 ff 09 0a 0b ff 0d 0e ff
a3ad ff ff ff ff ff ff ff ff ff 08 00 01 ff 0c 04 05
a3bd ff ff 02 03 ff 0f 06 07 ff 09 0a 0b ff 0d 0e ff
a3ed ff ff ff ff ff ff ff ff ff 08 00 01 ff 0c 04 05
a3dd ff ff 02 03 ff 0f 06 07 ff 09 0a 0b ff 0d 0e ff
a3ed ff ff ff ff ff ff ff ff ff 08 00 01 ff 0c 04 05
a3fd ff ff 02 03 ff 0f 06 07 ff 09 0a 0b ff 0d 0e ff
```

```
------------------------------------------------------------------------
a40d                                    Formatieren einer Diskette im GCR-Format.
a40d    a9  47      lda   #$47         Anzahl der zu formatierenden Tracks
*1 a40f 8d  ac  02  sta   $02ac        setzen
*0 a40f 8d  d7  fe  sta   $fed7        (gilt nicht für 1570)
a412    a9  03      lda   #$03         Puffernummer
a414    20  d3  d6  jsr   $d6d3        Parameter für Diskcontroller setzen
a417    a2  03      ldx   #$03         Puffernummer
a419    a9  00      lda   #$00         Diskettenseite 0
a41b    8d  b2  01  sta   $01b2        setzen
a41e    a9  f0      lda   #$f0         Jobcode für Formatieren einer Diskette
a420    85  3b      sta   $3b          setzen
a422    95  00      sta   $00,x        und an Diskcontroller übergeben
a424    20  b6  9f  jsr   $9fb6        Job ausführen und Ende abwarten
a427    c9  02      cmp   #$02         Fehler?
a429    b0  45      bcs   $a470        verzweige, wenn ja
a42b    a0  03      ldy   #$03         4 Leseversuche setzen (3+1, da Zählung mit 0)
a42d    a9  01      lda   #$01         Tracknummer 1
a42f    85  0c      sta   $0c          für Puffer 3 in Jobspeicher
a431    a9  00      lda   #$00         Sektornummer 0
a433    85  0d      sta   $0d          für Puffer 3 in Jobspeicher
a435    a9  80      lda   #$80         Jobcode für Block lesen
a437    95  00      sta   $00,x        an Diskcontroller übergeben
a439    20  b6  9f  jsr   $9fb6        Job ausführen und Ende abwarten
a43c    c9  02      cmp   #$02         Fehler?
a43e    90  05      bcc   $a445        verzweige, wenn nein
a440    88          dey                Zähler für Versuche vermindern
a441    10  ea      bpl   $a42d        und noch einmal probieren
a443    b0  2b      bcs   $a470        unbedingter Sprung; Fehler
*1 a445 a9  01      lda   #$01         Diskettenseite 1
*0 a445 60          rts                keine Behandlung für zweite Diskettenseite
*0 a446 ea          nop                (Rest)
a447    8d  b2  01  sta   $01b2        setzen
a44a    a9  f0      lda   #$f0         Jobcode für Formatieren einer Diskette
a44c    85  3b      sta   $3b          setzen
a44e    95  00      sta   $00,x        und an Diskcontroller übergeben
a450    20  b6  9f  jsr   $9fb6        Job ausführen und Ende abwarten
a453    c9  02      cmp   #$02         Fehler?
a455    b0  19      bcs   $a470        verzweige, wenn ja
a457    a0  03      ldy   #$03         Anzahl der Leseversuche (3+1)
a459    a9  24      lda   #$24         Tracknummer 36
a45b    85  0c      sta   $0c          für Puffer 3 in Jobspeicher
a45d    a9  00      lda   #$00         Sektornummer 0
a45f    85  0d      sta   $0d          für Puffer 3 in Jobspeicher
a461    a9  80      lda   #$80         Jobcode für Block lesen
a463    95  00      sta   $00,x        an Diskcontroller übergeben
a465    20  b6  9f  jsr   $9fb6        Job ausführen und Ende abwarten
a468    c9  02      cmp   #$02         Fehler?
a46a    b0  01      bcs   $a46d        verzweige, wenn ja
a46c    60          rts                Ende; alles ok
a46d    88          dey                Zähler für Versuche vermindern
a46e    10  e9      bpl   $a459        und nochmal probieren
a470    a2  00      ldx   #$00         Flag für Fehler setzen
a472    2c  98  02  bit   $0298        vorherigen Fehlerstatus prüfen
a475    8e  98  02  stx   $0298        neuen Status setzen
a478    10  01      bpl   $a47b        verzweige, wenn Fehlerbehandlung erfolgen soll
a47a    60          rts                sonst Ende; alles ok
a47b    4c  0a  e6  jmp   $e60a        zur Fehlerbehandlung; Ende
```

```
--------------------------------------------------------------------------
a47e                                    Verzögerungsschleife von ca. 30 Taktzyklen.
a47e   8a           txa                 X-Register merken
a47f   a2  05       ldx   #$05          Zählwert für Verzögerung
a481   d0  03       bne   $a486         unbedingter Sprung; weiter...
--------------------------------------------------------------------------
a483                                    Verzögerungsschleife von ca. 70 Taktzyklen.
a483   8a           txa                 X-Register merken
a484   a2  0d       ldx   #$0d          Zählwert für Verzögerung
a486   ca           dex                 Zähler vermindern
a487   d0  fd       bne   $a486         und weitermachen
a489   aa           tax                 X-Register zurückholen
a48a   60           rts                 Ende
--------------------------------------------------------------------------
a48b                                    Pufferzeiger für BAM retten.
a48b   a5  6d       lda   $6d          Pufferzeiger Lo
a48d   8d  ad  02   sta   $02ad        merken
a490   a5  6e       lda   $6e          Pufferzeiger H i
a492   8d  ae  02   sta   $02ae        merken
a495   60           rts                 Ende
--------------------------------------------------------------------------
a496                                    Pufferzeiger für BAM wieder holen.
a496   ad  ad  02   lda   $02ad        Pufferzeiger Lo
a499   85  6d       sta   $6d          wieder setzen
a49b   ad  ae  02   lda   $02ae        Pufferzeiger Hi
a49e   85  6e       sta   $6e          wieder setzen
a4a0   60           rts                 Ende
--------------------------------------------------------------------------
a4a1                                    Erweiterung der Routine zum Setzen des BAM- Zeigers auf ein
                                        bestimmtes Bitmuster für die zweite Diskettenseite.
a4a1   a6  7f       ldx   $7f          aktuelle Drivenummer
a4a3   bd  ff  00   lda   $00ff,x      Drivestatus holen
a4a6   f0  05       beq   $a4ad        verzweige, wenn in Ordnung
a4a8   a9  74       lda   #$74         sonst Nummer der Fehlermeldung
a4aa   20  48  e6   jsr   $e648        '74, DRIVE NOT READY' ausgeben
a4ad   20  19  f1   jsr   $f119        Kanalnummer für BAM holen
a4b0   20  df  f0   jsr   $f0df        BAM von Diskette lesen
a4b3   ad  f9  02   lda   $02f9        BAM 'dirty' Flag
a4b6   f0  07       beq   $a4bf        verzweige, wenn nicht gesetzt
a4b8   09  80       ora   #$80         sonst setzen
a4ba   8d  f9  02   sta   $02f9        und wieder abspeichern
a4bd   d0  03       bne   $a4c2        unbedingter Sprung
a4bf   20  8d  a5   jsr   $a58d        BAM auf Diskette schreiben
a4c2   20  8b  a4   jsr   $a48b        Pufferzeiger für BAM merken
a4c5   20  34  a5   jsr   $a534        neuen Pufferzeiger setzen
a4c8   a5  80       lda   $80          aktuelle Tracknummer
a4ca   38           sec                 minus 36
a4cb   e9  24       sbc   #$24         um Absolutwert für zweite Seite
a4cd   a8           tay                 zu erhalten
a4ce   b1  6d       lda   ($6d),y      entsprechenden Eintrag holen
a4d0   48           pha                 und merken
a4d1   20  96  a4   jsr   $a496        Pufferzeiger für BAM wieder holen
a4d4   68           pla                 BAM-Eintrag zurückholen
a4d5   60           rts                 Ende
--------------------------------------------------------------------------
a4d6 ff ...                             unbenutzter
a4dc ... ff                             Leerbereich
```

```
------------------------------------------------------------------------
*1 a4dd ff ...                          unbenutzter Leerbereich
*1 a4e6 ... ff                          bei der 1571
------------------------------------------------------------------------
*0 a4dd                                 Copyright Statement für 1570-Änderungen.
*0 a4dd 53 54 45 56 45 20 4c 41         steve la
*0 a4e5 4d 0d                           m
------------------------------------------------------------------------
a4e7                                    BAM-Belegung eines Sektors holen. Erweiterung für zweite
                                        Diskettenseite.
a4e7   a5  80       lda   $80           aktuelle Tracknummer
a4e9   38           sec                 minus 36
a4ea   e9  24       sbc   #$24          für Absolutwert der Tracknummer
a4ec   a8           tay                 setzen
a4ed   a5  81       lda   $81           aktuelle Sektornummer
a4ef   4a           lsr                 geteilt durch 8,
a4f0   4a           lsr                 da ein Byte für
a4f1   4a           lsr                 8 Sektoren zuständig ist
a4f2   18           clc                 Byteposition
a4f3   79  db  a5   adc   $a5db,y       plus Trackposition in der BAM
a4f6   a8           tay                 als Index
a4f7   a5  81       lda   $81           Sektornummer
a4f9   29  07       and   #$07          auf 0 bis 7 begrenzen
a4fb   aa           tax                 und als Index
a4fc   b9  46  01   lda   $0146,y       Bitmuster aus BAM holen
a4ff   3d  e9  ef   and   $efe9,x       und entsprechendes Sektor-Bit isolieren
a502   08           php                 Bit merken
a503   b9  46  01   lda   $0146,y       Bitmuster nochmal holen
a506   28           plp                 und Bit-Status zurückholen
a507   60           rts                 Ende
------------------------------------------------------------------------
a508                                    Anzahl der freien Blöcke eines Tracks in der BAM nach
                                        Freigeben eines Blocks erhöhen.
a508   20  8b  a4   jsr   $a48b         Pufferzeiger für BAM merken
a50b   20  34  a5   jsr   $a534         neuen Pufferzeiger setzen
a50e   a5  80       lda   $80           aktuelle Tracknummr
a510   38           sec                 minus 36,
a511   e9  24       sbc   #$24          um den Absolutwert
a513   a8           tay                 für die zweite Diskettenseite zu bekommen
a514   18           clc                 und
a515   b1  6d       lda   ($6d),y       entsprechende Anzahl der freien Blöcke der Spur
a517   69  01       adc   #$01          um eins
a519   91  6d       sta   ($6d),y       erhöhen
a51b   4c  96  a4   jmp   $a496         Pufferzeiger für BAM wieder holen; Ende
------------------------------------------------------------------------
a51e                                    Anzahl der freien Blöcke eines Tracks in der BAM nach
                                        Belegen eines Blocks vermindern.
a51e   20  8b  a4   jsr   $a48b         Pufferzeiger für BAM merken
a521   20  34  a5   jsr   $a534         neuen Pufferzeiger setzen
a524   a5  80       lda   $80           aktuelle Tracknummer
a526   38           sec                 minus 36,
a527   e9  24       sbc   #$24          um den Absolutwert
a529   a8           tay                 für die zweite Diskettenseite zu bekommen
a52a   38           sec                 und
a52b   b1  6d       lda   ($6d),y       entsprechende Anzahl der freien Blöcke der Spur
a52d   e9  01       sbc   #$01          um eins
a52f   91  6d       sta   ($6d),y       vermindern
a531   4c  96  a4   jmp   $a496         Pufferzeiger für BAM wieder holen; Ende
```

```
------------------------------------------------------------------------
a534                                    Pufferzeiger für BAM setzen.
a534    a2  0d       ldx   #$0d         Kanalnummer 13
a536    b5  a7       lda   $a7,x        zugehörige Puffernummer holen
a538    29  0f       and   #$0f         und isolieren
a53a    aa           tax                als Index
a53b    bd  e0  fe   lda   $fee0,x      Pufferadresse Hi holen
a53e    85  6e       sta   $6e          und setzen
a540    a9  dd       lda   #$dd         Pufferadresse Lo holen
a542    85  6d       sta   $6d          und setzen
a544    60           rts                Ende
------------------------------------------------------------------------
a545                                    BAM der zweiten Diskettenseite auf Richtigkeit der Angaben
                                        über die freien Blöcke überprüfen.
a545    a5  6f       lda   $6f          Wert aus Zwischenspeicher
a547    48           pha                merken
a548    a5  80       lda   $80          aktuelle Tracknummer
a54a    38           sec                minus 36, um
a54b    e9  24       sbc   #$24         den BAM-Absolutwert
a54d    a8           tay                zu erhalten
a54e    48           pha                Wert merken
a54f    20  8b  a4   jsr   $a48b        Pufferzeiger der BAM merken
a552    20  34  a5   jsr   $a534        neuen Pufferzeiger setzen
a555    b1  6d       lda   ($6d),y      Anzahl der freien Blöcke des Tracks
a557    48           pha                merken
a558    a9  00       lda   #$00         ersten Additionswert
a55a    85  6f       sta   $6f          setzen
a55c    a9  01       lda   #$01         Pufferadresse Hi für erweiterte BAM
a55e    85  6e       sta   $6e          der zweiten Diskettenseite setzen
a560    b9  db  a5   lda   $a5db,y      Position des Bitmusters in der BAM holen
a563    18           clc                und
a564    69  46       adc   #$46         auf die Pufferadresse ($0146 bis $01ba)
a566    85  6d       sta   $6d          umrechnen
a568    a0  02       ldy   #$02         Anzahl der Bitmuster pro Track minus 1
a56a    a2  07       ldx   #$07         Anzahl der Bits pro Byte minus 1
a56c    b1  6d       lda   ($6d),y      Bitmuster holen
a56e    3d  e9  ef   and   $efe9,x      und aktuelles Bit isolieren
a571    f0  02       beq   $a575        verzweige, wenn Block belegt
a573    e6  6f       inc   $6f          sonst Anzahl der freien Blöcke erhöhen
a575    ca           dex                Zähler für Bits vermindern
a576    10  f4       bpl   $a56c        und nächstes Bit untersuchen
a578    88           dey                Zähler für Bytes vermindern
a579    10  ef       bpl   $a56a        und nächstes Byte untersuchen
a57b    68           pla                Anzahl der freien Blöcke des Tracks
a57c    c5  6f       cmp   $6f          mit neu errechnetem Wert vergleichen
a57e    f0  05       beq   $a585        verzweige, wenn richtig
a580    a9  71       lda   #$71         sonst Nummer der Fehlermeldung
a582    20  45  e6   jsr   $e645        '71, DIR ERROR' ausgeben
a585    68           pla                aktuelle Tracknummer wieder holen
a586    a8           tay                und als Index
a587    68           pla                Zwischenspeicherwert zurückholen
a588    85  6f       sta   $6f          und wieder setzen
a58a    4c  96  a4   jmp   $a496        Pufferzeiger der BAM zurückholen; Ende
------------------------------------------------------------------------
a58d                                    BAM auf Diskette schreiben. Erweiterung für zweiseitigen
                                        Diskettenbetrieb.
a58d    ad  0f  18   lda   $180f        Steuerregister lesen
a590    29  20       and   #$20         Floppy im 1541-Modus?
```

```
a592    d0  03       bne   $a597       verzweige, wenn nein
a594    4c  8a  d5   jmp   $d58a       sonst BAM schreiben (für 1541); Ende
*1 a597 ad  ac  02   lda   $02ac       maximale Tracknummer der Diskette
*0 a597 ad  d7  fe   lda   $fed7       maximale Tracknummer holen
a59a    c9  25       cmp   #$25        mit 35 vergleichen
a59c    90  f6       bcc   $a594       verzweige, wenn kleiner
a59e    a6  f9       ldx   $f9         sonst Puffernummer holen
a5a0    bd  5b  02   lda   $025b,x     und Jobcode holen
a5a3    48           pha               Jobcode merken
a5a4    20  8a  d5   jsr   $d58a       BAM für Seite 0 auf Diskette
a5a7    20  8b  a4   jsr   $a48b       Pufferzeiger für BAM merken
a5aa    20  3a  ef   jsr   $ef3a       BAM von Diskette lesen
a5ad    20  08  f0   jsr   $f008       BAM-Puffer löschen
a5b0    a5  f9       lda   $f9         Puffernummer
a5b2    0a           asl               mal 2
a5b3    aa           tax               als Index
a5b4    a9  35       lda   #$35        Track 18 auf Seite 1 (insgesamt Track 53)
a5b6    95  06       sta   $06,x       in Jobspeicher
a5b8    a0  68       ldy   #$68        Pufferzeiger auf Ende der 1571-BAM
a5ba    b9  46  01   lda   $0146,y     Byte aus BAM holen
a5bd    91  6d       sta   ($6d),y     und in aktuellen Puffer schreiben
a5bf    88           dey               nächstes Byte
a5c0    10  f8       bpl   $a5ba       weitermachen, bis ganzer Puffer übertragen
a5c2    20  96  a4   jsr   $a496       Pufferzeiger für BAM wieder holen
a5c5    20  8a  d5   jsr   $d58a       BAM auf Diskette schreiben
a5c8    a5  f9       lda   $f9         Puffernummer
a5ca    0a           asl               mal 2
a5cb    aa           tax               als Index
a5cc    ad  85  fe   lda   $fe85       Tracknummer der BAM holen (18)
a5cf    95  06       sta   $06,x       und in Jobspeicher
a5d1    20  86  d5   jsr   $d586       BAM von Diskette lesen
a5d4    68           pla               Jobcode wieder holen
a5d5    a6  f9       ldx   $f9         Puffernummer
a5d7    9d  5b  02   sta   $025b,x     und Jobcode wieder setzen
a5da    60           rts               Ende
--------------------------------------------------------------------------
a5db                                   Tabelle mit den Positionen der einzelnen BAM-Einträge für
                                       jeden Track.
a5db 00 03 06 09 0c 0f 12 15 18 1b 1e 21 24 27 2a 2d
a5eb 30 33 36 39 3c 3f 42 45 48 4b 4e 51 54 57 5a 5d
a5fb 60 63 66
--------------------------------------------------------------------------
a5fe                                   Erweiterung des &-Befehls für zweiseitigen
                                       Diskettenbetrieb.
a5fe    ad  0f  18   lda   $180f       Steuerport lesen
a601    29  20       and   #$20        Floppy im 1541-Modus?
a603    f0  0f       beq   $a614       verzweige, wenn nein
a605    a0  00       ldy   #$00        (unsinniger Befehl)
a607    a2  00       ldx   #$00        (unsinniger Befehl)
a609    a9  01       lda   #$01        Zeiger für Beginn des INPUT-Puffers
a60b    8d  7a  02   sta   $027a       setzen
a60e    20  12  c3   jsr   $c312       Drivenummer holen und setzen
a611    4c  a8  e7   jmp   $e7a8       zurück zur &-Routine
a614    a9  8d       lda   #$8d        ASCII-Code für 'SHIFT RETURN'
a616    20  68  c2   jsr   $c268       Endezeichen im Befehlsstring suchen
a619    4c  a8  e7   jmp   $e7a8       zurück zur &-Routine; Ende
```

```
------------------------------------------------------------------------
a61c                                    Erweiterte Routine zum Analysieren und Ausführen von
                                        Befehlen im Befehlsstring.
a61c   20  46  c1    jsr   $c146       Befehl analysieren und ausführen
a61f   20  b2  81    jsr   $81b2       seriellen Bus auf Eingang
a622   a5  37        lda   $37         Flags für Busbetrieb
a624   29  7f        and   #$7f        auf langsamen 1541-Bus
a626   85  37        sta   $37         stellen
a628   4c  ff  eb    jmp   $ebff       zur Systemwarteschleife
------------------------------------------------------------------------
a62b                                    Ausschaltverzögerung für Drivemotor setzen.
a62b   a9  ff        lda   #$ff        Zähler Lo
a62d   85  48        sta   $48         setzen
a62f   a9  06        lda   #$06        Zähler Hi
a631   85  35        sta   $35         setzen
a633   60            rts               Ende
------------------------------------------------------------------------
a634                                    Drivemotor einschalten, wenn die Diskette gewechselt wurde,
                                        um sie zu zentrieren
a634   d0  07        bne   $a63d       verzweige, wenn Diskette gewechselt wurde
a636   ad  ab  02    lda   $02ab       Zähler für Anlaufen des Motors prüfen
a639   d0  10        bne   $a64b       verzweige, wenn Motor schon läuft
a63b   f0  1a        beq   $a657       unbedingter Sprung
a63d   a9  ff        lda   #$ff        Zähler für Anlaufen bei Zentrierung
a63f   8d  ab  02    sta   $02ab       setzen
a642   20  64  87    jsr   $8764       Drivemotor einschalten
a645   a9  01        lda   #$01        Flag für Diskettenwechsel
a647   85  1c        sta   $1c         setzen
a649   d0  0c        bne   $a657       unbedingter Sprung
a64b   ce  ab  02    dec   $02ab       Zähler für Motor bei Zentrierung vermindern
a64e   d0  07        bne   $a657       verzweige, wenn ungleich Null
a650   a5  20        lda   $20         sonst Flags für Drivestatus holen
a652   d0  03        bne   $a657       verzweige, wenn Flag für laufenden Motor gesetzt
a654   20  70  87    jsr   $8770       sonst Motor ausschalten
a657   4c  b1  f9    jmp   $f9b1       zurück zur Jobschleife für Kopfsteuerung
------------------------------------------------------------------------
a65a                                    Buscontroller für RESET initialisieren.
a65a   a9  02        lda   #$02        DATA OUT-Leitung auf Lo
a65c   8d  00  18    sta   $1800       setzen
a65f   a9  20        lda   #$20        Floppy auf 1571-Modus; Diskseite 0; seriellen
a661   8d  01  18    sta   $1801       Bus auf Eingang schalten
a664   4c  18  ff    jmp   $ff18       zurück zur RESET-Routine
------------------------------------------------------------------------
a667                                    BAM von Diskette lesen. Erweiterung für zweiseitigen
                                        Diskettenbetrieb.
a667   ad  0f  18    lda   $180f       Steuerport lesen
a66a   29  20        and   #$20        Floppy im 1541-Modus?
a66c   d0  03        bne   $a671       verzweige, wenn nein
a66e   4c  86  d5    jmp   $d586       zur Routine für 1541; BAM lesen
*1 a671 ad  ac  02    lda   $02ac       maximale Tracknummer holen
*0 a671 ad  d7  fe    lda   $fed7       maximale Tracknummer holen
a674   c9  25        cmp   #$25        mit 35 vergleichen
a676   90  f6        bcc   $a66e       verzweige, wenn kleiner
a678   20  8b  a4    jsr   $a48b       Pufferzeiger für BAM merken
a67b   a9  00        lda   #$00        Pufferzeiger für BAM
a67d   85  6d        sta   $6d         auf Null setzen
a67f   a6  f9        ldx   $f9         Puffernummer
a681   bd  e0  fe    lda   $fee0,x     Pufferadresse Hi holen
```

```
a684    85  6e       sta   $6e         und setzen
a686    a9  ff       lda   #$ff        Flag für Fehler ignorieren
a688    8d  98  02   sta   $0298       $0298 setzen
a68b    a5  f9       lda   $f9         Puffernummer
a68d    0a           asl               mal 2
a68e    aa           tax               als Index
a68f    a9  35       lda   #$35        53; Track 18 der Diskettenseite 1
a691    95  06       sta   $06,x       in Jobspeicher
a693    20  86  d5   jsr   $d586       Block von Diskette lesen
a696    c9  02       cmp   #$02        Fehler?
a698    6a           ror               Ergebnis in Bit 7 des Akkus
a699    29  80       and   #$80        auf Fehler testen
a69b    49  80       eor   #$80        und Fehlerbit invertieren
a69d    8d  af  01   sta   $01af       Fehlerstatus merken (0 heißt Fehler)
a6a0    10  0a       bpl   $a6ac       verzweige, wenn Fehler aufgetreten
a6a2    a0  68       ldy   #$68        sonst Zeiger auf Ende der 1571-BAM
a6a4    b1  6d       lda   ($6d),y     Byte aus aktuellem Puffer
a6a6    99  46  01   sta   $0146,y     in BAM-Puffer für zweite Seite übertragen
a6a9    88           dey               Zeiger auf nächstes Byte
a6aa    10  f8       bpl   $a6a4       und gesamten Puffer übertragen
a6ac    a9  ff       lda   #$ff        Flag für Fehler ignorieren
a6ae    8d  98  02   sta   $0298       setzen
a6b1    a5  f9       lda   $f9         Puffernummer
a6b3    0a           asl               mal 2
a6b4    aa           tax               als Index
a6b5    ad  85  fe   lda   $fe85       Tracknummer der BAM (18) holen
a6b8    95  06       sta   $06,x       und in Jobspeicher
a6ba    20  86  d5   jsr   $d586       Block von Diskette lesen
a6bd    c9  02       cmp   #$02        Fehler?
a6bf    90  10       bcc   $a6d1       verzweige, wenn nein
a6c1    aa           tax               Fehlernummer merken
a6c2    a9  24       lda   #$24        Track 36 als Maximum
*1 a6c4 8d  ac  02   sta   $02ac       setzen
*0 a6c4 8d  d7  fe   sta   $fed7       (nur für 1571)
a6c7    20  96  a4   jsr   $a496       Pufferzeiger für BAM wieder holen
a6ca    8a           txa               Fehlermeldung wieder holen
a6cb    20  0a  e6   jsr   $e60a       und Fehlermeldung ausgeben; Ende
a6ce    4c  44  d6   jmp   $d644       ggf. durch BUMP neuen Leseversuch
a6d1    a0  03       ldy   #$03        Pufferzeiger auf Diskettenkennzeichen ($80)
a6d3    b1  6d       lda   ($6d),y     Kennzeichen holen
a6d5    2d  af  01   and   $01af       und mit erstem Leseversuch auf Fehler prüfen
a6d8    30  03       bmi   $a6dd       verzweige, wenn beides in Ordnung
a6da    a9  24       lda   #$24        sonst anzeigen ,daß Diskette nur einseitig
a6dc    2c           .byte $2c         nächsten Befehl überspringen
a6dd    a9  47       lda   #$47        anzeigen, daß Diskette zweiseitig beschrieben
*1 a6df 8d  ac  02   sta   $02ac       maximale Tracknummer setzen
*0 a6df 8d  d7  fe   sta   $fed7       (nur für 1571)
a6e2    4c  96  a4   jmp   $a496       Pufferzeiger für BAM wieder holen; Ende
-------------------------------------------------------------------------
a6e5                                   Diskette auf zweiseitige Formatierung untersuchen.
a6e5    20  8c  d5   jsr   $d58c       Jobcode an Diskcontroller und Job ausführen
a6e8    48           pha               Rückmeldung merken
a6e9    c9  02       cmp   #$02        Fehler aufgetreten?
a6eb    b0  49       bcs   $a736       verzweige, wenn ja
a6ed    ad  0f  18   lda   $180f       Steuerport lesen
a6f0    29  20       and   #$20        Floppy im 1541-Modus?
a6f2    f0  42       beq   $a736       verzweige, wenn ja
```

```
a6f4    a9  47       lda   #$47       sonst maximale Tracknummer für 1571 (71)
*1 a6f6 8d  ac  02    sta   $02ac      setzen
*0 a6f6 8d  d7  fe    sta   $fed7      (nur für 1571)
a6f9    a9  ff       lda   #$ff       Flag für Fehler ignorieren
a6fb    8d  98  02    sta   $0298      setzen
a6fe    a5  16       lda   $16        ID 1 aus Blockheader holen
a700    48           pha               und merken
a701    a5  17       lda   $17        ID 2 holen
a703    48           pha               und merken
a704    a5  f9       lda   $f9        Puffernummer
a706    0a           asl               mal 2
a707    aa           tax               als Index
a708    a9  35       lda   #$35       Track 53; entspricht 18 für Seite 1
a70a    95  06       sta   $06,x      in Jobspeicher
a70c    a9  b0       lda   #$b0       Jobcode für 'SEEK'
a70e    20  8c  d5    jsr   $d58c      übergeben und Job ausführen
a711    c9  02       cmp   #$02       Fehler?
a713    68           pla               ID 2 zurückholen
a714    a8           tay               und merken
a715    68           pla               ID 1 zurückholen
a716    aa           tax               und merken
a717    b0  0b       bcs   $a724      verzweige, wenn Fehler aufgetreten
a719    e4  16       cpx   $16        alte ID 1 mit neuer vergleichen
a71b    d0  07       bne   $a724      verzweige, wenn ungleich
a71d    c4  17       cpy   $17        alte ID 2 mit neuer vergleichen
a71f    d0  03       bne   $a724      verzweige, wenn ungleich
a721    a9  47       lda   #$47       maximale Tracknummer auf 71
a723    2c           .byte $2c        nächsten Befehl überspringen
a724    a9  24       lda   #$24       maximale Tracknummer auf 36
*1 a726 8d  ac  02    sta   $02ac      setzen
*0 a726 8d  d7  fe    sta   $fed7      (nur für 1571)
a729    84  17       sty   $17        neue ID 2 setzen
a72b    86  16       stx   $16        neue ID 1 setzen
a72d    a5  f9       lda   $f9        Puffernunrner
a72f    0a           asl               mal 2
a730    aa           tax               als Index
a731    ad  85  fe    lda   $fe85      Tracknummer für BAM (18)
a734    95  06       sta   $06,x      in Jobspeicher holen
a736    68           pla               Rückmeldung zurückholen
a737    60           rts               Ende
--------------------------------------------------------------------------
a738                                   BAM-Puffer für erweiterte 1571-BAM löschen.
a738    20  3a  ef    jsr   $ef3a      Pufferzeiger für BAM setzen
a73b    ad  0f  18    lda   $180f      Steuerport lesen
a73e    29  20       and   #$20       Floppy.im 1541-Mdus?
a740    f0  0a       beq   $a74c      verzweIge, wenn Ja
a742    a9  00       lda   #$00       Füllwert
a744    a0  68       ldy   #$68       Index in Puffer
a746    99  46  01    sta   $0146,y    Byte in Puffer schreiben
a749    88           dey               bis der gesamte Puffer
a74a    10  fa       bpl   $a746      geleert ist
a74c    4c  08  f0    jmp   $f008      BAM-Hauptpuffer löschen; Ende
--------------------------------------------------------------------------
a74f                                   Bei Tracknummern größer als 35 werden diese in die
                                       physikalischen Tracknummern kleiner als 36 umgewandelt.
a74f    48           pha               Tracknummer merken
a750    ad  0f  18    lda   $180f      Steuerregister lesen
```

```
a753    29  20       and   #$20        Floppy im 1541-Modus?
a755    f0  08       beq   $a75f       verzweige, wenn ja
a757    68           pla               Tracknummer zurückholen
a758    c9  24       cmp   #$24        und mit 36 vergleichen
a75a    90  04       bcc   $a760       verzweige, wenn kleiner
a75c    e9  23       sbc   #$23        sonst 35 abziehen
a75e    24           .byte $24         nächsten Befehl überspringen
a75f    68           pla               Tracknummer zurückholen
a760    ae  d6  fe   ldx   $fed6       Anzahl der Diskettenzonen holen
a763    60           rts               Ende
--------------------------------------------------------------------------
a764                                   neue BAM im Speicher herstellen.
a764    20  05  f0   jsr   $f005       BAM.Puffer löschen
a767    ad  0f  18   lda   $180f       Steuerregister lesen
a76a    29  20       and   #$20        Floppy im 1541-Modus?
a76c    d0  03       bne   $a771       verzweige, wenn nein
a76e    a9  24       lda   #$24        maximale Tracknummer 36
a770    2c           .byte $2c         nächsten Befehl überspringen
a771    a9  47       lda   #$47        maximale Tracknummer 71
*1 a773 8d  ac  02   sta   $02ac       setzen
*0 a773 8d  d7  fe   sta   $fed7       (nur für 1571)
a776    4c  43  ee   jmp   $ee43       neue BAM herstellen; Ende
--------------------------------------------------------------------------
a779                                   Formatieren einer Diskette im GCR-Format ohne Aufbringen
                                       des Directory.
a779    ad  0f  18   lda   $180f       Steuerregister lesen
a77c    29  20       and   #$20        Floppy im 1541.Modus?
a77e    d0  03       bne   $a783       verzweige, wenn nein
a780    4c  c6  c8   jmp   $c8c6       Diskette einseitig langsam formatieren; Ende
a783    4c  0d  a4   jmp   $a40d       Diskette zweiseitig schnell formatieren; Ende
--------------------------------------------------------------------------
a786                                   RESET für seriellen Bus.
a786    ad  01  18   lda   $1801       Datenrichtungsregister für
a789    29  df       and   #$df        den seriellen Bus
a78b    8d  01  18   sta   $1801       setzen
a78e    20  83  a4   jsr   $a483       ca. 70 Taktzyklen warten
a791    a9  7f       lda   #$7f        IRQ-Status
a793    8d  0d  40   sta   $400d       löschen; Maske setzen
a796    a9  08       lda   #$08        Timer auf 'one shot'
a798    8d  0e  40   sta   $400e       Timer A setzen
a79b    8d  0f  40   sta   $400f       Timer B setzen
a79e    a9  00       lda   #$00        Timer A Hi
a7a0    8d  05  40   sta   $4005       löschen
a7a3    a9  06       lda   #$06        Timer A Lo
a7a5    8d  04  40   sta   $4004       auf 6 Taktzyklen setzen (für Schieberegister)
a7a8    a9  01       lda   #$01        Timer A
a7aa    8d  0e  40   sta   $400e       starten
a7ad    20  b2  81   jsr   $81b2       seriellen Bus auf Eingang schalten
a7b0    4c  59  ea   jmp   $ea59       auf ATN-Signal prüfen; Ende
--------------------------------------------------------------------------
a7b3                                   ATN-Bedienung des seriellen Bus im 1571-Modus.
*1 a7b3 ad  0f  18   lda   $180f       Steuerregister lesen
*0 a7b3 20  62  aa   jsr   $aa62       Steuerregister lesen
a7b6    29  20       and   #$20        Floppy im 1541.Modus?
a7b8    f0  03       beq   $a7bd       verzweige, wenn ja
a7ba    4c  ce  80   jmp   $80ce       sonst Kommando vom 1571-Bus übernehmen
a7bd    4c  5b  e8   jmp   $e85b       Kommando vom 1541-Bus übernehmen
```

```
------------------------------------------------------------------------
a7c0                                    Fortsetzung der RESET-Routine.
a7c0    78          sei                 Diskcontroller inaktivieren
a7c1    a2  45      ldx   #$45          Stackpointer
a7c3    9a          txs                 setzen
a7c4    4c  25  eb  jmp   $eb25         und weiter in RESET-Routine
------------------------------------------------------------------------
a7c7                                    Neue BAM im Speicher erzeugen. Erweiterung für zweiseitigen
                                        Diskettenbetrieb.
*1 a7c7 ad  0f  18  lda   $180f         Steuerregister lesen
*0 a7c7 4c  ce  a7  jmp   $a7ce         immer 1541-BAM erzeugen
a7ca    29  20      and   #$20          Floppy im 1541-Modus?
a7cc    d0  09      bne   $a7d7         verzweige, wenn nein
a7ce    a0  03      ldy   #$03          Pufferzeiger auf Diskettenkennzeichen
a7d0    a9  00      lda   #$00          Kennzeichen für 1541-Diskette
a7d2    91  6d      sta   ($6d),y       in Puffer schreiben
a7d4    4c  b7  ee  jmp   $eeb7         BAM für 1541 erzeugen; Ende
*1 a7d7 ad  ac  02  lda   $02ac         maximale Tracknummer holen
*0 a7d7 ad  d7  fe  lda   $fed7         maximale Tracknummer (36) holen
a7da    c9  25      cmp   #$25          mit 37 vergleichen (Fehler: eigentlich 36)
a7dc    90  f0      bcc   $a7ce         verzweige, wenn kleiner
a7de    a0  01      ldy   #$01          Nummer des ersten Tracks
a7e0    a2  00      ldx   #$00          Nummer des ersten Sektors
a7e2    c0  12      cpy   #$12          Track 18 erreicht?
a7e4    f0  34      beq   $a81a         verzweige, wenn ja
a7e6    8a          txa                 Sektornummer
a7e7    48          pha                 merken
a7e8    a9  00      lda   #$00          Register für
a7ea    85  6f      sta   $6f          Bitmuster
a7ec    85  70      sta   $70          eines Tracks
a7ee    85  71      sta   $71          löschen
a7f0    b9  2b  94  lda   $942b,y       Anzahl der Sektoren des Tracks holen
a7f3    aa          tax                 und merken
a7f4    38          sec                 Zustand für Sektor frei
a7f5    26  6f      rol   $6f          in das
a7f7    26  70      rol   $70          Bitmuster der BAM
a7f9    26  71      rol   $71          eintragen
a7fb    ca          dex                 nächsten Sektor
a7fc    d0  f6      bne   $a7f4         ebenso behandeln
a7fe    68          pla                 Sektornummer zurückholen
a7ff    aa          tax                 und wieder merken
a800    a5  6f      lda   $6f          erstes Byte des Bitmusters
a802    9d  46  01  sta   $0146,x      in BAM-Puffer schreiben
a805    a5  70      lda   $70          zweites Byte
a807    9d  47  01  sta   $0147,x      in BAM-Puffer schreiben
a80a    a5  71      lda   $71          drittes Byte des Bitmusters
a80c    9d  48  01  sta   $0148,x      ebenfalls in BAM-Puffer schreiben
a80f    e8          inx                 Pufferzeiger
a810    e8          inx                 plus 3, um
a811    e8          inx                 auf den Start des nächsten Bitmusters zu kommen
a812    e0  33      cpx   #$33          schon Eintrag für Track 18 erreicht?
a814    d0  04      bne   $a81a         verzweige, wenn nein
a816    e8          inx                 sonst Track 18
a817    e8          inx                 mit dem Pufferzeiger
a818    e8          inx                 überspringen
a819    c8          iny                 ebenso die
a81a    c8          iny                 Tracknummer auslassen
a81b    c0  24      cpy   #$24          schon 36 erreicht?
```

```
a81d   90  c3       bcc   $a7e2       weitermachen, wenn nein
a81f   20  b7  ee    jsr   $eeb7       BAM für Diskseite 0 ebenfalls anlegen
a822   a0  03       ldy   #$03        Pufferzeiger auf Diskettenkennzeichen
a824   a9  80       lda   #$80        Kennzeichen für 1571-Format
a826   91  6d       sta   ($6d),y     in Puffer schreiben
a828   a0  ff       ldy   #$ff        Pufferzeiger
a82a   a2  22       ldx   #$22        setzen
a82c   bd  2c  94    lda   $942c,x     und Anzahl der freien Blöcke jedes Tracks
a82f   91  6d       sta   ($6d),y     in den Puffer schreiben
a831   88           dey               Pufferzeiger
a832   ca           dex               vermindern
a833   10  f7       bpl   $a82c       und weitermachen
a835   a0  ee       ldy   #$ee        Pufferzeiger auf Track 18; Seite 1
a837   a9  00       lda   #$00        Anzahl der freien Blöcke gleich 0
a839   91  6d       sta   ($6d),y     setzen
a83b   4c  75  d0    jmp   $d075       Anzahl der freien Blöcke berechnen; Ende
--------------------------------------------------------------------------
a83e                                   Einen Block in der BAM freigeben.
a83e   ad  0f  18    lda   $180f       Steuerregister lesen
a841   29  20       and   #$20        Floppy im 1541-Modus?
a843   d0  06       bne   $a84b       verzweige, wenn nein
a845   20  cf  ef    jsr   $efcf       Zeiger in Bitmuster für Block holen
a848   4c  62  ef    jmp   $ef62       Block freigeben; Ende
a84b   a5  80       lda   $80         aktuelle Tracknummer
a84d   c9  24       cmp   #$24        kleiner als 36?
a84f   90  f4       bcc   $a845       verzweige, wenn ja
a851   20  a1  a4    jsr   $a4a1       Zeiger in Bitmuster für Block holen
a854   20  e7  a4    jsr   $a4e7       zuständiges Bit aus BAM holen
a857   d0  19       bne   $a872       verzweige, wenn Sektor frei
a859   1d  e9  ef    ora   $efe9,x     sonst Block freigeben
a85c   99  46  01    sta   $0146,y     und Bitmuster wieder eintragen
a85f   20  88  ef    jsr   $ef88       BAM 'dirty' Flag setzen
a862   20  08  a5    jsr   $a508       Anzahl der freien Blöcke plus 1
a865   a5  80       lda   $80         aktuelle Tracknummer
a867   c9  35       cmp   #$35        gleich 53; Track 18 von Seite 1
a869   f0  08       beq   $a873       verzweige, wenn ja
a86b   a5  7f       lda   $7f         Drivenummer
a86d   0a           asl               mal 2
a86e   aa           tax               als Index
a86f   4c  7f  ef    jmp   $ef7f       Anzahl der freien Blöcke erhöhen; Ende
a872   38           sec               Flag für Fehler; Block schon frei
a873   60           rts               Ende
--------------------------------------------------------------------------
a874                                   Block in der BAM belegen.
a874   ad  0f  18    lda   $180f       Steuerregister Lesen
a877   29  20       and   #$20        Floppy im 1541-Modus1
a879   d0  06       bne   $a881       verzweige, wenn nein
a87b   20  cf  ef    jsr   $efcf       Zeiger auf Bitmuster für Block holen
a87e   4c  96  ef    jmp   $ef96       und Block belegen; Ende
a881   a5  80       lda   $80         aktuelle Tracknummer
a883   c9  24       cmp   #$24        kleiner als 361
a885   90  f4       bcc   $a87b       verzweige, wenn ja
a887   20  a1  a4    jsr   $a4a1       Pufferzeiger auf Bitmuster holen
a88a   20  e7  a4    jsr   $a4e7       und Bit des Blocks holen
a88d   f0  19       beq   $a8a8       verzweige, wenn Block schon belegt
a88f   5d  e9  ef    eor   $efe9,x     sonst Block belegen
a892   99  46  01    sta   $0146,y     und in BAM eintragen
a895   20  88  ef    jsr   $ef88       BAM 'dirty' Flag setzen
```

```
a898   20  1e  a5    jsr   $a51e       Anzahl der freien Blöcke vermindern
a89b   a5  80        lda   $80         aktuelle Tracknummer
a89d   c9  35        cmp   #$35        gleich_53; Track.18 von Seite 1?
a89f   f0  07        beq   $a8a8       verzweIge, wenn ja
a8a1   a5  7f        lda   $7f         Drivenummr
a8a3   0a            asl               mal 2
a8a4   aa            tax               als Index
a8a5   4c  b2  ef    jmp   $efb2       Anzahl der freien Blöcke minus 1; Ende
a8a8   60            rts               Ende; Fehler: Block war schon belegt
--------------------------------------------------------------------------
a8a9                                   Nächsten freien Sektor eines Tracks suchen.
a8a9   ad  0f  18    lda   $180f       Steuerregister lesen
a8ac   29  20        and   #$20        Floppy.im 1541-Modus?
a8ae   d0  06        bne   $a8b6       verzweige, wenn nein
a8b0   20  11  f0    jsr   $f011       Pufferzeiger für BAM setzen
a8b3   4c  fd  f1    jmp   $f1fd       freien Block suchen; Ende
a8b6   a5  80        lda   $80         aktuelle Tracknummer
a8b8   c9  24        cmp   #$24        kleiner 36?
a8ba   90  f4        bcc   $a8b0       verzweige, wenn ja
a8bc   20  a1  a4    jsr   $a4a1       Zeiger auf BAM-Eintrag des Tracks setzen
a8bf   20  45  a5    jsr   $a545       auf freien Block prüfen
a8c2   b9  2c  94    lda   $942c,y     Anzahl der Sektoren des Tracks
a8c5   8d  4e  02    sta   $024e       übernehmen
a8c8   a5  81        lda   $81         Sektornummer
a8ca   cd  4e  02    cmp   $024e       mit Maximum vergleichen
a8cd   b0  09        bcs   $a8d8       verzweige, wenn größer
a8cf   20  e7  a4    jsr   $a4e7       Bit des Sektors in der BAM holen
a8d2   d0  06        bne   $a8da       verzweige, wenn Sektor frei
a8d4   e6  81        inc   $81         sonst nächste Sektornummer
a8d6   d0  f0        bne   $a8c8       und weitersuchen
a8d8   a9  00        lda   #$00        Flag für alles belegt
a8da   60            rts               setzen; Ende
--------------------------------------------------------------------------
a8db                                   Erweiterung zum Suchen eines freien Sektors für den
                                       doppelseitigen Diskettenbetrieb.
a8db   ad  0f  18    lda   $180f       Steuerregister lesen
a8de   29  20        and   #$20        Floppy.im 1541-Mods?
a8e0   d0  06        bne   $a8e8       verzweIge, wenn neIn
a8e2   a5  6f        lda   $6f         Anzahl der freien Blöcke des Tracks
a8e4   48            pha               merken
a8e5   4c  30  f1    jmp   $f130       zurück zur Behandlung im 1541-Modus
a8e8   a5  80        lda   $80         aktuelle Tracknummer
a8ea   c9  24        cmp   #$24        kleiner 36?
a8ec   90  f4        bcc   $a8e2       verzweige, wenn ja
a8ee   c9  35        cmp   #$35        mit 53; Track 18 für Seite 1 vergleichen
a8f0   f0  0e        beq   $a900       verzweige, wenn gleich
a8f2   a5  6f        lda   $6f         Anzahl der freien Blöcke des Tracks
a8f4   48            pha               merken
a8f5   20  a1  a4    jsr   $a4a1       Zeiger in BAM für Bitmuster setzen
a8f8   a8            tay               Anzahl der freien Blöcke merken
a8f9   68            pla               Zwischenspeicher
a8fa   85  6f        sta   $6f         wieder setzen
a8fc   98            tya               Anzahl der freien Blöcke
a8fd   4c  38  f1    jmp   $f138       und nächsten freien Block holen
a900   a9  00        lda   #$00        Suche auf Nachbarspur generieren und
a902   4c  38  f1    jmp   $f138       dort nächsten freien Block holen; Ende
```

```
--------------------------------------------------------------------------
a905                                    Bitmuster für Block aus BAM holen
a905    ad  0f  18    lda   $180f       Steuerport lesen
a908    29  20        and   #$20        Floppy im 1541-Modus?
a90a    d0  06        bne   $a912       verzweige, wenn nein
a90c    20  11  f0    jsr   $f011       Zeiger in BAM setzen
a90f    4c  c7  f1    jmp   $f1c7       Bitmuster holen; Ende
a912    a5  80        lda   $80         Tracknummer
a914    c9  24        cmp   #$24        kleiner 36?
a916    90  f4        bcc   $a90c       verzweige, wenn ja
a918    20  a1  a4    jsr   $a4a1       Zeiger in BAM setzen
a91b    4c  c9  f1    jmp   $f1c9       Bitmuster holen; Ende
--------------------------------------------------------------------------
a91e                                    Nächsten freien Block holen.
a91e    ad  0f  18    lda   $180f       Steuerport lesen
a921    29  20        and   #$20        Floppy im 1541-Modus?
a923    d0  06        bne   $a92b       verzweige, wenn nein
a925    20  11  f0    jsr   $f011       Zeiger in Bitmuster setzen
a928    4c  e2  f1    jmp   $f1e2       und freien Block holen; Ence
a92b    a5  80        lda   $80         Tracknummer
a92d    c9  24        cmp   #$24        kleiner 36?
a92f    90  f4        bcc   $a925       verzweige, wenn ja
a931    20  a1  a4    jsr   $a4a1       Zeiger in Bitmuster der BAM setzen
a934    4c  e4  f1    jmp   $f1e4       freien Block holen; Ende
--------------------------------------------------------------------------
a937                                    Gültigkeit der Angaben in der BAM prüfen.
a937    ad  0f  18    lda   $180f       Steuerport lesen
a93a    29  20        and   #$20        Floppy im 1541-Modus?
a93c    d0  03        bne   $a941       verzweige, wenn nein
a93e    4c  20  f2    jmp   $f220       Blockangaben überprüfen; Ende
*1 a941 ad  ac  02    lda   $02ac       maximale Tracknummer holen
*0 a941 ad  d7  fe    lda   $fed7       maximale Tracknummer (36) holen
a944    c9  25        cmp   #$25        kleiner als 37?
a946    90  f6        bcc   $a93e       verzweige, wenn ja
a948    a5  80        lda   $80         aktuelle Tracknummer
a94a    c9  24        cmp   #$24        kleiner als 36?
a94c    90  f0        bcc   $a93e       verzweige, wenn ja
a94e    4c  45  a5    jmp   $a545       Angaben in der BAM prüfen; Ende
--------------------------------------------------------------------------
a951                                    Anzahl der freien Blöcke auf der Diskette berechnen.
                                        Erweiterung für doppelseitige Disketten.
a951    9d  fa  02    sta   $02fa,x     freie Blöcke Lo setzen
a954    ad  0f  18    lda   $180f       Steuerport lesen
a957    29  20        and   #$20        Floppy im 1541-Modus?
a959    f0  23        beq   $a97e       verzweige, wenn ja
a95b    ad  d7  fe    lda   $fed7       maximale Tracknummer holen
a95e    c9  25        cmp   #$25        kleiner als 37?
a960    90  1c        bcc   $a97e       verzweige, wenn ja
a962    20  8b  a4    jsr   $a48b       Pufferzeiger für BAM merken
a965    20  34  a5    jsr   $a534       Pufferzeiger auf Eintrag für Track setzen
a968    a0  22        ldy   #$22        Zähler für Trackanzahl
a96a    ad  fa  02    lda   $02fa       Anzahl der freien Blöcke Lo
a96d    18            clc               zu
a96e    71  6d        adc   ($6d),y     der Anzahl des aktuellen Tracks addieren
a970    8d  fa  02    sta   $02fa       und merken
a973    90  03        bcc   $a978       verzweige, wenn kein Übertrag
a975    ee  fc  02    inc   $02fc       sonst Anzahl der freien Blöcke Hi erhöhen
a978    88            dey               Zähler für Tracks vermindern
```

```
a979   10  f2       bpl   $a96d       und weitermachen, bis alle Tracks addiert
a97b   4c  96  a4    jmp   $a496       Pufferzeiger für BAM wieder holen; Ende
a97e   60           rts               Ende
--------------------------------------------------------------------------
a97f                                  Anzahl der Blöcke einer Datei löschen.
a97f   95  b5       sta   $b5,x       Anzahl der Blöcke Lo löschen
a981   95  bb       sta   $bb,x       Anzahl der Blöcke Hi löschen
a983   a9  00       lda   #$00        Endekennzeichen des Puffers
a985   9d  44  02    sta   $0244,x     $0244,x löschen
a988   60           rts               Ende
--------------------------------------------------------------------------
a989                                  Diskette im 1571-Format formatieren.
a989   20  0d  a4    jsr   $a40d       Diskette formatieren; ohne Directory
a98c   a0  00       ldy   #$00        Flag für Fehler ignorieren
a98e   8c  98  02    sty   $0298       löschen
a991   60           rts               Ende
--------------------------------------------------------------------------
a992 ff ...                           unbenutzter
a99c ... ff                           Leerbereich
--------------------------------------------------------------------------
a99d                                  Drivestatus für ok setzen.
a99d   a9  00       lda   #$00        Code für OK
a99f   9d  ff  00    sta   $00ff,x     als Drivestatus setzen
a9a2   4c  b7  c1    jmp   $c1b7       zurück
--------------------------------------------------------------------------
a9a5                                  Drivestatus anhand Y-Register setzen.
a9a5   98           tya               Status übernehmen
a9a6   9d  ff  00    sta   $00ff,x     und setzen
a9a9   4c  64  c6    jmp   $c664       zurück
--------------------------------------------------------------------------
a9ac                                  Bedienung des seriellen Bus nach Auftreten eines ATN.
a9ac   ad  0f  18    lda   $180f       Steuerport lesen
a9af   29  20       and   #$20        Floppy im 1541-Modus?
a9b1   f0  03       beq   $a9b6       verzweige, wenn ja
a9b3   4c  5a  81    jmp   $815a       Busbedienung für 1571; Ende
a9b6   4c  d7  e8    jmp   $e8d7       Busbedienung für 1541; Ende
--------------------------------------------------------------------------
a9b9                                  Erweiterte Fehlerbehandlung für 1571 mit Burst-Statusbyte
                                      und Klartext.
a9b9   48           pha               Fehlernummer merken
a9ba   86  f9       stx   $f9         Puffernummer setzen
a9bc   ad  0f  18    lda   $180f       Steuerport lesen
a9bf   29  20       and   #$20        Floppy im 1541-Modus?
a9c1   f0  0f       beq   $a9d2       verzweig wenn ja
a9c3   24  37       bit   $37         Flags für Busbetrieb
a9c5   10  0b       bpl   $a9d2       verzweige, wenn langsamer Busbetrieb
a9c7   a5  37       lda   $37         Flags für Busbetrieb
a9c9   29  7f       and   #$7f        Bus auf langsamen Betrieb
a9cb   85  37       sta   $37         umschalten
a9cd   68           pla               Fehlernummer zurückholen
a9ce   aa           tax               und setzen
a9cf   4c  99  91    jmp   $9199       Burst-Statusbyte zum Computer; Ende
a9d2   4c  0d  e6    jmp   $e60d       Klartextmeldung ausgeben; Ende
```

```
------------------------------------------------------------------------
a9d5                                    Fehlerbehandlung für 1541/1571 bei LOAD.
a9d5   48           pha                 Fehlernummer merken
a9d6   ad  0f  18    lda   $180f        Steuerport lesen
a9d9   29  20        and   #$20         Floppy im 1541-Modus?
a9db   f0  17        beq   $a9f4        verzweige, wenn ja
a9dd   24  37        bit   $37          Flags für Busbetrieb
a9df   10  13        bpl   $a9f4        verzweige, wenn langsamer Busbetrieb
a9e1   a5  37        lda   $37          sonst Flags für Busbetrieb
a9e3   29  7f        and   #$7f         auf langsamen Busbetrieb
a9e5   85  37        sta   $37          umschalten
a9e7   78            sei                Diskcontroller inaktivieren
a9e8   a2  02        ldx   #$02         Nummer für 'FILE NOT FOUND'
a9ea   20  28  92    jsr   $9228        Burst-Statusbyte an Computer; Ende
a9ed   a9  00        lda   #$00         Sekundäradresse für LOAD
a9ef   85  83        sta   $83          setzen
a9f1   20  c0  da    jsr   $dac0        File schließen
a9f4   68            pla                Fehlernummer zurückholen
a9f5   4c  45  e6    jmp   $e645        Fehlermeldung ausgeben; Ende
------------------------------------------------------------------------
a9f8                                    Drivestatus auf inaktiv stellen
a9f8   a9  00        lda   #$00         Status auf Drive inaktiv
a9fa   85  20        sta   $20          stellen
a9fc   ad  0c  1c    lda   $1c0c        Steuerregister holen
a9ff   4c  66  f2    jmp   $f266        und zurück
------------------------------------------------------------------------
aa02                                    Behandlung des neuen U0-Befehls für 1571.
aa02   ad  00  02    lda   $0200        erstes Byte aus Befehlsstring
aa05   c9  55        cmp   #$55         mit 'U' vergleichen
aa07   d0  07        bne   $aa10        verzweige, wenn ungleich
aa09   ad  01  02    lda   $0201        zweites Zeichen aus Befehlsstring
aa0c   c9  30        cmp   #$30         mit '0' vergleichen
aa0e   f0  04        beq   $aa14        verzweige, wenn gleich
aa10   b9  00  02    lda   $0200,y      aktuelles Zeichen aus Befehlsstring
aa13   2c            .byte $2c          nächsten Befehl überspringen
aa14   a9  00        lda   #$00         Wert 0 als Flag für U0-Kommando
aa16   60            rts                Ende
------------------------------------------------------------------------
aa17                                    aktuellen Drivestatus holen.
aa17   a6  7f        ldx   $7f          Drivenummer
aa19   bd  ff  00    lda   $00ff,x      Flags für Drivestatus holen
aa1c   60            rts                Ende
------------------------------------------------------------------------
aa1d                                    Flags für Drivestatus setzen.
aa1d   95  1c        sta   $1c,x        Flag für Diskettenwechsel setzen
aa1f   9d  ff  00    sta   $00ff,x      Flags für Drivestatus setzen
aa22   4c  75  d0    jmp   $d075        zurück
------------------------------------------------------------------------
aa25                                    Flags für Drivestatus holen.
aa25   a6  7f        ldx   $7f          Drenummer
aa27   bd  ff  00    lda   $00ff,x      Flags für Drivestatus holen
aa2a   4c  1b  f0    jmp   $f01b        zurück
------------------------------------------------------------------------
aa2d                                    USER-Befehle ausführen (außer U0).
aa2d   a5  75        lda   $75          Adresse Lo der USER-Routine
aa2f   c9  67        cmp   #$67         auf Jobschleife prüfen
aa31   d0  09        bne   $aa3c        verzweige, wenn nein
aa33   a5  76        lda   $76          Adresse Hi der USER-Routine
```

```
aa35    c9  fe        cmp   #$fe        gleich Adresse der IRQ-Routine?
aa37    d0  03        bne   $aa3c       verzweige, wenn nein
aa39    00            brk               sonst Jobschleife aufrufen
aa3a    ea            nop               Rückkehr nach $aa3b
aa3b    60            rts               Ende
aa3c    6c  75  00    jmp   ($0075)     Sprung auf USER-Routine; Ende
--------------------------------------------------------------------------
*1 aa3f ff ...                          unbenutzter
*1 aa68 ... ff                          Leerbereich
--------------------------------------------------------------------------
*0 aa3f                                 Rückmeldung nach Job prüfen.
*0 aa3f c9  02        cmp   #$02        Fehler?
*0 aa41 90  07        bcc   $aa4a       verzweige, wenn nein
*0 aa43 c9  0f        cmp   #$0f        'DRIVE NOT READY'?
*0 aa45 f0  03        beq   $aa4a       verzweige, wenn ja
*0 aa47 4c  6b  d3    jmp   $d36b       zurück
*0 aa4a 4c  73  d3    jmp   $d373       Fehler ausgeben; Ende
--------------------------------------------------------------------------
*0 aa4d                                 Diskette im 1570-Format formatieren.
*0 aa4d 85  51        sta   $51         Tracknummer für Formatierung setzen
*0 aa4f 20  7c  87    jsr   $877c       lED am laufwerk einschalten
*0 aa52 20  89  a9    jsr   $a989       Diskette formatieren
*0 aa55 48            pha               Rückmeldung merken
*0 aa56 20  88  87    jsr   $8788       lED am laufwerk ausschalten
*0 aa59 68            pla               Rückmeldung zurückholen
*0 aa5a 60            rts               Ende
--------------------------------------------------------------------------
*0 aa5b                                 Filetyp auf Programmfile (PRG) testen.
*0 aa5b a5  e7        lda   $e7         Filetyp
*0 aa5d 29  07        and   #$07        reinen Filetyp isolieren
*0 aa5f c9  02        cmp   #$02        und mit 2 (PRG) vergleichen
*0 aa61 60            rts               Ende
--------------------------------------------------------------------------
*0 aa62                                 Steuerport lesen.
*0 aa62 ad  0f  18    lda   $180f       Steuerport lesen
*0 aa65 2c  01  18    bit   $1801       und seriellen Bus initialisieren
*0 aa68 60            rts               Ende
--------------------------------------------------------------------------
aa69 ff ...                             unbenutzter
beff ... ff                             Leerbereich
--------------------------------------------------------------------------
bf00                                    Systemsprungvektoren (werden nicht benutzt).
bf00    4c  88  9d    jmp   $9d88       IRQ-Routine für 1541
bf03    4c  de  9d    jmp   $9dde       IRQ-Routine für 1571
bf06    4c  b0  f2    jmp   $f2b0       Jobschleife für 1541
bf09    4c  ba  92    jmp   $92ba       Jobschleife für 1571
bf0c    4c  93  f3    jmp   $f393       Pufferzeiger setzen für 1541
bf0f    4c  d1  93    jmp   $93d1       Pufferzeiger setzen für 1571
bf12    4c  69  f9    jmp   $f969       Ausgang der Jobschleife für 1541
bf15    4c  b5  99    jmp   $99b5       Ausgang der Jobschleife für 1571
bf18    4c  00  fe    jmp   $fe00       auf Lesen umschalten
bf1b    4c  34  f9    jmp   $f934       Blockheader in GCR umwandeln
bf1e    4c  56  f5    jmp   $f556       SYNC-Signal abwarten für 1541
bf21    4c  54  97    jmp   $9754       SYNC-Signal abwarten für 1571
bf24    4c  e0  f8    jmp   $f8e0       Ausweichpuffer von GCR nach Binär für 1541
bf27    4c  65  99    jmp   $9965       Ausweichpuffer von GCR nach Binär für 1571
bf2a    4c  e9  f5    jmp   $f5e9       Prüfsumme über Datenblock berechnen
bf2d    4c  e6  f7    jmp   $f7e6       5 GCR-Bytes in 4 Binärwerte für 1541
```

```
bf30   4c  d9  98   jmp   $98d9      5 gcr-Bytes in 4 Binärwerte für 1571
bf33   4c  83  a4   jmp   $a483      Verzögerung von ca. 70 Taktzyklen
bf36   4c  f3  fe   jmp   $fef3      Verzögerung für seriellen Bus für 1541
bf39   4c  7e  a4   jmp   $a47e      Verzögerung für seriellen Bus für 1571
bf3c   4c  05  f0   jmp   $f005      BAM-Puffer löschen
bf3f   4c  d1  f0   jmp   $f0d1      Tracknummer für BAM holen
bf42   4c  46  c1   jmp   $c146      Befehl analysieren und ausführen
bf45   4c  68  c2   jmp   $c268      Befehlsstring durchsuchen
bf48   4c  b3  c2   jmp   $c2b3      Tabellen und Zeiger initialisieren
bf4b   4c  dc  c2   jmp   $c2dc      Zeiger für Dateien löschen
bf4e   4c  e6  86   jmp   $86e6      zentrale MFM-Steuerroutine
bf51   4c  64  87   jmp   $8764      Drivemotor anschalten
bf54   4c  70  87   jmp   $8770      Drivemotor ausschalten
bf57   4c  8e  80   jmp   $808e      (zeigt nicht auf Routine!!!)
bf5a   4c  1e  cf   jmp   $cf1e      Puffer für Diskbetrieb setzen
bf5d   4c  b4  d7   jmp   $d7b4      ON-Befehl von seriellem Bus
bf60   4c  c0  da   jmp   $dac0      CLOSE-Befehl
bf63   4c  0a  e6   jmp   $e60a      Fehlerbehandlung nach Befehl
bf66   4c  80  90   jmp   $9080      FAST-LOAD-UTILITY; File schnell laden
bf69   4c  4e  92   jmp   $924e      Prüfsumme über das ROM bilden
bf6c   4c  59  f2   jmp   $f259      Diskcontroller initialisieren für 1541
bf6f   4c  9c  f9   jmp   $f99c      Jobschleife für Drivemotoren für 1541
bf72   4c  ca  99   jmp   $99ca      Jobschleife für Drivemotoren für 1571
bf75   4c  95  fe   jmp   $fe95      (zeigt nicht auf Routine!!!)
-------------------------------------------------------------------------
bf78 ff ...                          unbenutzter
c0ff ... ff                          Leerbereich
-------------------------------------------------------------------------
  Ab hier ist das DOS 2.6 der 1541 fast vollständig erhalten worden!!!
-------------------------------------------------------------------------
```

```
------------------------------------------------------------------------
c100                                    Routine schaltet die LED am aktuellen Laufwerk ein und am
                                        anderen Laufwerk (nicht vorhanden!) aus.
c100   78          sei                 Diskcontroller inaktivieren
c101   a9  f7       lda   #$f7          Bit 3 des Diskcontrollers (LED für Drive 0)
c103   2d  00  1c    and   $1c00         löschen
c106   48           pha                 Bitwert merken
c107   a5  7f       lda   $7f           aktuelles Laufwerk (immer 0)
c109   f0  05       beq   $c110         unbedingter Sprung
c10b   68           pla                 LED-Wert zurückholen
c10c   09  00       ora   #$00          bei Laufwerk 0 keine Funktion
c10e   d0  03       bne   $c113         unbedingter Sprung
c110   68           pla                 LED-Wert zurückholen
c111   09  08       ora   #$08          LED-Bit setzen (Bit 3)
c113   8d  00  1c    sta   $1c00         LED an
c116   58           cli                 Diskcontroller wieder aktivieren
c117   60           rts                 Ende
------------------------------------------------------------------------
c118                                    Routine schaltet LED an Laufwerk 0 durch Setzen von Bit 3
                                        ein.
c118   78          sei                 Diskcontroller inaktivieren
c119   a9  08       lda   #$08          LED-Bit für Laufwerk 0
c11b   0d  00  1c    ora   $1c00         setzen
c11e   8d  00  1c    sta   $1c00         LED einschalten
c121   58           cli                 Diskcontroller wieder aktivieren
c122   60           rts                 Ende
------------------------------------------------------------------------
c123                                    Löschen der Flags des Fehlerstatus; dadurch wird der
                                        Fehlerzustand der Floppy gelöscht. Die LED wird beim
                                        nächsten Jobschleifendurchlauf ausgeschaltet.
c123   a9  00       lda   #$00          Flags löschen:
c125   8d  6c  02    sta   $026c         Fehlernummer = 0
c128   8d  6d  02    sta   $026d         Flag für LED-Blinken = 0
c12b   60           rts                 Ende
------------------------------------------------------------------------
c12c                                    Initialisierung des LED-Blinkens bei Auftreten eines
                                        Fehlers. Aktiviert werden dabei die Fehlerflags des
                                        Diskcontrollers, um der Jobschleife mitzuteilen, daß ein
                                        Fehler aufgetreten ist.
c12c   78          sei                 Diskcontroller inaktivieren
c12d   8a           txa                 X-Registerinhalt
c12e   48           pha                 retten
c12f   a9  50       lda   #$50          Fehlernummer
c131   8d  6c  02    sta   $026c         setzen
c134   a2  00       ldx   #$00          Index
c136   bd  ca  fe    lda   $feca,x       Defaultwert für LED-Bit ($08)
c139   8d  6d  02    sta   $026d         Blinkflag der LED setzen
c13c   0d  00  1c    ora   $1c00         LED-Bit setzen
c13f   8d  00  1c    sta   $1c00         LED einschalten
c142   68           pla                 X-Wert
c143   aa           tax                 zurückholen
c144   58           cli                 Diskcontroller wieder aktivieren
c145   60           rts                 Ende
```

```
------------------------------------------------------------------------
c146                                    Wertet den Befehlsstring vom Computer aus und springt ggf.
                                        zu den entsprechenden Routinen. Ansonsten wird eine
                                        Fehlermeldung ausgegeben. Eine entsprechende vorherige
                                        Fehleranzeige wird gelöscht.
c146    a9  00      lda   #$00         Flag für 'BAM nicht auf Diskette schreiben'
c148    8d  f9  02  sta   $02f9        löschen
c14b    ad  8e  02  lda   $028e        Standard Laufwerksnummer
c14e    85  7f      sta   $7f          als aktuelles Laufwerk speichern
c150    20  bc  e6  jsr   $e6bc        'OK'-Meldung bereitstellen
c153    a5  84      lda   $84          Sekundäradresse
c155    10  09      bpl   $c160        verzweige, wenn Befehl anliegt
c157    29  0f      and   #$0f         Sekundäradresse auf 15 begrenzen
c159    c9  0f      cmp   #$0f         Kommandokanal (15)?
c15b    f0  03      beq   $c160        verzweige, wenn ja
c15d    4c  b4  d7  jmp   $d7b4        OPEN-Routine, da Kanalnummer ungleich 15
c160    20  b3  c2  jsr   $c2b3        Flags für Befehsübernahme setzen
c163    b1  a3      lda   ($a3),y      Zeichen ab $0200 holen
c165    8d  75  02  sta   $0275        und abspeichern
c168    a2  0b      ldx   #$0b         in der Tabelle der Befehlsworte
c16a    bd  89  fe  lda   $fe89,x      nach Kommando suchen
c16d    cd  75  02  cmp   $0275        und vergleichen
c170    f0  08      beq   $c17a        verzweige, wenn gefunden
c172    ca          dex                Zeiger auf nächstes Kommando
c173    10  f5      bpl   $c16a        und weitersuchen
c175    a9  31      lda   #$31         Nummer der Fehlermeldung
c177    4c  c8  c1  jmp   $c1c8        '31, SYNTAX ERROR' ausgeben
c17a    8e  2a  02  stx   $022a        Nummer des Kommandos abspeichern
c17d    e0  09      cpx   #$09         Test auf Kommando mit Dateinamen
c17f    90  03      bcc   $c184        im Befehlsstring
c181    20  ee  c1  jsr   $c1ee        wenn ja, Befehlsstring prüfen
c184    ae  2a  02  ldx   $022a        Nummer des Befehls holen und
c187    bd  95  fe  lda   $fe95,x      dessen Sprungadresse nehmen
c18a    85  6f      sta   $6f          Adresse Lo speichern
c18c    bd  a1  fe  lda   $fea1,x      Adresse Hi
c18f    85  70      sta   $70          ebenfalls speichern
c191    6c  6f  00  jmp   ($006f)      Sprung auf Befehl; Ende
------------------------------------------------------------------------
c194                                    Abschluß eines Befehls. Stellt die 'OK'-Meldung bereit,
                                        sofern das Fehlerflag in $026c = 0 ist. Ansonsten erfolgt
                                        der Aufruf der Fehlerroutine.
c194    a9  00      lda   #$00         Flag für 'BAM nicht auf Diskette schreiben'
c196    8d  f9  02  sta   $02f9        löschen
c199    ad  6c  02  lda   $026c        Fehlerflag prüfen und
c19c    d0  2a      bne   $c1c8        zur Fehlerbehandlung, wenn ungleich Null
c19e    a0  00      ldy   #$00         Flags löschen:
c1a0    98          tya                Fehlernummer löschen
c1a1    84  80      sty   $80          Spurnummer löschen
c1a3    84  81      sty   $81          Sektornummer löschen
c1a5    84  a3      sty   $a3          Zeiger in INPUT-Puffer löschen
c1a7    20  c7  e6  jsr   $e6c7        'OK'-Parameter setzen
c1aa    20  23  c1  jsr   $c123        Fehlerflags löschen
c1ad    a5  7f      lda   $7f          Standardwert für Laufwerk neu
c1af    8d  8e  02  sta   $028e        setzen ..
c1b2    aa          tax                Flag für 'Laufwerk aktiv'
c1b3    4c  9d  a9  jmp   $a99d        löschen
c1b6    ea          nop                (Rest aus 1541-ROM)
```

```
c1b7   20  bd  c1    jsr   $c1bd       INPUT-Puffer löschen ($0200-0228)
c1ba   4c  da  d4    jmp   $d4da       interne Kanäle löschen
-------------------------------------------------------------------------
c1bd                                   löscht den INPUT-Puffer von $0200 bis $0228 durch
                                       Überschreiben mit 0.
c1bd   a0  28        ldy   #$28        Index auf 40 Zeichen
c1bf   a9  00        lda   #$00        Füllwert
c1c1   99  00  02    sta   $0200,y     in Puffer schreiben
c1c4   88            dey               nächstes Byte
c1c5   10  fa        bpl   $c1c1       und weitermachen, bis Puffer leer
c1c7   60            rts               Ende
-------------------------------------------------------------------------
c1c8                                   Fehlermeldung ausgeben, wobei Track- und Sektornummer
                                       gleich Null gesetzt werden. Fehler- nummer muß bei
                                       Einsprung in Astehen.
c1c8   a0  00        ldy   #$00        Index löschen
c1ca   84  80        sty   $80         Spurnummer
c1cc   84  81        sty   $81         Sektornummer
c1ce   4c  d5  a9    jmp   $a9d5       1571 Fehlerbehandlung; Ende
-------------------------------------------------------------------------
c1d1                                   Untersucht den Befehlsstring auf einen Doppelpunkt ':'.
                                       Wird einer gefunden, so steht seine Position in Y und wird
                                       minus 2 nach $027a ge- speichert. Anschließend wird die lED
                                       am lauf- werk eingeschaltet.
c1d1   a2  00        ldx   #$00        Zeiger auf Filetabelle
c1d3   8e  7a  02    stx   $027a       löschen
c1d6   a9  3a        lda   #$3a        ASCII Wert für ':'
c1d8   20  68  c2    jsr   $c268       Befehlsstring auf ':' durchsuchen
c1db   f0  05        beq   $c1e2       verzweige, wenn nicht gefunden
c1dd   88            dey               Zeiger auf ':' in Y
c1de   88            dey               Zeiger vor ':' setzen
c1df   8c  7a  02    sty   $027a       zeigt dann auf laufwerksnummer
c1e2   4c  68  c3    jmp   $c368       Laufwerksnummer setzen; LED an; Ende
-------------------------------------------------------------------------
c1e5                                   Sucht nach einem Doppelpunkt ':' im Befehlsstring.
c1e5   a0  00        ldy   #$00        Stelle für Suchbeginn
c1e7   a2  00        ldx   #$00        Anzahl der gefundenen Kommas
c1e9   a9  3a        lda   #$3a        ASCII-Code für ':'
c1eb   4c  68  c2    jmp   $c268       Befehlsstring durchsuchen
-------------------------------------------------------------------------
c1ee                                   Prüft die Eingabezeile auf die Bestandteile von
                                       mehrteiligen Befehlen oder Sonderzeichen ('wild cards').
c1ee   20  e5  c1    jsr   $c1e5       Sucht nach ':' in Befehlsstring
c1f1   d0  05        bne   $c1f8       verzweige, wenn gefunden
c1f3   a9  34        lda   #$34        Nummer der Fehlermeldung
c1f5   4c  c8  c1    jmp   $c1c8       '34, SYNTAX ERROR' ausgeben
c1f8   88            dey               Index auf
c1f9   88            dey               Position der Drivenummer
c1fa   8c  7a  02    sty   $027a       setzen
c1fd   8a            txa               Komma vor dem Doppelpunkt?
c1fe   d0  f3        bne   $c1f3       ja, dann 'SYNTAX ERROR'
c200   a9  3d        lda   #$3d        ASCII-Code für '='
c202   20  68  c2    jsr   $c268       Zeile auf '=' durchsuchen
c205   8a            txa               Komma gefunden?
c206   f0  02        beq   $c20a       verzweige, wenn nein
```

```
c208   a9  40       lda   #$40        sonst setze Bit 6
c20a   09  21       ora   #$21        setze Bit 0 und 5 (Funktion s.u.)
c20c   8d  8b  02   sta   $028b       Flags für Befehlssyntax
c20f   e8           inx              Position des Kommas plus 1
c210   8e  77  02   stx   $0277       Länge von Filename 1 setzen
c213   8e  78  02   stx   $0278       Länge von Filename 2 gleichsetzen
c216   ad  8a  02   lda   $028a       Joker (*) Flag; Joker gefunden?
c219   f0  0d       beq   $c228       verzweige, wenn nein
c21b   a9  80       lda   #$80        Vorhandensein von '*' anzeigen
c21d   0d  8b  02   ora   $028b       durch Setzen von Bit 7
c220   8d  8b  02   sta   $028b       als Flag für Befehlssyntax
c223   a9  00       lda   #$00        Jokerflag
c225   8d  8a  02   sta   $028a       löschen
c228   98           tya              wurde ein '=' gefunden?
c229   f0  29       beq   $c254       verzweige, wenn nein
c22b   9d  7a  02   sta   $027a,x     Zeiger auf Filetabelle
c22e   ad  77  02   lda   $0277       Länge von Filename 1
c231   8d  79  02   sta   $0279       als Zeiger auf Name 2 speichern
c234   a9  8d       lda   #$8d        Zeilenendekennzeichen
c236   20  68  c2   jsr   $c268       Zeile bis zum Ende durchsuchen
c239   e8           inx              Anzahl der Kommas
c23a   8e  78  02   stx   $0278       merken
c23d   ca           dex              Originalwert wiederherstellen
c23e   ad  8a  02   lda   $028a       '*' als Joker vorhanden
c241   f0  02       beq   $c245       verzweige, wenn nein
c243   a9  08       lda   #$08        Bit 3 als Flag setzen
c245   ec  77  02   cpx   $0277       Noch weitere Kommas vorhanden?
c248   f0  02       beq   $c24c       verzweige, wenn nein
c24a   09  04       ora   #$04        Bit 2 als Flag setzen
c24c   09  03       ora   #$03        Bit 0 und 1 als Flags setzen
c24e   4d  8b  02   eor   $028b       Status ändern
c251   8d  8b  02   sta   $028b       Neue Flags für Syntax abspeichern
c254   ad  8b  02   lda   $028b       Flags für Befehlssyntax
c257   ae  2a  02   ldx   $022a       Nummer des auszuführenden Befehls
c25a   3d  a5  fe   and   $fea5,x     Prüfen auf Standardsyntax
c25d   d0  01       bne   $c260       verzweige, wenn fehlerhaft
c25f   60           rts              Ende
c260   8d  6c  02   sta   $026c       Fehlerflag setzen
c263   a9  30       lda   #$30        Nummer der Fehlermeldung
c265   4c  c8  c1   jmp   $c1c8       '30, SYNTAX ERROR'ausgeben
-------------------------------------------------------------------------
c268                                  Durchsucht den Befehlsstring auf ein bestimmtes Zeichen (in
                                      Akku). Y bestimmt die Startposition beim Suchen; X enthält
                                      den Zeiger in die Tabelle der Filenamenzeiger. Es wird auf
                                      die Sonderzeichen im Befehlsstring, z.B. '*', '?' und ','
                                      geachtet. Wenn das gesuchte Zeichen gefunden wird, wird das
                                      Z-Flag gesetzt.
c268   8d  75  02   sta   $0275       gesuchtes Zeichen
c26b   cc  74  02   cpy   $0274       Länge der Befehlszeile erreicht?
c26e   b0  2e       bcs   $c29e       verzweige, wenn ja
c270   b1  a3       lda   ($a3),y     Zeichen aus INPUT-Puffer holen
c272   c8           iny              Index auf nächstes Zeichen
c273   cd  75  02   cmp   $0275       mit gesuchtem Zeichen vergleichen
c276   f0  28       beq   $c2a0       verzweige, wenn identisch
c278   c9  2a       cmp   #$2a        vergleiche auf '*'
c27a   f0  04       beq   $c280       verzweige, wenn ja
c27c   c9  3f       cmp   #$3f        vergleiche auf '?'
```

```
c27e   d0  03       bne   $c283       verzweige, wenn nein
c280   ee  8a  02    inc   $028a       Jokerflag setzen
c283   c9  2c       cmp   #$2c        vergleiche auf ','
c285   d0  e4       bne   $c26b       verzweige, wenn nein
c287   98           tya               augenblickliche Position
c288   9d  7b  02    sta   $027b,x     des Kommas in Tabelle
c28b   ad  8a  02    lda   $028a       Jokerflag
c28e   29  7f       and   #$7f        gesetzt?
c290   f0  07       beq   $c299       verzweige, wenn nein
c292   a9  80       lda   #$80        Bit 7
c294   95  e7       sta   $e7,x       als Flag für Joker merken
c296   8d  8a  02    sta   $028a       und in Jokerflag abspeichern
c299   e8           inx               Anzahl der Kommas erhöhen
c29a   e0  04       cpx   #$04        schon 4 Kommas?
c29c   90  cd       bcc   $c26b       weitermachen, wenn nein
c29e   a0  00       ldy   #$00        Index löschen
c2a0   ad  74  02    lda   $0274       Länge der Befehlszeile
c2a3   9d  7b  02    sta   $027b,x     als Flag in Tabelle
c2a6   ad  8a  02    lda   $028a       Jokerflag abfragen
c2a9   29  7f       and   #$7f        Bits 0-6 auswählen
c2ab   f0  04       beq   $c2b1       verzweige, wenn kein Joker
c2ad   a9  80       lda   #$80        Flag für Joker als Bit 7
c2af   95  e7       sta   $e7,x       in Tabelle setzen
c2b1   98           tya               Länge der Befehlszeile
c2b2   60           rts               Ende
--------------------------------------------------------------------------
c2b3                                  Initialisiert alle Kommandotabellen und -zeiger und prüft
                                      die Länge der Eingabezeile. Setzt alle Variablen und Zeiger
                                      zurück. Gibt bei zu langer Zeile eine Fehlermeldung aus.
                                      Sollen nur die Parameter gelöscht werden,so erfolgt der
                                      Einsprung bei $c2dc.
c2b3   a4  a3       ldy   $a3         Zeiger in INPUT-Puffer
c2b5   f0  14       beq   $c2cb       verzweige, wenn jetzt Null
c2b7   88           dey               Zeiger vermindern
c2b8   f0  10       beq   $c2ca       verzweige, wenn Eins
c2ba   20  02  aa    jsr   $aa02       auf 'UD' testen; Zeichen aus String holen
c2bd   c9  0d       cmp   #$0d        Marke für Zeilenende?
c2bf   f0  0a       beq   $c2cb       verzweige, wenn ja
c2c1   88           dey               Zeiger in Puffer vermindern
c2c2   20  02  aa    jsr   $aa02       auf 'U0' testen; Zeichen holen
c2c5   c9  0d       cmp   #$0d        Marke für Zeilenende?
c2c7   f0  02       beq   $c2cb       verzweige, wenn ja
c2c9   c8           iny               Zeiger auf alten ert
c2ca   c8           iny               setzen
c2cb   8c  74  02    sty   $0274       und als Zeilenlänge speichern
c2ce   c0  2a       cpy   #$2a        mit max. Länge (42) vergleichen
c2d0   a0  ff       ldy   #$ff        Index setzen
c2d2   90  08       bcc   $c2dc       verzweige, wenn kleiner als 42
c2d4   8c  2a  02    sty   $022a       Befehlsnummer löschen
c2d7   a9  32       lda   #$32        Nummer der Fehlermeldung
c2d9   4c  c8  c1    jmp   $c1c8       '32, SYNTAX ERROR' ausgeben
c2dc   a0  00       ldy   #$00        Index löschen
c2de   98           tya               Füllwert für Flags; Speicherstellen löschen
c2df   85  a3       sta   $a3         Zeiger auf INPUT-Puffer LO
c2e1   8d  58  02    sta   $0258       Recordlänge
c2e4   8d  4a  02    sta   $024a       gerade behandelter Filetyp
```

```
c2e7   8d  96  02    sta   $0296       Filetyp
c2ea   85  d3       sta   $d3         Zeiger auf ersten Filenamen
c2ec   8d  79  02    sta   $0279       Zeiger auf zweiten Filenamen
c2ef   8d  77  02    sta   $0277       Länge des ersten Filenamens
c2f2   8d  78  02    sta   $0278       Länge des zweiten Filenamens
c2f5   8d  8a  02    sta   $028a       Jokerflag
c2f8   8d  6c  02    sta   $026c       Fehlerflag
c2fb   a2  05       ldx   #$05        Index für 5 Zeichen setzen
c2fd   9d  79  02    sta   $0279,x     Zeiger auf Filetabelle
c300   95  d7       sta   $d7,x       Zeiger auf Directory-Sektoren
c302   95  dc       sta   $dc,x       Zeiger auf Directory-Einträge
c304   95  e1       sta   $e1,x       Tabelle der Laufwerksnummern
c306   95  e6       sta   $e6,x       Tabelle der Jokerflags
c308   9d  7f  02    sta   $027f,x     Spurnummern der Files
c30b   9d  84  02    sta   $0284,x     Sektornummern der Files
c30e   ca           dex               nächstes Zeichen
c30f   d0  ec       bne   $c2fd       und weitermachen
c311   60           rts               Ende
-------------------------------------------------------------------------
c312                                   Laufwerksnummern holen und setzen.
c312   ad  78  02    lda   $0278       Länge des zweiten Filenamens
c315   8d  77  02    sta   $0277       Länge des erster Filenamens
c318   a9  01       lda   #$01        Wert 1
c31a   8d  78  02    sta   $0278       Länge von Filename 2
c31d   8d  79  02    sta   $0279       Zeiger auf Filename 2
c320   ac  8e  02    ldy   $028e       zuletzt benutztes Laufwerk
c323   a2  00       ldx   #$00        Index löschen
c325   86  d3       stx   $d3         Zeiger auf Filenamen 1
c327   bd  7a  02    lda   $027a,x     Position des x-ten Filenamens
c32a   20  3c  c3    jsr   $c33c       Laufwerksnummer holen
c32d   a6  d3       ldx   $d3         Zeiger auf Filename 1
c32f   9d  7a  02    sta   $027a,x     Position des Namens abspeichern
c332   98           tya               Laufwerksnummer
c333   95  e2       sta   $e2,x       in Tabelle abspeichern
c335   e8           inx               nächster Wert
c336   ec  78  02    cpx   $0278       alle Nummern geholt?
c339   90  ea       bcc   $c325       verzweige, wenn nein
c33b   60           rts
-------------------------------------------------------------------------
c33c                                   Holt die Drivenummer aus dem Befehlsstring oder setze
                                       ansonsten den Standardwert (0). A enthält bei Einsprung den
                                       Zeiger in den INPUT-Puffer auf die Drivenummer.
c33c   aa           tax               Zeiger merken
c33d   a0  00       ldy   #$00        Index setzen
c33f   a9  3a       lda   #$3a        ':'
c341   dd  01  02    cmp   $0201,x     Doppelunkt hinter Nummer?
c344   f0  0c       beq   $c352       verzweige, wenn ja
c346   dd  00  02    cmp   $0200,x     Doppelpunkt an dieser Stelle?
c349   d0  16       bne   $c361       verzweige, wenn nein
c34b   e8           inx               nächster Wert
c34c   98           tya               Laufwerksnummer
c34d   29  01       and   #$01        auf 0 und 1 beschränken
c34f   a8           tay               und nach Y
c350   8a           txa               Position zurückholen
c351   60           rts               Ende
c352   bd  00  02    lda   $0200,x     Laufwerksnummer holen
c355   e8           inx               Zeiger um 2
```

```
c356   e8          inx             erhöhen
c357   c9  30       cmp   #$30      vergleiche mit '0'
c359   f0  f2       beq   $c34d     verzweige, wenn ja
c35b   c9  31       cmp   #$31      vergleiche mit '1'
c35d   f0  ee       beq   $c34d     verzweige, wenn ja
c35f   d0  eb       bne   $c34c     unbedingter Sprung
c361   98          tya             Standardwert setzen
c362   09  80       ora   #$80      Flag setzen, daß Standardwert
c364   29  81       and   #$81      benutzt werden muß
c366   d0  e7       bne   $c34f     unbedingter Sprung
--------------------------------------------------------------------------
c368                                Setzt aktuelle Drivenummer und schaltet die LED am Laufwerk
                                    ein.
c368   a9  00       lda   #$00      Befehlssyntaxflag
c36a   8d  8b  02    sta   $028b     löschen
c36d   ac  7a  02    ldy   $027a     Position der Laufwerksnummer
c370   b1  a3       lda   ($a3),y   Zeichen aus Befehlsstring holen
c372   20  bd  c3    jsr   $c3bd     auf gültige Laufwerksnummer testen
c375   10  11       bpl   $c388     verzweige, wenn gültig
c377   c8          iny             Zeiger erhöhen
c378   cc  74  02    cpy   $0274     Länge der Befehlszeile
c37b   b0  06       bcs   $c383     verzweige, wenn Ende erreicht
c37d   ac  74  02    ldy   $0274     Länge der Befehlszeile
c380   88          dey             minus 1
c381   d0  ed       bne   $c370     Laufwerksnummer suchen
c383   ce  8b  02    dec   $028b     Flags für nicht gefunden setzen
c386   a9  00       lda   #$00      Standardwert für Laufwerk
c388   29  01       and   #$01      isolieren und
c38a   85  7f       sta   $7f       setzen
c38c   4c  00  c1    jmp   $c100     LED einschalten; Ende
--------------------------------------------------------------------------
c38f                                Schaltet die Drivenummer in $7f durch Inversion von Bit 0
                                    um.
c38f   a5  7f       lda   $7f       Drivenummer holen
c391   49  01       eor   #$01      Bit 0 invertieren
c393   29  01       and   #$01      andere Werte ausschließen
c395   85  7f       sta   $7f       neue Nummer abspeichern
c397   60          rts             Ende
--------------------------------------------------------------------------
c398                                Zeiger für Filenamen setzen und Filetyp holen und setzen.
c398   a0  00       ldy   #$00      Index löschen
c39a   ad  77  02    lda   $0277     Ende des ersten Filenamens
c39d   cd  78  02    cmp   $0278     evt. gleich Position des 2. Namens?
c3a0   f0  16       beq   $c3b8     ja, dann nur ein Filename vorhanden
c3a2   ce  78  02    dec   $0278     Zeiger herstellen
c3a5   ac  78  02    ldy   $0278     und holen
c3a8   b9  7a  02    lda   $027a,y   Zeiger auf Filenamen holen
c3ab   a8          tay             als Index benutzen und
c3ac   b1  a3       lda   ($a3),y   Filetyp aus Befehlsstring holen
c3ae   a0  04       ldy   #$04      Index in Filetyptabelle setzen
c3b0   d9  bb  fe    cmp   $febb,y   zeigt auf 'S', 'P', 'U', 'R'
c3b3   f0  03       beq   $c3b8     verzweigt, wenn Filetyp gefunden
c3b5   88          dey             Zähler erniedrigen
c3b6   d0  f8       bne   $c3b0     und weiter versuchen
c3b8   98          tya             Nummer des Filetyps oder $00
c3b9   8d  96  02    sta   $0296     merken
c3bc   60          rts             Ende
```

```
--------------------------------------------------------------------------
c3bd                                    Testet auf Drivenummer. Wird werder 0 noch 1 gefunden, so
                                        wird in A das Bit 7 gesetzt, ansonsten werden alle
                                        ungültigen Bits gelöscht.
c3bd   c9  30       cmp   #$30          Drive O?
c3bf   f0  06       beq   $c3c7         verzweige, wenn ja
c3c1   c9  31       cmp   #$31          Drive 1?
c3c3   f0  02       beq   $c3c7         verzweige, wenn ja
c3c5   09  80       ora   #$80          sonst Bit 7=1
c3c7   29  81       and   #$81          andere Werte ausschließen
c3c9   60           rts                 Ende
--------------------------------------------------------------------------
c3ca                                    Geeignete Vorkehrungen zum Suchen einer Datei treffen.
c3ca   a9  00       lda   #$00          Null in den
c3cc   85  6f       sta   $6f           Zwischenspeicher schreiben
c3ce   8d  8d  02   sta   $028d         Flag für Leseversuche setzen
c3d1   48           pha                 Akkuinhalt retten
c3d2   ae  78  02   ldx   $0278         Anzahl der Drivenummern zum Prüfen
c3d5   68           pla                 Akkuinhalt zurückholen
c3d6   05  6f       ora   $6f           mit Zwischenwert verknüpfen
c3d8   48           pha                 Schleife zum Prüfen aller vorhan-
c3d9   a9  01       lda   #$01          denen Drivenummern auf korrekte
c3db   85  6f       sta   $6f           Verarbeitung; das Ergebnis dieser
c3dd   ca           dex                 Verknüpfung wird als Index verwen-
c3de   30  0f       bmi   $c3ef        det, um den richtigen Wert aus der
c3e0   b5  e2       lda   $e2,x         Suchtabelle ab $C440 zu lesen.
c3e2   10  04       bpl   $c3e8         verzweige, wenn positiv
c3e4   06  6f       asl   $6f           Wert
c3e6   06  6f       asl   $6f           mal 4 nehmen
c3e8   4a           lsr                 Drivenummer 0 oder 1?
c3e9   90  ea       bcc   $c3d5         verzweige bei Drivenummer 0
c3eb   06  6f       asl   $6f           Wert mal 8
c3ed   d0  e6       bne   $c3d5         verzweige, wenn ungleich Null
c3ef   68           pla                 vorigen Wert zurückholen
c3f0   aa           tax                 Akkuinhalt als Index
c3f1   bd  3f  c4   lda   $c43f,x       Wert aus Suchtabelle lesen
c3f4   48           pha                 und auf Stack retten
c3f5   29  03       and   #$03          Alle Bits außer 0 und 1 löschen
c3f7   8d  8c  02   sta   $028c         Anzahl der nötigen Lesezugriffe
c3fa   68           pla                 Akkuinhalt zurückholen
c3fb   0a           asl                 Auf Bit 6=1 testen
c3fc   10  3e       bpl   $c43c         verzweige, wenn Bit 6=0 war
c3fe   a5  e2       lda   $e2           erste Drivenummer zum Lesen holen
c400   29  01       and   #$01          andere Werte ausschließen
c402   85  7f       sta   $7f           und als aktuelle Nummer abspeichern
c404   ad  8c  02   lda   $028c         Anzahl der nötigen Lesezugriffe
c407   f0  2b       beq   $c434         verzweige, wenn kein Zugriff
c409   20  3d  c6   jsr   $c63d         Drive, wenn nötig, initialisieren
c40c   f0  12       beq   $c420         verzweige, wenn Status ok
c40e   20  8f  c3   jsr   $c38f         auf anderes Drive umschalten
c411   a9  00       lda   #$00          Anzahl der Lesezugriffe
c413   8d  8c  02   sta   $028c         löschen
c416   20  3d  c6   jsr   $c63d         Drive, wenn nötig, initialisieren
c419   f0  1e       beq   $c439         verzweige, wenn Status ok
c41b   a9  74       lda   #$74          Nummer der Fehlermeldung
c41d   20  c8  c1   jsr   $c1c8         '74, DRIVE NOT READY' ausgeben
c420   20  8f  c3   jsr   $c38f         auf anderes Drive umschalten
```

```
c423   20  3d  c6    jsr   $c63d       Drive, wenn nötig, initialisieren
c426   08              php             Drivestatus merken
c427   20  8f  c3    jsr   $c38f       auf anderes Drive umschalten
c42a   28              plp             Drivestatus zurückholen
c42b   f0  0c          beq   $c439       verzweige, wenn Status ok
c42d   a9  00          lda   #$00        Anzahl der Lesezugriffe
c42f   8d  8c  02      sta   $028c       löschen
c432   f0  05          beq   $c439       unbedingter Sprung
c434   20  3d  c6      jsr   $c63d       Drive, wenn nötig, initialisieren
c437   d0  e2          bne   $c41b       verzweige bei Fehler
c439   4c  00  c1      jmp   $c100       LED für aktuelles Drive einschalten
c43c   2a              rol               zweite Drivenummer in A schieben
c43d   4c  00  c4      jmp   $c400       nächsten Zugriff vorbereiten
--------------------------------------------------------------------------------
c440                                     Bytes zum Suchen einer Datei
c440 00 80 41 01 01 01 01 81
c448 81 81 81 42 42 42 42
--------------------------------------------------------------------------------
c44f                                     Sucht alle Files, die im Befehlsstring gefordert werden und
                                         stellt danach die erforderlichen Zeiger zur weiteren
                                         Bearbeitung her. Aussprung mit RTS, wenn File gefunden
                                         wurde.
c44f   20  ca  c3      jsr   $c3ca       Drive(s) auf Zugriff vorbereiten
c452   a9  00          lda   #$00        Suche nach gültigem File
c454   8d  92  02      sta   $0292       anzeigen
c457   20  ac  c5      jsr   $c5ac       Zeiger setzen und Suchen beginnen
c45a   d0  19          bne   $c475       verzweige, wenn Eintrag gefunden
c45c   ce  8c  02      dec   $028c       Anzahl der Zugriffe vermindern
c45f   10  01          bpl   $c462       verzweige, wenn noch Zugriffe nötig
c461   60              rts               Ende
c462   a9  01          lda   #$01        Flag für Lesezugriff
c464   8d  8d  02      sta   $028d       setzen
c467   20  8f  c3      jsr   $c38f       auf anderes Drive umschalten
c46a   20  00  c1      jsr   $c100       LED für aktuelles Drive einschalten
c46d   4c  52  c4      jmp   $c452       und Suche fortsetzen
c470   20  17  c6      jsr   $c617       nächsten gültigen Eintrag suchen
c473   f0  10          beq   $c485       aufhören, wenn nicht gefunden
c475   20  d8  c4      jsr   $c4d8       Fileeinträge prüfen
c478   ad  8f  02      lda   $028f       Fileeinträge gefunden?
c47b   f0  01          beq   $c47e       verzweige, wenn nein
c47d   60              rts               alles gefunden, also Ende
c47e   ad  53  02      lda   $0253       Flag für gesuchten Eintrag gefunden
c481   30  ed          bmi   $c470       verzweige, wenn nicht gefunden;A=$FF
c483   10  f0          bpl   $c475       unbedingter Sprung
c485   ad  8f  02      lda   $028f       Einträge alle gefunden?
c488   f0  d2          beq   $c45c       weiter suchen, wenn nein
c48a   60              rts               alles gefunden, also Ende
--------------------------------------------------------------------------------
c48b                                     Sucht den nächsten Filenamen im Directory. Wird er auf
                                         einem Laufwerk nicht gefunden, so wird auf das andere
                                         umgeschaltet.
c48b   20  04  c6      jsr   $c604       Zeiger setzen und Suchen beginnen
c48e   f0  1a          beq   $c4aa       verzweige, wenn nicht gefunden
c490   d0  28          bne   $c4ba       unbedingter Sprung
c492   a9  01          lda   #$01        Flag für Lesezugriff
c494   8d  8d  02      sta   $028d       setzen
c497   20  8f  c3      jsr   $c38f       auf anderes Drive umschalten
c49a   20  00  c1      jsr   $c100       LED für aktuelles Drive einschalten
```

```
c49d   a9  00       lda   #$00        Suche nach gültigem File
c49f   8d  92  02   sta   $0292       anzeigen
c4a2   20  ac  c5   jsr   $c5ac       Zeiger setzen und Suche beginnen
c4a5   d0  13       bne   $c4ba       verzweige, wenn Filename gefunden
c4a7   8d  8f  02   sta   $028f       Flag für Einträge gefunden setzen
c4aa   ad  8f  02   lda   $028f       Flag für Fileeinträge gefunden
c4ad   d0  28       bne   $c4d7       verzweige, wenn alle Namen gefunden
c4af   ce  8c  02   dec   $028c       Zähler für Zugriffe vermindern
c4b2   10  de       bpl   $c492       verzweige, wenn Zugriff noch nötig
c4b4   60           rts               Ende
c4b5   20  17  c6   jsr   $c617       Nächsten gültigen Eintrag suchen
c4b8   f0  f0       beq   $c4aa       kein weiterer Eintrag mehr?
c4ba   20  d8  c4   jsr   $c4d8       gesuchte Einträge gefunden?
c4bd   ae  53  02   ldx   $0253       Flag für gesuchten Eintrag gefunden
c4c0   10  07       bpl   $c4c9       verzweige, wenn Eintrag gefunden
c4c2   ad  8f  02   lda   $028f       Flag für Einträge gefunden
c4c5   f0  ee       beq   $c4b5       weitersuchen, wenn nicht gefunden
c4c7   d0  0e       bne   $c4d7       Eintrag gefunden; Erde
c4c9   ad  96  02   lda   $0296       Filetyp überprüfen
c4cc   f0  09       beq   $c4d7       kein Filetyp; Ende
c4ce   b5  e7       lda   $e7,x       Parameter aus Tabelle holen
c4d0   29  07       and   #$07        Filetyp aussondern
c4d2   cd  96  02   cmp   $0296       mit gefundenem vergleichen
c4d5   d0  de       bne   $c4b5       weitersuchen, wenn ungleich
c4d7   60           rts               File gefunden; Ende
-------------------------------------------------------------------------------
c4d8                                  Vergleichen aller Fileeinträge im Directory mit denen im
                                      Befehlsstring; Gleichheit anzeigen.
c4d8   a2  ff       ldx   #$ff        Flag für Eintrag gefunden
c4da   8e  53  02   stx   $0253       setzen
c4dd   e8           inx               X = $00
c4de   8e  8a  02   stx   $028a       Jokerflag löschen
c4e1   20  89  c5   jsr   $c589       Tabelle durchsuchen
c4e4   f0  06       beq   $c4ec       verzweige, wenn noch ein Name fehlt
c4e6   60           rts               Ende; alle Files gefunden
c4e7   20  94  c5   jsr   $c594       Zeiger für nächstes File setzen
c4ea   d0  fa       bne   $c4e6       kein weiteres File; dann Ende
c4ec   a5  7f       lda   $7f         Drivenummer holen und mit der des
c4ee   55  e2       eor   $e2,x       nächsten Filenamens vergleichen
c4f0   4a           lsr               Bit 0 ins Carry-Flag schieben
c4f1   90  0b       bcc   $c4fe       verzweige, wenn Nummer identisch
c4f3   29  40       and   #$40        Test auf Standarddrivenummer
c4f5   f0  f0       beq   $c4e7       nächsten Eintrag suchen
c4f7   a9  02       lda   #$02        kein Eintrag mehr auf Standard-
c4f9   cd  8c  02   cmp   $028c       laufwerk, also Suche auf anderem
c4fc   f0  e9       beq   $c4e7       Drive fortsetzen
c4fe   bd  7a  02   lda   $027a,x     nächsten Filenamen prüfen und
c501   aa           tax               Startwert für Suche setzen
c502   20  a6  c6   jsr   $c6a6       Ende des Namens im Befehlsstring suchen
c505   a0  03       ldy   #$03        Y ist Index in die Eingabezeile
c507   4c  1d  c5   jmp   $c51d       Eingabezeile weiter prüfen
c50a   bd  00  02   lda   $0200,x     Zeichen im INPUT-Puffer mit
c50d   d1  94       cmp   ($94),y     gefundenem Eintrag vergleichen
c50f   f0  0a       beq   $c51b       verzweige, wenn gleich
c511   c9  3f       cmp   #$3f        vergleiche mit '?'
c513   d0  d2       bne   $c4e7       verzweige, wenn ungleich
c515   b1  94       lda   ($94),y     Zeichen aus Directorypuffer
c517   c9  a0       cmp   #$a0        Name schon zu Ende?
```

```
c519   f0  cc       beq   $c4e7       verzweige, wenn ja
c51b   e8           inx               Zeiger in INPUT-Puffer erhöhen
c51c   c8           iny               Zeiger in Directorypuffer erhöhen
c51d   ec  76  02    cpx   $0276       Ende des Namens schon erreicht?
c520   b0  09       bcs   $c52b       verzweige, wenn ja
c522   bd  00  02    lda   $0200,x     nächstes Zeichen holen
c525   c9  2a       cmp   #$2a        '*' Joker?
c527   f0  0c       beq   $c535       verzweige, wenn ja
c529   d0  df       bne   $c50a       weitersuchen
c52b   c0  13       cpy   #$13        Ende des Directoryeintrags?
c52d   b0  06       bcs   $c535       verzweige, wenn ja
c52f   b1  94       lda   ($94),y     Zeichen aus Directorypuffer
c531   c9  a0       cmp   #$a0        'Shift Space'; Ende des Namens?
c533   d0  b2       bne   $c4e7       verzweige, wenn nein
c535   ae  79  02    ldx   $0279       Position des gefundenen Namens
c538   8e  53  02    stx   $0253       merken
c53b   b5  e7       lda   $e7,x       Filetypangabe holen
c53d   29  80       and   #$80        Bit 7 absondern
c53f   8d  8a  02    sta   $028a       und als Joker-Flag abspeichern
c542   ad  94  02    lda   $0294       Zeiger auf Eintrag holen
c545   95  dd       sta   $dd,x       und in Tabelle abspeichern
c547   a5  81       lda   $81         Sektornummer des Fileeintrages
c549   95  d8       sta   $d8,x       in Tabelle eintragen
c54b   a0  00       ldy   #$00        Index zurücksetzen
c54d   b1  94       lda   ($94),y     Filetyp holen
c54f   c8           iny               Zeiger auf nächstes Zeichen
c550   48           pha               Filetyp auf Stack retten
c551   29  40       and   #$40        Bit 6 für Scratch-Schutz
c553   85  6f       sta   $6f         isolieren und abspeichern
c555   68           pla               Filetyp zurückholen
c556   29  df       and   #$df        Bit 5 löschen
c558   30  02       bmi   $c55c       verzweige, wenn File geschlossen
c55a   09  20       ora   #$20        Bit 5 wieder setzen
c55c   29  27       and   #$27        Bits 3,4,6 ud 7 löschen
c55e   05  6f       ora   $6f         Bit 6 zurückholen
c560   85  6f       sta   $6f         Ergebnis in Zwischenspeicher
c562   a9  80       lda   #$80        alle unwichtigen
c564   35  e7       and   $e7,x       Bits in der
c566   05  6f       ora   $6f         Tabelle werden gelöscht und
c568   95  e7       sta   $e7,x       das Ergebnis wieder abgespeichert
c56a   b5  e2       lda   $e2,x       Drivenummerntabelle auf
c56c   29  80       and   #$80        die gleiche Art
c56e   05  7f       ora   $7f         bereinigen
c570   95  e2       sta   $e2,x       und setzen
c572   b1  94       lda   ($94),y     Tracknummer des Files holen
c574   9d  80  02    sta   $0280,x     und ersten Track abspeichern
c577   c8           iny               Zeiger auf nächstes Zeichen
c578   b1  94       lda   ($94),y     Sektornummer des Files holen
c57a   9d  85  02    sta   $0285,x     und als ersten Sektor abspeichern
c57d   ad  58  02    lda   $0258       Recordlänge
c580   d0  07       bne   $c589       verzweige, wenn schon geholt
c582   a0  15       ldy   #$15        Zeiger setzen
c584   b1  94       lda   ($94),y     und Recordlänge aus Puffer holen
c586   8d  58  02    sta   $0258       als aktuelle Länge abspeichern
c589   a9  ff       lda   #$ff        Flag für Fileeinträge gefunden
c58b   8d  8f  02    sta   $028f       setzen
c58e   ad  78  02    lda   $0278       Anzahl der Filenamen
c591   8d  79  02    sta   $0279       setzen und
```

```
c594   ce  79  02   dec   $0279       vermindern
c597   10  01       bpl   $c59a       verzweige, wenn größer oder gleich Null
c599   60           rts               kein Filename mehr; Ende
c59a   ae  79  02   ldx   $0279       Anzahl der Filenamen
c59d   b5  e7       lda   $e7,x       File schon gefunden?
c59f   30  05       bmi   $c5a6       nein, wenn Bit 7 noch gesetzt
c5a1   bd  80  02   lda   $0280,x     Tracknummer schon geholt
c5a4   d0  ee       bne   $c594       ja, wenn ungleich Null
c5a6   a9  00       lda   #$00        Flag
c5a8   8d  8f  02   sta   $028f       zurücksetzen und
c5ab   60           rts               Ende, da alle Namen gefunden
-------------------------------------------------------------------------
c5ac                                  Vorbereitung zum Suchen im Directory. Setzen aller
                                      Standardwerte.
c5ac   a0  00       ldy   #$00        Sektor des ersten Directoryeintrags
c5ae   8c  91  02   sty   $0291       setzen
c5b1   88           dey               Y = $FF
c5b2   8c  53  02   sty   $0253       Flag für Fileeintrag gefunden
c5b5   ad  85  fe   lda   $fe85       Wert 18; Directorytrack
c5b8   85  80       sta   $80         als aktuellen Track übernehmen
c5ba   a9  01       lda   #$01        Sektor 1
c5bc   85  81       sta   $81         als aktuellen Sektor setzen
c5be   8d  93  02   sta   $0293       Flag für letzten Sektor im File
c5c1   20  75  d4   jsr   $d475       angegebenen Sektor lesen
c5c4   ad  93  02   lda   $0293       wurde der letzte Sektor gelesen?
c5c7   d0  01       bne   $c5ca       verzweige, wenn nein
c5c9   60           rts               Ende
c5ca   a9  07       lda   #$07        Zähler für Fileeinträge
c5cc   8d  95  02   sta   $0295       setzen
c5cf   a9  00       lda   #$00        erstes Byte (Tracknummer)
c5d1   20  f6  d4   jsr   $d4f6       aus dem Puffer holen und
c5d4   8d  93  02   sta   $0293       abspeichern
c5d7   20  e8  d4   jsr   $d4e8       Zeiger in aktuellen Puffer setzen
c5da   ce  95  02   dec   $0295       Directoryzähler vermindern
c5dd   a0  00       ldy   #$00        erstes Byte (Filetyp)
c5df   b1  94       lda   ($94),y     aus dem Direetory lesen
c5e1   d0  18       bne   $c5fb       verzweige, wenn kein DEL File
c5e3   ad  91  02   lda   $0291       wurde schon ein DEL File gefunden?
c5e6   d0  2f       bne   $c617       ja, weiter im Directory suchen
c5e8   20  3b  de   jsr   $de3b       sonst Track' und Sektornummer holen
c5eb   a5  81       lda   $81         Sektornummer
c5ed   8d  91  02   sta   $0291       übernehmen
c5f0   a5  94       lda   $94         Pufferzeiger Lo als Zeiger
c5f2   ae  92  02   ldx   $0292       auf ersten Directoryeintrag
c5f5   8d  92  02   sta   $0292       setzen
c5f8   f0  1d       beq   $c617       verzweige, wenn Zeiger gleich Null
c5fa   60           rts               Suche erledigt; also Ende
c5fb   a2  01       ldx   #$01        gültiger Eintrag gefunden
c5fd   ec  92  02   cpx   $0292       war gültiger Eintrag verlangt?
c600   d0  2d       bne   $c62f       verzweige, wenn ja
c602   f0  13       beq   $c617       es wird nach einem DEL File gesucht; also weiter
c604   ad  85  fe   lda   $fe85       Wert 18; Directorytrack
c607   85  80       sta   $80         als aktuelle Tracknummer speichern
c609   ad  90  02   lda   $0290       Sektornummer des aktuellen Direc'
c60c   85  81       sta   $81         toryblocks speichern
c60e   20  75  d4   jsr   $d475       gewünschten Block lesen
c611   ad  94  02   lda   $0294       $0294 Zeiger auf aktuellen Fileeintrag
c614   20  c8  d4   jsr   $d4c8       Zeiger auf Eintrag setzen
```

```
c617   a9  ff       lda   #$ff        Flag für Eintrag gefunden
c619   8d  53  02    sta   $0253       setzen
c61c   ad  95  02    lda   $0295       Zähler für Fileeinträge
c61f   30  08        bmi   $c629       schon alle Einträge geprüft?
c621   a9  20        lda   #$20        nein
c623   20  c6  d1    jsr   $d1c6       Pufferzeiger um 32 erhöhen
c626   4c  d7  c5    jmp   $c5d7       und weitersuchen
c629   20  4d  d4    jsr   $d44d       nächsten Directoryblock lesen
c62c   4c  c4  c5    jmp   $c5c4       und auf Eintrag untersuchen
c62f   a5  94        lda   $94         Directoryzeiger Lo
c631   8d  94  02    sta   $0294       aktueller Pufferzeiger
c634   20  3b  de    jsr   $de3b       Track- und Sektornummer holen
c637   a5  81        lda   $81         Sektornummer als aktuellen Wert
c639   8d  90  02    sta   $0290       des Directoryblocks merken
c63c   60            rts               Ende
--------------------------------------------------------------------------
c63d                                   Testet auf Diskette im Laufwerk und initiali- siert, wenn
                                       die Diskette gewechselt wurde.
c63d   a5  68        lda   $68         Flag zum Sperren des Initialisierens
c63f   d0  28        bne   $c669       ungleich Null, dann Ende
c641   a6  7f        ldx   $7f         aktuelle Drivenummer
c643   56  1c        lsr   $1c,x       wurde Diskette gewechselt?
c645   90  22        bcc   $c669       nein, dann Ende
c647   a9  ff        lda   #$ff        Job-Fehlerflag
c649   8d  98  02    sta   $0298       setzen
c64c   20  0e  d0    jsr   $d00e       auf Diskette im Laufwerk prüfen
c64f   a0  ff        ldy   #$ff        Fehlerflag in Y setzen
c651   c9  02        cmp   #$02        SYNC-Signal gefunden?
c653   f0  0a        beq   $c65f       Fehlermeldung, wenn nein
c655   c9  03        cmp   #$03        Blockheader gefunden?
c657   f0  06        beq   $c65f       Fehlermeldung, wenn nein
c659   c9  0f        cmp   #$0f        Laufwerk ansprechar?
c65b   f0  02        beq   $c65f       Fehlermeldung, wenn nein
c65d   a0  00        ldy   #$00        alles ok; Fehlerflag löschen
c65f   a6  7f        ldx   $7f         aktuelle Drivenummer
c661   4c  a5  a9    jmp   $a9a5       Drivestatus setzen; zurück mit jmp $c664
c664   d0  03        bne   $c669       verzweige, wenn Fehler
c666   20  42  d0    jsr   $d042       Drive initialisieren
c669   a6  7f        ldx   $7f         aktuelle Drivenummer
c66b   4c  17  aa    jmp   $aa17       Drivestatus holen; Ende
--------------------------------------------------------------------------
c66e                                   Schreibt Filenamen aus dem Befehlsstring in den
                                       Directorypuffer. A enthält die Länge des Namens; X enthält
                                       die Position im Befehlsstring und Y enthält die
                                       Puffernummer.
c66e   48            pha               Länge des Filenamens retten
c66f   20  a6  c6    jsr   $c6a6       Ende der Eingabezeile suchen
c672   20  88  c6    jsr   $c688       durch X festgelegten Teil in Puffer
c675   68            pla               Y schreiben und Länge wieder holen
c676   38            sec               Subtraktion vorbereiten und Länge
c677   ed  4b  02    sbc   $024b       mit maximaler Länge vergleichen
c67a   aa            tax               Ergebnis nach X
c67b   f0  0a        beq   $c687       Länge gleich maximaler Länge?
c67d   90  08        bcc   $c687       nein; gröBer? dann verzweigen
c67f   a9  a0        lda   #$a0        kleiner, dann den Rest mit
c681   91  94        sta   ($94),y     'Shift Space' auffüllen
c683   c8            iny               nächstes Byte
c684   ca            dex               Zähler minus 1
```

```
c685   d0  fa       bne   $c681       weitermachen, wenn ungleich Null
c687   60           rts               Ende
-------------------------------------------------------------------------
c688                                  Schreibt den Inhalt des INPUT-Puffers in einen anderen
                                      Puffer. Y enthält die Nummer des anderen Puffers; X enthält
                                      die Position des ersten Zeichens im INPUT-Puffer.
c688   98           tya               Puffernummer nach A
c689   0a           asl               mal 2
c68a   a8           tay               als Index verwenden
c68b   b9  99  00    lda   $0099,y     Pufferzeiger Lo aus Tabelle
c68e   85  94        sta   $94         als Zeiger in Directorypuffer Lo
c690   b9  9a  00    lda   $009a,y     Pufferzeiger Hi aus Tabelle
c693   85  95        sta   $95         als Zeiger in Directorypuffer Hi
c695   a0  00        ldy   #$00        Zeiger auf Null
c697   bd  00  02    lda   $0200,x     Zeichen aus INPUT-Puffer holen
c69a   91  94        sta   ($94),y     und in neuen Puffer schreiben
c69c   c8            iny               Zeiger auf nächstes Zeichen
c69d   f0  06        beq   $c6a5       Ende, wenn Y größer 255
c69f   e8            inx               Zähler plus 1
c6a0   ec  76  02    cpx   $0276       letztes Zeichen bereits erreicht?
c6a3   90  f2        bcc   $c697       weitermachen, wenn nein
c6a5   60            rts               Ende, da alle Zeichen geschrieben
-------------------------------------------------------------------------
c6a6                                  Ende des Namens im Befehlsstring suchen, dessen
                                      Anfangsposition im INPUT-Puffer sich in X befindet.
c6a6   a9  00        lda   #$00        Länge des Namens
c6a8   8d  4b  02    sta   $024b       setzen
c6ab   8a            txa               Position des ersten Zichens
c6ac   48            pha               auf Stack retten
c6ad   bd  00  02    lda   $0200,x     Zeichen aus INPUT-Buffer holen
c6b0   c9  2c        cmp   #$2c        mit ',' vergleichen
c6b2   f0  14        beq   $c6c8       Ende, wenn gleich
c6b4   c9  3d        cmp   #$3d        mit '=' vergleichen
c6b6   f0  10        beq   $c6c8       Ende, wenn gleich
c6b8   ee  4b  02    inc   $024b       Länge des Namens erhöhen
c6bb   e8            inx               Index erhöhen
c6bc   a9  0f        lda   #$0f        Wert 15
c6be   cd  4b  02    cmp   $024b       mit Länge vergleichen
c6c1   90  05        bcc   $c6c8       Ende, wenn Länge größer
c6c3   ec  74  02    cpx   $0274       mit Zeilenende vergleichen
c6c6   90  e5        bcc   $c6ad       weitermachen, wenn kleiner
c6c8   8e  76  02    stx   $0276       Ende des Namens abspeichern
c6cb   68            pla               Anfangsposition zurückholen
c6cc   aa            tax               und als Index
c6cd   60            rts               Ende
-------------------------------------------------------------------------
c6ce                                  Holt einen Fileeintrag aus dem Directory über den internen
                                      Lesekanal; Sekundäradresse 17.
c6ce   a5  83        lda   $83         aktuelle Sekundäradresse
c6d0   48            pha               auf Stack
c6d1   a5  82        lda   $82         aktuelle Kanalnummer
c6d3   48            pha               auf Stack
c6d4   20  de  c6    jsr   $c6de       Eintrag über den Lesekanal holen
c6d7   68            pla               Kanalnummer zurückholen
c6d8   85  82        sta   $82         und setzen
c6da   68            pla               Sekundäradresse zurückholen
```

```
c6db   85  83       sta   $83         und setzen
c6dd   60           rts               Ende
------------------------------------------------------------------------
c6de                                  Routine zum Holen der Fileeinträge.
c6de   a9  11       lda   #$11        17 als Sekundäradresse für internen
c6e0   85  83       sta   $83         Lesekanal setzen
c6e2   20  eb  d0   jsr   $d0eb       Kanal suchen und zum Lesen öffnen
c6e5   20  e8  d4   jsr   $d4e8       Directorypufferzeiger neu setzen
c6e8   ad  53  02   lda   $0253       Flag für Eintrag vorhanden
c6eb   10  0a       bpl   $c6f7       letzter Eintrag?
c6ed   ad  8d  02   lda   $028d       ja, Flag für Zugriff auf anderes
c6f0   d0  0a       bne   $c6fc       Laufwerk testen
c6f2   20  06  c8   jsr   $c806       Kein weiterer Zugriff; BLOCKS FREE
c6f5   18           clc               Meldung schreiben und
c6f6   60           rts               Ende
c6f7   ad  8d  02   lda   $028d       Zugriff auf anderes Drive?
c6fa   f0  1f       beq   $c71b       verzweige, wenn nein
c6fc   ce  8d  02   dec   $028d       Flag vermindern; noch ein Zugriff?
c6ff   d0  0d       bne   $c70e       verzweige, wenn ja
c701   ce  8d  02   dec   $028d       Flag nochmals vermindern
c704   20  8f  c3   jsr   $c38f       auf anderes Drive umschalten
c707   20  06  c8   jsr   $c806       BLOCKS FREE Meldung schreiben
c70a   38           sec               Flag für Umschalten setzen
c70b   4c  8f  c3   jmp   $c38f       auf anderes Drive umschalten; Ende
c70e   a9  00       lda   #$00        Anzahl der Blöcke Hi
c710   8d  73  02   sta   $0273       setzen
c713   8d  8d  02   sta   $028d       Flag für Diskettenzugriff löschen
c716   20  b7  c7   jsr   $c7b7       Directoryüberschrift schreiben
c719   38           sec               Flag für weitere Einträge setzen
c71a   60           rts               Ende
c71b   a2  18       ldx   #$18        24; Länge eines Directoryeintrags
c71d   a0  1d       ldy   #$1d        Position des Hi Bytes der Filelänge
c71f   b1  94       lda   ($94),y     Anzahl derBlöcke Hi
c721   8d  73  02   sta   $0273       in Zwischenspeicher schreiben
c724   f0  02       beq   $c728       verzweige, wenn Länge Null
c726   a2  16       ldx   #$16        22; Länge des Eintrags minus 2
c728   88           dey               zeigt auf Anzahl der Blöcke Lo
c729   b1  94       lda   ($94),y     Anzahl der Blöcke Lo holen
c72b   8d  72  02   sta   $0272       und in Zwischenspeicher
c72e   e0  16       cpx   #$16        Länge minus 2
c730   f0  0a       beq   $c73c       verzweige, wenn X gleich $16
c732   c9  0a       cmp   #$0a        vergleiche Blockzahl Lo mit 10
c734   90  06       bcc   $c73c       verzweige, wenn kleiner
c736   ca           dex               Länge minus 1
c737   c9  64       cmp   #$64        vergleiche Blockzahl Lo mit 100
c739   90  01       bcc   $c73c       verzweige, wenn kleiner
c73b   ca           dex               Länge minus 1
c73c   20  ac  c7   jsr   $c7ac       Directorypuffer löschen
c73f   b1  94       lda   ($94),y     Y=0; Hole Filetyp
c741   48           pha               auf Stack
c742   0a           asl               teste auf Bit 6; Bit 7 ins Carry
c743   10  05       bpl   $c74a       verzweige, wenn nicht gesetzt
c745   a9  3c       lda   #$3c        'kleiner' Zeichen für Scratch Schutz
c747   9d  b2  02   sta   $02b2,x     hinter den Filetyp in Puffer
c74a   68           pla               Filetyp zurückholen
c74b   29  0f       and   #$0f        Bits 0 bis 3 absondern
c74d   a8           tay               als Index in Filetypentabelle
c74e   b9  c5  fe   lda   $fec5,y     3. Buchstaben holen
```

```
c751   9d  b1  02    sta   $02b1,x     und in Puffer schreiben
c754   ca              dex               nächstes Zeichen
c755   b9  c0  fe    lda   $fec0,y     2. Buchstaben holen
c758   9d  b1  02    sta   $02b1,x     und in Puffer schreiben
c75b   ca              dex               nächstes Zeichen
c75c   b9  bb  fe    lda   $febb,y     1. Buchstabe holen
c75f   9d  b1  02    sta   $02b1,x     und in Puffer schreiben
c762   ca              dex               Zeiger
c763   ca              dex               minus 2
c764   b0  05          bcs   $c76b       verzweige, wenn File geschlossen
c766   a9  2a          lda   #$2a        '*' als Zeichen
c768   9d  b2  02    sta   $02b2,x     vor Filetyp in den Puffer schreiben
c76b   a9  a0          lda   #$a0        'Shift Space'
c76d   9d  b1  02    sta   $02b1,x     in den Puffer schreiben
c770   ca              dex               nächstes Zeichen
c771   a0  12          ldy   #$12        Endeposition des Filenamens
c773   b1  94          lda   ($94),y     Filenamen aus aktuellem Puffer
c775   9d  b1  02    sta   $02b1,x     in Directorypuffer schreiben
c778   ca              dex               Zeiger auf
c779   88              dey               nächstes Zeichen
c77a   c0  03          cpy   #$03        schon 3 Zeichen?
c77c   b0  f5          bcs   $c773       weitermachen, wenn nein
c77e   a9  22          lda   #$22        "" Anführungszeichen
c780   9d  b1  02    sta   $02b1,x     vor Filenamen in Puffer schreiben
c783   e8              inx               Zeiger auf nächstes Zeichen
c784   e0  20          cpx   #$20        32; Länge des Fileeintrags
c786   b0  0b          bcs   $c793       Ende schon erreicht?
c788   bd  b1  02    lda   $02b1,x     Zeichen aus Directorypuffer holen
c78b   c9  22          cmp   #$22        mit '"' vergleichen
c78d   f0  04          beq   $c793       verzweige, wenn gleich
c78f   c9  a0          cmp   #$a0        mit 'Shift Space' vergleichen
c791   d0  f0          bne   $c783       verzweige, wenn ungleich
c793   a9  22          lda   #$22        durch '"' ersetzen
c795   9d  b1  02    sta   $02b1,x     und schreiben
c798   e8              inx               nächstes Zeichen
c799   e0  20          cpx   #$20        Ende des Namens schon erreicht?
c79b   b0  0a          bcs   $c7a7       verzweige, wenn ja
c79d   a9  7f          lda   #$7f        Bit 7
c79f   3d  b1  02    and   $02b1,x     in den restlichen Zeichen
c7a2   9d  b1  02    sta   $02b1,x     löschen
c7a5   10  f1          bpl   $c798       unbedingter Sprung
c7a7   20  b5  c4    jsr   $c4b5       nächsten Directoryeintrag suchen
c7aa   38              sec               Flag für weitere Einträge setzen
c7ab   60              rts               Ende
--------------------------------------------------------------------------
c7ac                                     Löschen des Puffers für den Namen im Directory durch Füllen
                                         mit $20 ('Space').
c7ac   a0  1b          ldy   #$1b        Länge des Directorypuffers
c7ae   a9  20          lda   #$20        'Space'
c7b0   99  b0  02    sta   $02b0,y     in den Puffer schreiben
c7b3   88              dey               nächstes Zeichen
c7b4   d0  fa          bne   $c7b0       und weitermachen; 27 mal
c7b6   60              rts               Ende
```

```
------------------------------------------------------------------------
c7b7                                    Kopf des Directory für Anzeige erzeugen und in den
                                        Directory-Puffer $02bO-$02d4 schreiben.
c7b7   20  19  f1    jsr   $f119        Zeiger für BAM setzen
c7ba   20  df  f0    jsr   $f0df        BAM, wenn nötig, von Diskette lesen
c7bd   20  ac  c7    jsr   $c7ac        Directorypuffer löschen
c7c0   a9  ff        lda   #$ff         Zwischenspeicher
c7c2   85  6f        sta   $6f          setzen
c7c4   a6  7f        ldx   $7f          aktuelle Drivenummer
c7c6   8e  72  02    stx   $0272        als Blockanzahl Lo in Specher
c7c9   a9  00        lda   #$00         Null
c7cb   8d  73  02    sta   $0273        als Blockanzahl Hi
c7ce   a6  f9        ldx   $f9          aktuelle Puffernummer
c7d0   bd  e0  fe    lda   $fee0,x      Pufferadresse Hi als aktuellen
c7d3   85  95        sta   $95          Pufferzeiger Hi setzen
c7d5   ad  88  fe    lda   $fe88        144; Position des Diskettennamens
c7d8   85  94        sta   $94          als Pufferzeiger Lo merken
c7da   a0  16        ldy   #$16         Anzahl der Zeichen im Namen
c7dc   b1  94        lda   ($94),y      Zeichen des Namens aus Puffer holen
c7de   c9  a0        cmp   #$a0         'Shift Space'?
c7e0   d0  0b        bne   $c7ed        verzweige, wenn nein
c7e2   a9  31        lda   #$31         ASCII-Code für '1'
c7e4   2c            .byte $2c          nächsten Befehl überspringen
c7e5   b1  94        lda   ($94),y      Zeichen aus Puffer holen
c7e7   c9  a0        cmp   #$a0         'Shift Space' ?
c7e9   d0  02        bne   $c7ed        verzweige, wenn nein
c7eb   a9  20        lda   #$20         'Space'
c7ed   99  b3  02    sta   $02b3,y      in den Directorypuffer schreiben
c7f0   88            dey                Zeiger auf nächstes Zeichen
c7f1   10  f2        bpl   $c7e5        weitermachen, bis Puffer voll
c7f3   a9  12        lda   #$12         'RVS ON'
c7f5   8d  b1  02    sta   $02b1        in den Puffer schreiben
c7f8   a9  22        lda   #$22         1111
c7fa   8d  b2  02    sta   $02b2        vor und
c7fd   8d  c3  02    sta   $02c3        hinter den Namen schreiben
c800   a9  20        lda   #$20         'Space'
c802   8d  c4  02    sta   $02c4        als letztes Zeichen des Namens
c805   60            rts                Ende
------------------------------------------------------------------------
c806                                    Zeile mit 'BLOCKS FREE' erzeugen und in den Di-
                                        rectorypuffer schreiben.
c806   20  ac  c7    jsr   $c7ac        Directorypuffer löschen
c809   a0  0b        ldy   #$0b         12 Bytes
c80b   b9  17  c8    lda   $c817,y      'BLOCKS FREE.' holen und
c80e   99  b1  02    sta   $02b1,y      in den Puffer schreiben
c811   88            dey                nächstes Zeichen
c812   10  f7        bpl   $c80b        und weitermachen, bis alles übertragen
c814   4c  4d  ef    jmp   $ef4d        Anzahl der freien Blöcke holen; Ende
------------------------------------------------------------------------
c817                                    Bytes für 'BLOCKS FREE'
c817 42 4c 4f 43 4b 53 20 46 52 45 45 2e
------------------------------------------------------------------------
c823                                    SCRATCH-Befehl (S-Befehl).
c823   20  98  c3    jsr   $c398        Filetyp ermitteln und Werte setzen
c826   20  20  c3    jsr   $c320        Drivenummer aus Befehlsstring holen
c829   20  ca  c3    jsr   $c3ca        Zugriff auf Laufwerk vorbereiten
c82c   a9  00        lda   #$00         Anzahl der gelöschten Files
c82e   85  86        sta   $86          setzen
c830   20  9d  c4    jsr   $c49d        Ersten Directoryeintrag suchen
```

```
c833   30 3d       bmi   $c872       verzweige, wenn nicht gefunden
c835   20 b7 dd    jsr   $ddb7       File ordnungsgemäß geschlossen?
c838   90 33       bcc   $c86d       verzweige, wenn nein; kein SCRATCH
c83a   a0 00       ldy   #$00        Zeiger auf Fileeintrag
c83c   b1 94       lda   ($94),y     Filetyp holen
c83e   29 40       and   #$40        Bit 6 isolieren; SCRATCH-Schutz?
c840   d0 2b       bne   $c86d       verzweige, wenn ja; kein SCRATCH
c842   20 b6 c8    jsr   $c8b6       Filetyp = $00 setzen; BAM schreiben
c845   a0 13       ldy   #$13        Index auf 19. Zeichen
c847   b1 94       lda   ($94),y     Tracknummer des ersten Side-Sektors
c849   f0 0a       beq   $c855       verzweige, wenn nicht vorhanden
c84b   85 80       sta   $80         Tracknummer übernehmen
c84d   c8          iny               nächstes Zeichen
c84e   b1 94       lda   ($94),y     Sektornummer holen und
c850   85 81       sta   $81         ebenfalls übernehmen
c852   20 7d c8    jsr   $c87d       Side-Sektoren löschen
c855   ae 53 02    ldx   $0253       Nummer der gefundenen Datei
c858   a9 20       lda   #$20        Bit 5
c85a   35 e7       and   $e7,x       in Tabelle testen
c85c   d0 0d       bne   $c86b       verzweige, wenn gesetzt; File offen
c85e   bd 80 02    lda   $0280,x     Tracknummer des Files holen und
c861   85 80       sta   $80         übernehmen
c863   bd 85 02    lda   $0285,x     Sektornummer
c866   85 81       sta   $81         ebenfalls übernehmen
c868   20 7d c8    jsr   $c87d       Datei löschen; BAM neu schreiben
c86b   e6 86       inc   $86         Anzahl der gelöschten Files erhöhen
c86d   20 8b c4    jsr   $c48b       nächsten Dateieintrag suchen
c870   10 c3       bpl   $c835       verzweige, wenn vorhanden; SCRATCH
c872   a5 86       lda   $86         Anzahl der gelöschten Files
c874   85 80       sta   $80         für die Ausgabe bereitstellen
c876   a9 01       lda   #$01        Nummer der (Fehler)Meldung
c878   a0 00       ldy   #$00        Nummer des Sektors
c87a   4c a3 c1    jmp   $c1a3       '01, FILES SCRATCHED' ausgeben
--------------------------------------------------------------------------------
c87d                                 Datei löschen und Blöcke in der BAM wieder freigeben; BAM
                                     schreiben.
c87d   20 5f ef    jsr   $ef5f       Ersten Dateiblock wieder freigeben
c880   20 75 d4    jsr   $d475       internen Lesekanal öffnen (Sekundäradresse 17)
c883   20 19 f1    jsr   $f119       Puffernummer für BAM nach X holen
c886   b5 a7       lda   $a7,x       aktuelle Kanalnummer
c888   c9 ff       cmp   #$ff        ist der Puffer inaktiv?
c88a   f0 08       beq   $c894       verzweige, wenn ja
c88c   ad f9 02    lda   $02f9       Flag für BAM nicht auf Diskette
c88f   09 40       ora   #$40        schreiben; Bit 6 setzen
c891   8d f9 02    sta   $02f9       Anzeige, daß beide Puffer aktiv
c894   a9 00       lda   #$00        Parameter für gerade aktiven
c896   20 c8 d4    jsr   $d4c8       Puffer setzen
c899   20 56 d1    jsr   $d156       Tracknummer aus Puffer holen
c89c   85 80       sta   $80         und merken
c89e   20 56 d1    jsr   $d156       Sektornummer aus Puffer holen
c8a1   85 81       sta   $81         und merken
c8a3   a5 80       lda   $80         Tracknummer (letzter Block)?
c8a5   d0 06       bne   $c8ad       verzweige, wenn ungleich Null
c8a7   20 f4 ee    jsr   $eef4       BAM schreiben
c8aa   4c 27 d2    jmp   $d227       lesekanal wieder freigeben
c8ad   20 5f ef    jsr   $ef5f       Block in BAM freigeben
c8b0   20 4d d4    jsr   $d44d       Nächsten Block des Files lesen
c8b3   4c 94 c8    jmp   $c894       und ebenfalls freigeben
```

```
------------------------------------------------------------------------------
c8b6                                    Fileeintrag im Directory löschen.
c8b6   a0  00       ldy   #$00          Zeiger auf erstes Zeichen im
c8b8   98           tya                 Fileeintrag setzen (Filetyp)
c8b9   91  94       sta   ($94),y       und diesen löschen (DEL)
c8bb   20  5e  de   jsr   $de5e         geänderten Block wieder schreiben
c8be   4c  99  d5   jmp   $d599         und Ende des Jobs abwarten; Ende
------------------------------------------------------------------------------
c8c1                                    BACKUP-Befehl (D-Befehl; n.v.).
c8c1   a9  31       lda   #$31          Nummer der Fehlermeldung
c8c3   4c  c8  c1   jmp   $c1c8         '31, SYNTAX ERROR' ausgeben
------------------------------------------------------------------------------
c8c6                                    Routine zum Formatieren einer Diskette.
c8c6   a9  4c       lda   #$4c          OP-Code für JMP
c8c8   8d  00  06   sta   $0600         in Puffer schreiben
c8cb   a9  c7       lda   #$c7          Lo-Byte der Adresse
c8cd   8d  01  06   sta   $0601         in Puffer
c8d0   a9  fa       lda   #$fa          Hi-Byte der Adresse ($fac7)
c8d2   8d  02  06   sta   $0602         ebenfalls in Puffer
c8d5   a9  03       lda   #$03          Für Puffer 3 die aktuelle
c8d7   20  d3  d6   jsr   $d6d3         Track- und Sektornummer setzen
c8da   a5  7f       lda   $7f           aktuelle Drivenummer
c8dc   09  e0       ora   #$e0          Jobcode; Programm im Puffer starten
c8de   85  03       sta   $03           in Jobspeicher; an Diskcontroll
c8e0   a5  03       lda   $03           Rückmeldung lesen
c8e2   30  fc       bmi   $c8e0         auf Endemeldung warten
c8e4   c9  02       cmp   #$02          o.k. Meldung?
c8e6   90  07       bcc   $c8ef         verzweige, wenn ja
c8e8   a9  03       lda   #$03          Nummer der Fehlermeldung
c8ea   a2  00       ldx   #$00          für Drive 0
c8ec   4c  0a  e6   jmp   $e60a         '21, READ ERROR' ausgeben
c8ef   60           rts                 Job fehlerlos beendet
------------------------------------------------------------------------------
c8f0                                    COPY-Befehl (C-Befehl).
c8f0   a9  e0       lda   #$e0          BAM-Puffer
c8f2   8d  4f  02   sta   $024f         in Belegungsplan freigeben
c8f5   20  d1  f0   jsr   $f0d1         Track- und Sektor der BAM setzen
c8f8   20  19  f1   jsr   $f119         Puffernummer für BAM holen
c8fb   a9  ff       lda   #$ff          Code für Puffer unbenutzt
c8fd   95  a7       sta   $a7,x         für BAM-Puffer setzen
c8ff   a9  0f       lda   #$0f          alle Käle
c901   8d  56  02   sta   $0256         freigeben
c904   20  e5  c1   jsr   $c1e5         ':' in Eingabezeile suchen
c907   d0  03       bne   $c90c         verzweige, wenn gefunden
c909   4c  c1  c8   jmp   $c8c1         '31, SYNTAX ERROR' ausgeben
c90c   20  f8  c1   jsr   $c1f8         Eingabezeile prüfen
c90f   20  20  c3   jsr   $c320         Drivenummern in Tabelle eintragen
c912   ad  8b  02   lda   $028b         Flags für Syntaxprüfung
c915   29  55       and   #$55          mit %01010101 prüfen
c917   d0  0f       bne   $c928         verzweige, wenn normales COPY
c919   ae  7a  02   ldx   $027a         Filetabelle
c91c   bd  00  02   lda   $0200,x       Zeichen des entsprechenden Filenamens
c91f   c9  2a       cmp   #$2a          mit '*' vergleichen
c921   d0  05       bne   $c928         verzweige, wenn kein '*'
c923   a9  30       lda   #$30          Nummer der Fehlermeldung
c925   4c  c8  c1   jmp   $c1c8         '30, SYNTAX ERROR' ausgeben
c928   ad  8b  02   lda   $028b         Flags für Befehlssyntax
c92b   29  d9       and   #$d9          mit %11011001 prüfen
```

```
c92d   d0  f4       bne   $c923       verzweige, wenn Syntax unkorrekt
c92f   4c  52  c9    jmp   $c952       sonst ordnungsgemäß kopieren
-------------------------------------------------------------------------
c932                                   Parameter für das Kopieren einer ganzen Diskette setzen
                                       (n.v.).
c932   a9  00       lda   #$00        Werte löschen:
c934   8d  58  02    sta   $0258       Recordlänge
c937   8d  8c  02    sta   $028c       Anzahl der Drivezugriffe
c93a   8d  80  02    sta   $0280       Tracknummer des Files für Puffer 0
c93d   8d  81  02    sta   $0281       Tracknummer des Files für Puffer 1
c940   a5  e3       lda   $e3         Standardwert für Drivenummer (0)
c942   29  01       and   #$01        Bit 0 isolieren
c944   85  7f       sta   $7f         und als aktuelle Drivenummer setzen
c946   09  01       ora   #$01        $01 als Sektornummer
c948   8d  91  02    sta   $0291       des ersten Directoryblocks setzen
c94b   ad  7b  02    lda   $027b       zweiten Fileeintrag gleich dem
c94e   8d  7a  02    sta   $027a       ersten setzen
c951   60           rts               Ende
-------------------------------------------------------------------------
c952                                   Datei(en) in ein File kopieren
c952   20  4f  c4    jsr   $c44f       Datei im Directory suchen
c955   ad  78  02    lda   $0278       Anzahl der Filenamen im Befehl
c958   c9  03       cmp   #$03        weniger als drei?
c95a   90  45       bcc   $c9a1       verzweige, wenn ja
c95c   a5  e2       lda   $e2         erste Drivenummer
c95e   c5  e3       cmp   $e3         gleich der zweiten Drivenummer?
c960   d0  3f       bne   $c9a1       verzweige, wenn nein
c962   a5  dd       lda   $dd         Eintrag des ersten Files
c964   c5  de       cmp   $de         gleich Eintrag des zweiten Files?
c966   d0  39       bne   $c9a1       verzweige, wenn nein
c968   a5  d8       lda   $d8         Directoryblock des ersten Eintrags
c96a   c5  d9       cmp   $d9         gleich dem des zweiten Eintrags?
c96c   d0  33       bne   $c9a1       verzweige, wenn nein
c96e   20  cc  ca    jsr   $cacc       gewünschtes File vorhanden?
c971   a9  01       lda   #$01        Zeiger auf zweiten Filenamen
c973   8d  79  02    sta   $0279       auf erstes Zeichen setzen
c976   20  fa  c9    jsr   $c9fa       File im Directory suchen
c979   20  25  d1    jsr   $d125       und Filetyp holen
c97c   f0  04       beq   $c982       verzweige, wenn SCRATCHED-File
c97e   c9  02       cmp   #$02        Dateitypen identisch?
c980   d0  05       bne   $c987       verzweige, wenn ja
c982   a9  64       lda   #$64        Nummer der Fehlermeldung
c984   20  c8  c1    jsr   $c1c8       '64, FILE TYPE MISMATCH' ausgeben
c987   a9  12       lda   #$12        Nummer des internen Schreibkanals
c989   85  83       sta   $83         (18) setzen
c98b   ad  3c  02    lda   $023c       Kanalstatus
c98e   8d  3d  02    sta   $023d       übernehmen
c991   a9  ff       lda   #$ff        Code für nicht benutzten Kanal
c993   8d  3c  02    sta   $023c       setzen
c996   20  2a  da    jsr   $da2a       erstes File kopieren
c999   a2  02       ldx   #$02        und APPEND weiterer Files
c99b   20  b9  c9    jsr   $c9b9       durchführen
c99e   4c  94  c1    jmp   $c194       Ende
c9a1   20  a7  c9    jsr   $c9a7       Files kopieren
c9a4   4c  94  c1    jmp   $c194       und fertig
```

```
------------------------------------------------------------------------
c9a7                                    Kopieren der Dateien.
c9a7   20  e7  ca    jsr   $cae7        prüfen, ob File vorhanden
c9aa   a5  e2        lda   $e2          Drivenummer des ersten Files
c9ac   29  01        and   #$01         isolieren
c9ae   85  7f        sta   $7f          und als aktuelle Drivenummer setzen
c9b0   20  86  d4    jsr   $d486        internen Schreibkanal öffnen
c9b3   20  e4  d6    jsr   $d6e4        neue Datei in Directory eintragen
c9b6   ae  77  02    ldx   $0277        Zeiger in ersten Filenamen
c9b9   8e  79  02    stx   $0279        als aktuellen Zeiger übernehmen
c9bc   20  fa  c9    jsr   $c9fa        Directoryblock(s) lesen
c9bf   a9  11        lda   #$11         Nummer des internen Lesekanals
c9c1   85  83        sta   $83          als aktuelle Sekundäradresse setzen
c9c3   20  eb  d0    jsr   $d0eb        unbenutzten Lesekanal suchen
c9c6   20  25  d1    jsr   $d125        Abfrage auf relative Datei
c9c9   d0  03        bne   $c9ce        verzweige, wenn kein relatives File
c9cb   20  53  ca    jsr   $ca53        Kopieren von relativen Dateien
c9ce   a9  08        lda   #$08         EOI-Signal
c9d0   85  f8        sta   $f8          setzen
c9d2   4c  d8  c9    jmp   $c9d8        weiter...
c9d5   20  9b  cf    jsr   $cf9b        letztes Byte auf Diskette schreiben
c9d8   20  35  ca    jsr   $ca35        Byte über internen Lesekanal holen
c9db   a9  80        lda   #$80         Byte 7 abfragen, d.h.
c9dd   20  a6  dd    jsr   $dda6        auf letzten Record testen
c9e0   f0  f3        beq   $c9d5        verzweige, wenn noch weitere folgen
c9e2   20  25  d1    jsr   $d125        Filetyp holen
c9e5   f0  03        beq   $c9ea        verzweige, wenn weitere Records
c9e7   20  9b  cf    jsr   $cf9b        letztes Datenbyte schreiben
c9ea   ae  79  02    ldx   $0279        aktuelle Filenamenzeiger
c9ed   e8            inx               erhöhen und mit der
c9ee   ec  78  02    cpx   $0278        Länge des Filenamen vergleichen
c9f1   90  c6        bcc   $c9b9        verzweige, wenn kleiner
c9f3   a9  12        lda   #$12         Nummer für internen Schreibkanal
c9f5   85  83        sta   $83          als aktuelle Kanalnummer setzen
c9f7   4c  02  db    jmp   $db02        COPY-Kanal und File schließen
------------------------------------------------------------------------
c9fa                                    Internen Kanal zum Lesen eines Files öffnen.
c9fa   ae  79  02    ldx   $0279        Zeiger auf Filenamen
c9fd   b5  e2        lda   $e2,x        zugehörige Drivenummer holen
c9ff   29  01        and   #$01         isolieren
ca01   85  7f        sta   $7f          und als aktuelle Drivenummer merken
ca03   ad  85  fe    lda   $fe85        Nummer 18, Directorytrack
ca06   85  80        sta   $80          als aktuelle Tracknummer setzen
ca08   b5  d8        lda   $d8,x        richtigen Sektor des Directory
ca0a   85  81        sta   $81          ebenfalls übernehmen
ca0c   20  75  d4    jsr   $d475        Directoryblock lesen
ca0f   ae  79  02    ldx   $0279        Zeiger für aktuellen Filenamen
ca12   b5  dd        lda   $dd,x        zum Holen der Position des Eintrags
ca14   20  c8  d4    jsr   $d4c8        setzen; Fileparameter holen
ca17   ae  79  02    ldx   $0279        Zeiger für aktuellen Filenamen
ca1a   b5  e7        lda   $e7,x        als Index auf Filetypmaske
ca1c   29  07        and   #$07         Filetyp daraus isolieren und
ca1e   8d  4a  02    sta   $024a        als aktuellen Filetyp merken
ca21   a9  00        lda   #$00         Null als
ca23   8d  58  02    sta   $0258        Recordlänge setzen; kein relatives File
ca26   20  a0  d9    jsr   $d9a0        internen Lesekanal öffnen
ca29   a0  01        ldy   #$01         Filetyp
ca2b   20  25  d1    jsr   $d125        auf relative Datei testen
```

```
ca2e   f0  01       beq   $ca31       verzweige, wenn keine relative Datei
ca30   c8           iny               Y=2
ca31   98           tya               gibt Anzahl der Jobs an (1 oder 2)
ca32   4c  c8  d4   jmp   $d4c8       Track- und Sektorparameter setzen; Ende
-------------------------------------------------------------------------
ca35                                  Byte über den internen Lesekanal holen und Sekundäradresse
                                      für Lesekanal setzen.
ca35   a9  11       lda   #$11        17 als Sekundäradresse des internen Lesekanals
ca37   85  83       sta   $83         setzen
ca39   20  9b  d3   jsr   $d39b       ein Byte über den Lesekanal holen
ca3c   85  85       sta   $85         und zwischenspeichern
ca3e   a6  82       ldx   $82         Kanalnummer holen und als Index in
ca40   b5  f2       lda   $f2,x       die Kanlstatustabelle benutzen
ca42   29  08       and   #$08        EOI-Bit isolieren
ca44   85  f8       sta   $f8         und abspeichern
ca46   d0  0a       bne   $ca52       verzweige, wenn EOI gesetzt
ca48   20  25  d1   jsr   $d125       Filetyp holen; auf relatives File testen
ca4b   f0  05       beq   $ca52       verzweige, wenn kein relatives File
ca4d   a9  80       lda   #$80        Flag für letzten Record
ca4f   20  97  dd   jsr   $dd97       setzen
ca52   60           rts               Ende
-------------------------------------------------------------------------
ca53                                  Spezialroutine zum Kopieren relativer Dateien
ca53   20  d3  d1   jsr   $d1d3       Drivenummer setzen
ca56   20  cb  e1   jsr   $e1cb       Parameter auf letzten Record setzen
ca59   a5  d6       lda   $d6         Zeiger in Side-Sektor
ca5b   48           pha               merken
ca5c   a5  d5       lda   $d5         Nummer des aktuellen Side-Sektors
ca5e   48           pha               merken
ca5f   a9  12       lda   #$12        interner Schreibkanal (Sekundäradresse 18)
ca61   85  83       sta   $83         als aktuellen Kanal setzen
ca63   20  07  d1   jsr   $d107       freien Schreibkanal entsprechend suchen
ca66   20  d3  d1   jsr   $d1d3       Drivenummer für Kanal setzen
ca69   20  cb  e1   jsr   $e1cb       letzten Side-Sektor setzen
ca6c   20  9c  e2   jsr   $e29c       Block von Diskette in Puffer lesen
ca6f   a5  d6       lda   $d6         Zeiger in Side-Sektor
ca71   85  87       sta   $87         zwischenspeichern
ca73   a5  d5       lda   $d5         Nummer des aktuellen Si-Sektors
ca75   85  86       sta   $86         zwischenspeichern
ca77   a9  00       lda   #$00        Parameter für weitere Behandlung löschen
ca79   85  88       sta   $88         Zwischenspeicher
ca7b   85  d4       sta   $d4         Zeiger auf Beginn des Records
ca7d   85  d7       sta   $d7         Zeiger auf Position in Rcord
ca7f   68           pla               Nummer des aktuellen Side-Sektors
ca80   85  d5       sta   $d5         zurückholen
ca82   68           pla               Zeiger in Side-Sektor
ca83   85  d6       sta   $d6         zurückholen
ca85   4c  3b  e3   jmp   $e33b       Ende
-------------------------------------------------------------------------
ca88                                  RENAME-Befehl (R-Befehl).
ca88   20  20  c3   jsr   $c320       Drivenummer(n) aus Befehlszeile holen
ca8b   a5  e3       lda   $e3         Standardlaufwerksnummer (0)
ca8d   29  01       and   #$01        Bit für Nummer isolieren
ca8f   85  e3       sta   $e3         Nummer 'bereinigt' wieder abspeichern
ca91   c5  e2       cmp   $e2         mit vorheriger Nummer vergleichen
ca93   f0  02       beq   $ca97       verzweige, wenn gleich
ca95   09  80       ora   #$80        Bit 7 als Flag für Suche auf 2 Drives
ca97   85  e2       sta   $e2         setzen
```

```
ca99   20  4f  c4   jsr   $c44f       File in Directory (Directories) suchen
ca9c   20  e7  ca   jsr   $cae7       Filename schon vorhanden?
ca9f   a5  e3       lda   $e3         Drivenummer (Standardwert)
caa1   29  01       and   #$01        Wert 0 oder 1 isolieren
caa3   85  7f       sta   $7f         als aktuelle Nummer übernehmen
caa5   a5  d9       lda   $d9         Sektornummer des Fileeintrags
caa7   85  81       sta   $81         als aktuelle Sektornummer setzen
caa9   20  57  de   jsr   $de57       benötigten Directorysektor lesen
caac   20  99  d5   jsr   $d599       auf Ende des Jobs warten
caaf   a5  de       lda   $de         Zeiger auf Directoryeintrag
cab1   18           clc               Addition vorbereiten
cab2   69  03       adc   #$03        zeigt jetzt auf Filenamen im Eintrag
cab4   20  c8  d4   jsr   $d4c8       Pufferzeiger auf Filenamen setzen
cab7   20  93  df   jsr   $df93       Nummer des aktiven Puffers holen
caba   a8           tay               ins Y-Register
cabb   ae  7a  02   ldx   $027a       Zeiger auf Filetabelle
cabe   a9  10       lda   #$10        Anzahl 16; maximale Länge des Filenamens
cac0   20  6e  c6   jsr   $c66e       Filenamen in Puffer schreiben
cac3   20  5e  de   jsr   $de5e       Directorysektor wieder auf Diskette
cac6   20  99  d5   jsr   $d599       auf Ende des Jobs warten
cac9   4c  94  c1   jmp   $c194       Ende; Diskstatus bereitstellen
-------------------------------------------------------------------------------
cacc                                  Prüft auf Vorhandensein des aktuellen Files.
cacc   a5  e8       lda   $e8         Maske für Fileidentifikation
cace   29  07       and   #$07        Filetyp isolieren
cad0   8d  4a  02   sta   $024a       und als aktuellen Filetyp abspeichern
cad3   ae  78  02   ldx   $0278       Zeiger auf Filenamen
cad6   ca           dex               minus 1
cad7   ec  77  02   cpx   $0277       mit Anfangswert vergleichen
cada   90  0a       bcc   $cae6       verzweige, wenn kleiner
cadc   bd  80  02   lda   $0280,x     entsprechende Tracknummer holen
cadf   d0  f5       bne   $cad6       verzweige, wenn vorhanden (ungleich 0)
cae1   a9  62       lda   #$62        Fehlernummer
cae3   4c  c8  c1   jmp   $c1c8       '62, FILE NOT FOUND' ausgeben
cae6   60           rts               Ende, alles ok
-------------------------------------------------------------------------------
cae7                                  Prüft auf Namensgleichheit zweier Files.
cae7   20  cc  ca   jsr   $cacc       File mit aktuellem Namen vorhanden?
caea   bd  80  02   lda   $0280,x     zugehörige Tracknummer
caed   f0  05       beq   $caf4       verzweige, wenn nicht vorhanden (gleich 0)
caef   a9  63       lda   #$63        Nummer der Fehlermeldung
caf1   4c  c8  c1   jmp   $c1c8       '63, FILE EXISTS' ausgeben
caf4   ca           dex               Zeiger auf Tracknummer minus 1
caf5   10  f3       bpl   $caea       weitermachen, wenn größer gleich 0
caf7   60           rts               Ende, kein File mit gleichem Namen
-------------------------------------------------------------------------------
caf8                                  MEMORY-Befehle (Steuerroutine).
caf8   ad  01  02   lda   $0201       zweites Zeichen aus Befehlsstring
cafb   c9  2d       cmp   #$2d        mit I_I vergleichen
cafd   d0  4c       bne   $cb4b       verzweige, wenn kein I_I
caff   ad  03  02   lda   $0203       sonst viertes Byte aus Befehlsstring
cb02   85  6f       sta   $6f         als Adressbyte Lo speichern
cb04   ad  04  02   lda   $0204       fünftes Byte aus Befehlsstring
cb07   85  70       sta   $70         als Adressbyte Hi speichern
cb09   a0  00       ldy   #$00        Y-Register zurücksetzen
cb0b   ad  02  02   lda   $0202       drittes Zeichen aus Befehlsstring
cb0e   c9  52       cmp   #$52        mit 'R' vergleichen (für M-R)
cb10   f0  0e       beq   $cb20       verzweige zu M-R-Befehl, wenn gleich
```

```
cb12   20  58  f2    jsr   $f258       (RTS)
cb15   c9  57        cmp   #$57        mit 'W' vergleichen (für M-W)
cb17   f0  37        beq   $cb50       verzweige zu M-W-Befehl, wenn gleich
cb19   c9  45        cmp   #$45        mit 'E' vergleichen (für M-E)
cb1b   d0  2e        bne   $cb4b       zum Fehlerausgang, wenn ungleich
cb1d   6c  6f  00    jmp   ($006f)     M-E-Befehl ausführen
-------------------------------------------------------------------------
cb20                                   M-R-Befehl.
cb20   b1  6f        lda   ($6f),y     Byte an der gegebenen Adresse holen
cb22   85  85        sta   $85         und für Ausgabe zwischenspeichern
cb24   ad  74  02    lda   $0274       Länge des Befehlsstrings
cb27   c9  06        cmp   #$06        mit 6 vergleichen
cb29   90  1a        bcc   $cb45       verzweige, wenn nur 1 Byte; Ende
cb2b   ae  05  02    ldx   $0205       Anzahl der gewünschten Bytes holen
cb2e   ca            dex               minus 1
cb2f   f0  14        beq   $cb45       verzweige, wenn nur 1 Byte gewünscht war
cb31   8a            txa               Anzahl der zu lesende Bytes
cb32   18            clc               Addition vorbereiten
cb33   65  6f        adc   $6f         Adressbyte Lo addieren
cb35   e6  6f        inc   $6f         Adressbyte Lo auf 2. Byte einstellen
cb37   8d  49  02    sta   $0249       Endadresse beim Lesen merken
cb3a   a5  6f        lda   $6f         Adresse Lo
cb3c   85  a5        sta   $a5         als Lesezeiger Lo setzen
cb3e   a5  70        lda   $70         Adresse Hi
cb40   85  a6        sta   $a6         als Lesezeiger Hi setzen
cb42   4c  43  d4    jmp   $d443       Bytes holen und auf Bus ausgeben
cb45   20  eb  d0    jsr   $d0eb       unbenutzten Lesekanal suchen
cb48   4c  3a  d4    jmp   $d43a       Byte zum Computer
cb4b   a9  31        lda   #$31        Nummer der Fehlermeldung
cb4d   4c  c8  c1    jmp   $c1c8       '31, SYNTAX ERROR' ausgeben
-------------------------------------------------------------------------
cb50                                   M-W-Befehl.
cb50   b9  06  02    lda   $0206,y     Bytewert aus Befehlsstring holen
cb53   91  6f        sta   ($6f),y     und an gegebener Adresse abspeichern
cb55   c8            iny               Zeiger auf nächstes Byte
cb56   cc  05  02    cpy   $0205       schon alle Bytes abgespeichert?
cb59   90  f5        bcc   $cb50       verzweige, wenn nein; weitermachen
cb5b   60            rts               Ende
-------------------------------------------------------------------------
cb5c                                   USER-Befehle (Steuerroutine).
cb5c   ac  01  02    ldy   $0201       zweites Zeichen aus Befehlsstring
cb5f   c0  30        cpy   #$30        gleich '0' (U0-Befehl)?
cb61   d0  09        bne   $cb6c       verzweige, wenn nein
cb63   4c  30  80    jmp   $8030       zur Sonderbehandlung der U0-Befehlstypen
cb66   ea  ...       nop               Hier stand ursprünglich die Behandlung
cb6b   ... ea        nop               des U0-Befehls im DOS der VC1541!!!
cb6c   20  72  cb    jsr   $cb72       Adresse setzen; USER-Befehl ausführen
cb6f   4c  94  c1    jmp   $c194       Ende; (Fehler-)Meldung bereitstellen
-------------------------------------------------------------------------
cb72                                   Adresse für entsprechenden USER-Befehl bereit- stellen und
                                       Befehl ausführen. Y muß dabei das zweite Zeichen des
                                       Befehls enthalten.
cb72   88            dey               ASCII-Code des zweiten Zeichens minus 1
cb73   98            tya               für weitere Bearbeitung
cb74   29  0f        and   #$0f        Zahlenwert des Bytes feststellen
cb76   0a            asl               mal 2 nehmen
cb77   a8            tay               und als Zeiger in Adresstabelle benutzen
cb78   b1  6b        lda   ($6b),y     Adresse Lo des USER-Befehls holen
```

```
cb7a   85  75       sta   $75         und setzen
cb7c   c8           iny               nächstes Byte
cb7d   b1  6b       lda   ($6b),y     Adresse Hi des USER-Befehls holen
cb7f   85  76       sta   $76         und ebenfalls setzen
cb81   4c  2d  aa   jmp   $aa2d       USER-Befehl ausführen...
--------------------------------------------------------------------------
cb84                                  Befehl '#'; Öffnen eines Direktzugriffskanals.
cb84   ad  8e  02   lda   $028e       Drivenummer des letzten Jobs
cb87   85  7f       sta   $7f         als aktuelle Nummer übernehmen
cb89   a5  83       lda   $83         Kanalnummer
cb8b   48           pha               merken
cb8c   20  3d  c6   jsr   $c63d       Drive bei Bedarf initialisieren
cb8f   68           pla               Kanalnummer zurückholen
cb90   85  83       sta   $83         und wieder abspeichern
cb92   ae  74  02   ldx   $0274       Länge des Befehlsstrings
cb95   ca           dex               mit 1 vergleichen, ob ein bestimmter
cb96   d0  0d       bne   $cba5       gewünscht wird
cb98   a9  01       lda   #$01        nur '#'-Befehl; Puffernummer egal
cb9a   20  e2  d1   jsr   $d1e2       freien Kanal und Puffer belegen
cb9d   4c  f1  cb   jmp   $cbf1       Parameter setzen; Ende
cba0   a9  70       lda   #$70        Nummer für Fehlermeldung
cba2   4c  c8  c1   jmp   $c1c8       '70, NO CHANNEL' ausgeben; Ende
cba5   a0  01       ldy   #$01        Zeiger auf Puffernummer setzen und
cba7   20  7c  cc   jsr   $cc7c       diese aus dem Befehlsstring holen
cbaa   ae  85  02   ldx   $0285       Puffernummer nach X und
cbad   e0  05       cpx   #$05        mit dem maximal möglichen Wert vergleichen
cbaf   b0  ef       bcs   $cba0       verzweige, wenn größer; kein Puffer frei
cbb1   a9  00       lda   #$00        Masken für die Puffernummer
cbb3   85  6f       sta   $6f         löschen
cbb5   85  70       sta   $70         löschen
cbb7   38           sec               Carrybit als Maskenbit setzen
cbb8   26  6f       rol   $6f         Setzen des entsprechendn Bitmusters
cbba   26  70       rol   $70         für jeden Puffer, um eine Vergleichsmaske
cbbc   ca           dex               aufzubauen
cbbd   10  f9       bpl   $cbb8       verzweige, für nächsten Puffer
cbbf   a5  6f       lda   $6f         Maske für Laufwerk 0
cbc1   2d  4f  02   and   $024f       mit Pufferbelegung vergleichen
cbc4   d0  da       bne   $cba0       zur Fehlerroutine, wenn Puffer belegt
cbc6   a5  70       lda   $70         Maske für Laufwerk 1
cbc8   2d  50  02   and   $0250       mit Pufferbelegung vergleichen
cbcb   d0  d3       bne   $cba0       zur Fehlerroutine, wenn Puffer belegt
cbcd   a5  6f       lda   $6f         Maske für Pufferbelegung (Drive 0)
cbcf   0d  4f  02   ora   $024f       Belegung in Tabelle eintragen
cbd2   8d  4f  02   sta   $024f       und Tabelle abspeichern
cbd5   a5  70       lda   $70         Maske für Pufferbelegung (Drive 1)
cbd7   0d  50  02   ora   $0250       Belegung in Tabelle eintragen
cbda   8d  50  02   sta   $0250       und Tabelle abspeichern
cbdd   a9  00       lda   #$00        Standardkanalwert laden
cbdf   20  e2  d1   jsr   $d1e2       und unbenutzten Kanal suchen
cbe2   a6  82       ldx   $82         aktuelle Kanalnummer
cbe4   ad  85  02   lda   $0285       Puffernummr für Kanal
cbe7   95  a7       sta   $a7,x       in Kanalbelegung eintragen
cbe9   aa           tax               Puffernummer als Index
cbea   a5  7f       lda   $7f         aktuelle Drivenummer holen
cbec   95  00       sta   $00,x       und in Jobspeicher für entsprechenden
cbee   9d  5b  02   sta   $025b,x     Puffer eintragen
cbf1   a6  83       ldx   $83         Sekundäradresse
cbf3   bd  2b  02   lda   $022b,x     Kanalstatus aus Tabelle holen
```

```
cbf6    09  40       ora   #$40        Bit für Schreiben setzen
cbf8    9d  2b  02    sta   $022b,x     und als neuen Status abspeichern
cbfb    a4  82       ldy   $82         Kanalnummer als Index
cbfd    a9  ff       lda   #$ff        Kennzeichen für Puffer aktiv
cbff    99  44  02    sta   $0244,y     in Puffertabelle
cc02    a9  89       lda   #$89        Flag für Schreib-/Lesebetrieb
cc04    99  f2  00    sta   $00f2,y     für Kanal setzen
cc07    b9  a7  00    lda   $00a7,y     zugehörige Puffernummer holen
cc0a    99  3e  02    sta   $023e,y     und in Kanaltabelle eintragen
cc0d    0a           asl               Puffernummer mal 2
cc0e    aa           tax               als Index setzen
cc0f    a9  01       lda   #$01        Wert für Pufferzeiger auf zweites Byte
cc11    95  99       sta   $99,x       in Puffer setzen
cc13    a9  0e       lda   #$0e        Code für Direktzugriff
cc15    99  ec  00    sta   $00ec,y     als Filetyp für Bearbeitung setzen
cc18    4c  94  c1    jmp   $c194       Ende; Diskstatus bereitstellen
-------------------------------------------------------------------------
cc1b                                   BLOCK-Befehle (Steuerroutine)
cc1b    a0  00       ldy   #$00        Index-Register
cc1d    a2  00       ldx   #$00        vorbesetzen
cc1f    a9  2d       lda   #$2d        ASCII-Coce für '-'.
cc21    20  68  c2    jsr   $c268       Zeichen in Befehlsstring suchen
cc24    d0  0a       bne   $cc30       verzweige, wenn gefunden
cc26    a9  31       lda   #$31        Nummer der Fehlermeldung
cc28    4c  c8  c1    jmp   $c1c8       '31, SYNTAX ERROR' ausgeben; Ende
cc2b    a9  30       lda   #$30        Nummer der Fehlermeldung
cc2d    4c  c8  c1    jmp   $c1c8       '30, SYNTAX ERROR' ausgeben
cc30    8a           txa               Flag für 'Komma gefunden' testen
cc31    d0  f8       bne   $cc2b       verzweige zur Fehlermeldung, wenn Komma
cc33    a2  05       ldx   #$05        Zeiger in Zeichentabelle für BLOCK-Befehle
cc35    b9  00  02    lda   $0200,y     3. Zeichen aus Befehlsstring holen
cc38    dd  5d  cc    cmp   $cc5d,x     und mit Tabelle (P E W R F A) vergleichen
cc3b    f0  05       beq   $cc42       verzweige, wenn Zeichen gefunden
cc3d    ca           dex               Zeiger auf nächstes Zeichen in Tabelle
cc3e    10  f8       bpl   $cc38       weitersuchen, wenn noch Zeichen vorhanden
cc40    30  e4       bmi   $cc26       unbedingter Sprung; Befehl nicht erkannt
cc42    8a           txa               Code des Befehlszeichens nach A
cc43    09  80       ora   #$80        Bit 7 zeigt aktuellen Befehl an
cc45    8d  2a  02    sta   $022a       Befehlsnummer für Ausführung abspeichern
cc48    20  6f  cc    jsr   $cc6f       Blockparameter holen und prüfen
cc4b    ad  2a  02    lda   $022a       Befehlsnummer zurückholen
cc4e    0a           asl               mal 2
cc4f    aa           tax               und als Index in Tabelle
cc50    bd  64  cc    lda   $cc64,x     Befehlsadresse Hi aus Tabelle holen
cc53    85  70       sta   $70         und setzen
cc55    bd  63  cc    lda   $cc63,x     Befehlsadresse Lo aus Tabelle holen
cc58    85  6f       sta   $6f         und setzen
cc5a    6c  6f  00    jmp   ($006f)     Befehl ausführen; Ende
-------------------------------------------------------------------------
cc5d 41 46 52 57 45 50                 Bytes der Namen der BLOCK-Befehle 'A F R W E P'
-------------------------------------------------------------------------
cc63                                   Adressen der BLOCK-Befehle
cc63 03 cd                             $cd03 B-A
cc65 f5 cc                             $ccf5 B-F
cc67 56 cd                             $cd56 B-R
cc69 73 cd                             $cd73 B-II
cc6b a3 cd                             $cda3 B-E
cc6d bd cd                             $cdbd B-P
```

```
------------------------------------------------------------------------
cc6f                                    Parameter der BLOCK-Befehle holen und auf Syntax prüfen.
cc6f    a0  00      ldy   #$00         Index-Register
cc71    a2  00      ldx   #$00         zurücksetzen
cc73    a9  3a      lda   #$3a         ASCII-Code für ':'
cc75    20  68  c2  jsr   $c268        Zeichen in Befehlsstring suchen
cc78    d0  02      bne   $cc7c        verzweige, wenn gefunden
cc7a    a0  03      ldy   #$03         Zeiger auf 4. Zeichen in Befehlsstring
cc7c    b9  00  02  lda   $0200,y      Zeichen an dieser Position holen
cc7f    c9  20      cmp   #$20         und mit ' ' (SPACE) vergleichen
cc81    f0  08      beq   $cc8b        verzweige, wenn Zeichen gleich' '
cc83    c9  1d      cmp   #$1d         vergleiche Zeichen mit 'CURSOR LEFT'
cc85    f0  04      beq   $cc8b        verzweige bei Übereinstimmung
cc87    c9  2c      cmp   #$2c         vergleiche mit ',' (KOMMA)
cc89    d0  07      bne   $cc92        verzweige, wenn Zeichen kein ','
cc8b    c8          iny                Zeiger in Befehlsstring erhöhen
cc8c    cc  74  02  cpy   $0274        schon maximale Länge erreicht?
cc8f    90  eb      bcc   $cc7c        verzweige, wenn nein
cc91    60          rts                Ende
cc92    20  a1  cc  jsr   $cca1        Parameter des Befehlsstrings konvertieren
cc95    ee  77  02  inc   $0277        Zeiger in Befehlsstring plus 1
cc98    ac  79  02  ldy   $0279        Zeiger auf zweiten Teil des Befehlsstrings?
cc9b    e0  04      cpx   #$04         Maximalzahl der erte erreicht?
cc9d    90  ec      bcc   $cc8b        verzweige, wenn nein; weitersuchen
cc9f    b0  8a      bcs   $cc2b        unbedingter Sprung; Fehlerausgabe
------------------------------------------------------------------------
cca1                                    Konvertiert die ASCII-Werte aus dem INPUT. Puffer in HEX-
                                        Werte und legt diese in den Spur. und Sektornummertabellen
                                        ab.
cca1    a9  00      lda   #$00         Rechenbereich
cca3    85  6f      sta   $6f          für die
cca5    85  70      sta   $70          Parameterumwandlung
cca7    85  72      sta   $72          löschen
cca9    a2  ff      ldx   #$ff         Index bereitstellen
ccab    b9  00  02  lda   $0200,y      Zeichen aus Befehlsstring holen
ccae    c9  40      cmp   #$40         Test auf Ziffer
ccb0    b0  18      bcs   $ccca        verzweige, wenn keine Ziffer
ccb2    c9  30      cmp   #$30         Test auf Steuerzeichen
ccb4    90  14      bcc   $ccca        verzweige, wenn Steuerzeichen
ccb6    29  0f      and   #$0f         sonst Zahlenwert des Bytes feststellen
ccb8    48          pha                und merken
ccb9    a5  70      lda   $70          Werte im Rechenbereich
ccbb    85  71      sta   $71          um eine Stelle
ccbd    a5  6f      lda   $6f          weiter nach rechts
ccbf    85  70      sta   $70          verschieben
ccc1    68          pla                Zahlenwert zurückholen
ccc2    85  6f      sta   $6f          und abspeichern
ccc4    c8          iny                Zeiger auf nächstes Zeichen
ccc5    cc  74  02  cpy   $0274        schon alle Zeichen erfaßt?
ccc8    90  e1      bcc   $ccab        verzweige, wenn nein
ccca    8c  79  02  sty   $0279        Zeiger abspeichern
cccd    18          clc                errechnete Hex.erte
ccce    a9  00      lda   #$00         zu einem
ccd0    e8          inx                Byte
ccd1    e0  03      cpx   #$03         zusammenfassen
ccd3    b0  0f      bcs   $cce4        und im Rechenbereich
```

```
ccd5   b4  6f       ldy   $6f,x       abspeichern
ccd7   88           dey               nächster Wert
ccd8   30  f6       bmi   $ccd0       und weitermachen
ccda   7d  f2  cc    adc   $ccf2,x     Wert aus Dezimaltabelle addieren
ccdd   90  f8       bcc   $ccd7       verzweige, wenn kein Überlauf
ccdf   18           clc               Carry zur weiteren Bearbeitung wieder löschen
cce0   e6  72       inc   $72         nächsthöheres Byte bei Überlauf erhöhen
cce2   d0  f3       bne   $ccd7       verzweige, wenn noch kein Nulldurchgang
cce4   48           pha               Summe (Sektornummer) merken
cce5   ae  77  02    ldx   $0277       Zeiger in Befehlsstring zurückholen
cce8   a5  72       lda   $72         errechnete Tracknummer
ccea   9d  80  02    sta   $0280,x     in Tabelle ablegen
cced   68           pla               Summe (Sektornummer) zurückholen
ccee   9d  85  02    sta   $0285,x     und ebenfalls in Tabelle ablegen
ccf1   60           rts               Ende
-------------------------------------------------------------------------
ccf2 01 0a 64                         Dezimal-Umwandlungstabelle (1, 10, 100)
-------------------------------------------------------------------------
ccf5                                  B-F-Befehl (BLOCK-FREE).
ccf5   20  f5  cd    jsr   $cdf5       Track- und Sektorparameter holen
ccf8   20  5f  ef    jsr   $ef5f       BAM entspreched ändern; Dirty-Flag setzen
ccfb   4c  94  c1    jmp   $c194       Diskstatus bereitstellen; Erde
-------------------------------------------------------------------------
ccfe                                  Programmrest
ccfe   a9  01       lda   #$01        wird in der 1570/71
cd00   8d  f9  02    sta   $02f9       nicht mehr verwendet.
-------------------------------------------------------------------------
cd03                                  B-A-Befehl (BLOCK-ALLOCATE).
cd03   20  f5  cd    jsr   $cdf5       Track- und Sektorparameter holen
cd06   a5  81       lda   $81         aktuelle Sektornummer
cd08   48           pha               merken
cd09   20  fa  f1    jsr   $f1fa       nächsten freien Sektor suchen
cd0c   f0  0b       beq   $cd19       verzweige, wenn kein freier Sektor mehr
cd0e   68           pla               aktuelle Sektornummer zurückholen
cd0f   c5  81       cmp   $81         mit neuer Sektornummer vergleichen
cd11   d0  19       bne   $cd2c       verzweige, wenn ungleich
cd13   20  90  ef    jsr   $ef90       neuen Block in BAM belegen
cd16   4c  94  c1    jmp   $c194       fertig; Diskstatus bereitstellen
cd19   68           pla               aktuelle Sektornummer zurückholen
cd1a   a9  00       lda   #$00        Sektor 0 neu setzen
cd1c   85  81       sta   $81         und als aktuellen Sektor nehmen
cd1e   e6  80       inc   $80         Tracknummer erhöhen
cd20   a5  80       lda   $80         Tracknummer (neuer Wert)
*1 cd22 cd  ac  02    cmp   $02ac       mit maximaler Tracknummer vergleichen (36,72)
*0 cd22 cd  d7  fe    cmp   $fed7       mit maximaler Tracknummer vergleichen (36)
cd25   b0  0a       bcs   $cd31       verzweige, wenn größer/gleich
cd27   20  fa  f1    jsr   $f1fa       erneut einen freien Sektor suchen
cd2a   f0  ee       beq   $cd1a       und weitermachen, bis gefunden oder Fehler
cd2c   a9  65       lda   #$65        Nummer der Fehlermeldung
cd2e   20  45  e6    jsr   $e645       '65, NO BLOCK' ausgeben; Ende
cd31   a9  65       lda   #$65        Nummer der Fehlermeldung
cd33   20  c8  c1    jsr   $c1c8       '65, NO BLOCK' ausgeben
-------------------------------------------------------------------------
cd36                                  Unterroutine des B-R-Befehls zum Testen der Parameter und
                                      zum Lesen des Blocks von der Diskette.
cd36   20  f2  cd    jsr   $cdf2       Track- und Sektornummer setzen
cd39   4c  60  d4    jmp   $d460       Block von der Diskette lesen
```

```
------------------------------------------------------------------------
cd3c                                    Unterroutine des B-R-Befehls zum Holen eines Bytes aus dem
                                        aktuellen Puffer in den Akku.
cd3c   20  2f  d1    jsr   $d12f        Zeiger in den aktuellen Puffer setzen
cd3f   a1  99        lda   ($99,x)      Byte aus dem Puffer holen
cd41   60            rts                Ende
------------------------------------------------------------------------
cd42                                    Unterroutine des B-R-Befehls zum Lesen eines Blocks von der
                                        Diskette.
cd42   20  36  cd    jsr   $cd36        Parameter testen; Block lesen
cd45   a9  00        lda   #$00         Lo-Wert für Pufferzeiger
cd47   20  c8  d4    jsr   $d4c8        Pufferzeiger setzen
cd4a   20  3c  cd    jsr   $cd3c        ein Byte aus dem Puffer holen
cd4d   99  44  02    sta   $0244,y      als Endekennzeichen merken
cd50   a9  89        lda   #$89         Schreib-/Leseflag setzen
cd52   99  f2  00    sta   $00f2,y      in Kanalstatustabelle eintragen
cd55   60            rts                Ende
------------------------------------------------------------------------
cd56                                    B-R-Befehl (BLOCK-READ).
cd56   20  42  cd    jsr   $cd42        Block lesen
cd59   20  ec  d3    jsr   $d3ec        Byte aus dem Puffer für Ausgabe bereitstellen
cd5c   4c  94  c1    jmp   $c194        Diskstatus bereitstellen
------------------------------------------------------------------------
cd5f                                    U1-Befehl (BLOCK-READ-Ersatz).
cd5f   20  6f  cc    jsr   $cc6f        Parameter für den Befehl holen
cd62   20  42  cd    jsr   $cd42        Block lesen
cd65   b9  44  02    lda   $0244,y      $0244,y Zeiger auf Ende der Daten
cd68   99  3e  02    sta   $023e,y      als aktuelles Byte speichern
cd6b   a9  ff        lda   #$ff         Wert $FF (255)
cd6d   99  44  02    sta   $0244,y      als neues Endekennzeichen merken
cd70   4c  94  c1    jmp   $c194        Diskstatus bereitstellen
------------------------------------------------------------------------
cd73                                    B-W-Befehl (BLOCK-WRITE).
cd73   20  f2  cd    jsr   $cdf2        Kanal zum Schreiben öffnen
cd76   20  e8  d4    jsr   $d4e8        Pufferzeiger setzen
cd79   a8            tay                Pufferzeiger nach Y
cd7a   88            dey                minus 1
cd7b   c9  02        cmp   #$02         Pufferzeiger Lo mit 2 vergleichen
cd7d   b0  02        bcs   $cd81        verzweige, wenn größer/gleich
cd7f   a0  01        ldy   #$01         Index auf 1 setzen
cd81   a9  00        lda   #$00         Wert für Pufferzeiger Lo
cd83   20  c8  d4    jsr   $d4c8        Pufferzeiger entsprechend setzen
cd86   98            tya                A=1
cd87   20  f1  cf    jsr   $cff1        Inhalt von A in den Puffer schreiben
cd8a   8a            txa                Puffernummer mal 2
cd8b   48            pha                merken
cd8c   20  64  d4    jsr   $d464        Block auf Diskette schreiben
cd8f   68            pla                Puffernummer mal 2 zurückholen
cd90   aa            tax                und wieder nach X
cd91   20  ae  ff    jsr   $ffae        Byte aus dem Puffer ins Ausgaberegister
cd94   4c  94  c1    jmp   $c194        Diskstatus bereitstellen
------------------------------------------------------------------------
cd97                                    U2-Befehl (BLOCK-WRITE-Ersatz).
cd97   20  6f  cc    jsr   $cc6f        Blockparameter holen
cd9a   20  f2  cd    jsr   $cdf2        Parameter setzen; Kanal öffnen
cd9d   20  64  d4    jsr   $d464        Block auf die Diskette schreiben
cda0   4c  94  c1    jmp   $c194        Diskstatus bereitstellen
```

```
------------------------------------------------------------------------
cda3                                    B-E-Befehl (BLOCK-EXECUTE).
cda3    20  58  f2    jsr   $f258       (RTS)
cda6    20  36  cd    jsr   $cd36       Block in den Puffer lesen
cda9    a9  00        lda   #$00        Pufferadresse Lo auf 0
cdab    85  6f        sta   $6f         setzen
cdad    a6  f9        ldx   $f9         Puffernummer in X
cdaf    bd  e0  fe    lda   $fee0,x     zugehörige Adresse Hi holen
cdb2    85  70        sta   $70         und ebenfalls setzen
cdb4    20  ba  cd    jsr   $cdba       Programm im Puffer ausführen
cdb7    4c  94  c1    jmp   $c194       Diskstatus bereitstellen
------------------------------------------------------------------------
cdba                                    Ausführen eines Programms im Puffer bei B-E.
cdba    6c  6f  00    jmp   ($006f)     Sprung in den Puffer
------------------------------------------------------------------------
cdbd                                    B-P-Befehl (BLO:K'":;-::
cdbd    20  d2  cd    jsr   $cdd2       Kanal öffnen; P.Jf.e-r,..rr.=.- - c c-
cdc0    a5  f9        lda   $f9         Puffernummer
cdc2    0a            asl               mal 2
cdc3    aa            tax               und als Index rcc- (
cdc4    ad  86  02    lda   $0286       neuen ert ces :.."e-::;c-s
cdc7    95  99        sta   $99,x       übernehC1e"1
cdc9    20  2f  d1    jsr   $d12f       Puffer- und Kar,ö.-...rr.er re.er
cdcc    20  ee  d3    jsr   $d3ee       Byte aus dem :ffer hclen
cdcf    4c  94  c1    jmp   $c194       Diskstatus bereitstellen
------------------------------------------------------------------------
cdd2                                    Kanal öffnen und Puffer belegen; bei Fehler Ausgabe von
                                        '70, NO CHANNEL'.
cdd2    a6  d3        ldx   $d3         Zeiger in Befehlsstring
cdd4    e6  d3        inc   $d3         plus 1
cdd6    bd  85  02    lda   $0285,x     zugehörige Kanalnummer aus Tabelle holen
cdd9    a8            tay               nach Y
cdda    88            dey               minus 1
cddb    88            dey               minus 1
cddc    c0  0c        cpy   #$0c        war Kanalnummer gröBer/gleich 14?
cdde    90  05        bcc   $cde5       verzweige, wenn nein
cde0    a9  70        lda   #$70        Nummer der Fehlermeldung
cde2    4c  c8  c1    jmp   $c1c8       '70, NO CHANNEL' ausgeben; Ende
cde5    85  83        sta   $83         Sekundäradresse abspeichern
cde7    20  eb  d0    jsr   $d0eb       zugehörigen Kanal prüfen
cdea    b0  f4        bcs   $cde0       verzweige, Fehler, wenn schon geöffnet
cdec    20  93  df    jsr   $df93       Puffernummer holen
cdef    85  f9        sta   $f9         und setzen
cdf1    60            rts               Ende
------------------------------------------------------------------------
cdf2                                    Testen aller Parameter auf legalen Block und belegten
                                        Puffer. Sind diese Werte in Ordnung, so werden sie zum
                                        Lesen bereitgestellt-
cdf2    20  d2  cd    jsr   $cdd2       Kanal öffnen und Puffernummer holen
cdf5    a6  d3        ldx   $d3         Zeiger in Befehlsstring
cdf7    bd  85  02    lda   $0285,x     Kanalnummer aus Tabelle holen
cdfa    29  01        and   #$01        gibt immer 0
cdfc    85  7f        sta   $7f         als aktuelle Drivenummer übernehmen
cdfe    bd  87  02    lda   $0287,x     zugehörige Sektornummer aus Tabelle
ce01    85  81        sta   $81         übernehmen
ce03    bd  86  02    lda   $0286,x     zugehörige Tracknummer aus Tabelle
ce06    85  80        sta   $80         übernehmen
```

```
ce08   20  5f  d5    jsr   $d55f       Track- und Sektorwerte in Ordnung?
ce0b   4c  00  c1    jmp   $c100       LED am Laufwerk einschalten; Ende
------------------------------------------------------------------------
ce0e                                   Suchen eines Datenblocks in einer relativen Datei.
ce0e   20  2c  ce    jsr   $ce2c       Gesamtzahl der Bytes berechnen
ce11   20  6e  ce    jsr   $ce6e       geteilt durch 254 für Recordnummer
ce14   a5  90        lda   $90         Rest der Division ist gleich
ce16   85  d7        sta   $d7         Zeiger in Datenblock
ce18   20  71  ce    jsr   $ce71       geteilt durch 120 für Side-Sektor
ce1b   e6  d7        inc   $d7         Zeiger in Datenblock plus 2, da der Linker am
ce1d   e6  d7        inc   $d7         Anfang jedes Sektors übergangen werden muß
ce1f   a5  8b        lda   $8b         errechnete Side-Sektor-Nummer
ce21   85  d5        sta   $d5         übernehmen
ce23   a5  90        lda   $90         Rest der Division
ce25   0a            asl               mal 2
ce26   18            clc               und
ce27   69  10        adc   #$10        plus 16
ce29   85  d6        sta   $d6         ist Zeiger in Side-Sektor
ce2b   60            rts               Ende
------------------------------------------------------------------------
ce2c                                   Errechnen der Position eines Records.
ce2c   20  d9  ce    jsr   $ced9       Löschen des Ergebnisspeichers
ce2f   85  92        sta   $92         0 nach S92
ce31   a6  82        ldx   $82         aktuelle Kanalnummer?
ce33   b5  b5        lda   $b5,x       Recordnummer Lo
ce35   85  90        sta   $90         in Rechenregister
ce37   b5  bb        lda   $bb,x       Recordnummer Hi
ce39   85  91        sta   $91         in Rechenregister
ce3b   d0  04        bne   $ce41       verzweige, wenn ungleich 0
ce3d   a5  90        lda   $90         Recordnummer Lo gleich O?
ce3f   f0  0b        beq   $ce4c       verzweige, wenn ja
ce41   a5  90        lda   $90         Recordnummer Lo
ce43   38            sec               Subtraktion vorbereiten und
ce44   e9  01        sbc   #$01        1 von Recordnummer Lo abziehen
ce46   85  90        sta   $90         ert wieder setzen
ce48   b0  02        bcs   $ce4c       verzweige, wenn Ergebnis größer 0
ce4a   c6  91        dec   $91         sonst Recordnummer Hi minus 1
ce4c   b5  c7        lda   $c7,x       Recordlänge holen
ce4e   85  6f        sta   $6f         und in Rechenregister
ce50   46  6f        lsr   $6f         Test auf ungeraden ert der Recordlänge
ce52   90  03        bcc   $ce57       verzweige, wenn ert gerade
ce54   20  ed  ce    jsr   $ceed       Ergebnisse plus Registerwerte in Register
ce57   20  e5  ce    jsr   $cee5       Registerinhalt mal 2
ce5a   a5  6f        lda   $6f         Resultat schon erhalten?
ce5c   d0  f2        bne   $ce50       weitermachen, wenn nein
ce5e   a5  d4        lda   $d4         Zeiger auf Beginn des Records
ce60   18            clc               zu dem vorhandenen
ce61   65  8b        adc   $8b         Registerinhalt addieren und
ce63   85  8b        sta   $8b         Ergebnis abspeichern
ce65   90  06        bcc   $ce6d       verzweige, wenn kein Oberlauf auftrat
ce67   e6  8c        inc   $8c         sonst nächsthöhere Speicherstelle plus 1
ce69   d0  02        bne   $ce6d       verzweige, wenn auch hier kein Oberlauf
ce6b   e6  8d        inc   $8d         sonst wiederum höhere Speicherstelle plus 1
ce6d   60            rts               und fertig
```

```
--------------------------------------------------------------------------
ce6e                                    Divisionsroutine. Bei Einsprung ab $ce6e erfolgt die
                                        Divison durch 254; bei Einsprung ab $ce71 erfolgt die
                                        Division durch 120. Am Ende der Berechnung steht der
                                        Quotient in $8b,8c,8d und der Rest in $90.
ce6e   a9  fe       lda   #$fe          Wert für Division durch 254
ce70   2c           .byte $2c           nächsten Berehl überspringen

ce71   a9  78       lda   #$78          Wert für Division durch 120
ce73   85  6f       sta   $6f           als Divisor speichern
ce75   a2  03       ldx   #$03          Rechenregister
ce77   b5  8f       lda   $8f,x         für
ce79   48           pha                 Division
ce7a   b5  8a       lda   $8a,x         vorbereiten, indem
ce7c   95  8f       sta   $8f,x         die
ce7e   68           pla                 Registerinhalte
ce7f   95  8a       sta   $8a,x         ausgetauscht
ce81   ca           dex                 werden
ce82   d0  f3       bne   $ce77         jeweils 3 Bytes tauschen
ce84   20  d9  ce   jsr   $ced9         Ergebnisspeicher löschen
ce87   a2  00       ldx   #$00          Registerinhalte
ce89   b5  90       lda   $90,x         nach unten
ce8b   95  8f       sta   $8f,x         verschieben
ce8d   e8           inx                 und zwar
ce8e   e0  04       cpx   #$04          alle 5 Bytes
ce90   90  f7       bcc   $ce89         um ein Byte ($90-94) nach ($8f-93)
ce92   a9  00       lda   #$00          Division vorbereiten
ce94   85  92       sta   $92           Register löschen
ce96   24  6f       bit   $6f           Bit 7 testen
ce98   30  09       bmi   $cea3         verzweige, wenn gesetzt
ce9a   06  8f       asl   $8f           mal 2
ce9c   08           php                 Status merken
ce9d   46  8f       lsr   $8f           geteilt durch 2
ce9f   28           plp                 Bit 7 jetzt in Carry
cea0   20  e6  ce   jsr   $cee6         Register $90,91,92 mal 2 plus Carry
cea3   20  ed  ce   jsr   $ceed         Register $8b,8c,8d plus $90,91,92
cea6   20  e5  ce   jsr   $cee5         Register $90,91,92 mal 2
cea9   24  6f       bit   $6f           Bit 7 testen
ceab   30  03       bmi   $ceb0         verzweige, wenn gesetzt
cead   20  e2  ce   jsr   $cee2         Register $90,91,92 mal 4
ceb0   a5  8f       lda   $8f           Übertrag
ceb2   18           clc                 zu Rechenergebnis
ceb3   65  90       adc   $90           addieren
ceb5   85  90       sta   $90           und abspeichern
ceb7   90  06       bcc   $cebf         verzweige, wenn kein Überlauf
ceb9   e6  91       inc   $91           sonst nächsthöheres Register plus 1
cebb   d0  02       bne   $cebf         verzweige, wenn kein Überlauf
cebd   e6  92       inc   $92           sonst wieder höheres Register plus 1
cebf   a5  92       lda   $92           schon alle Verschiebungen
cec1   05  91       ora   $91           für Division durchgeführt?
cec3   d0  c2       bne   $ce87         weitermachen, wenn nein
cec5   a5  90       lda   $90           sonst
cec7   38           sec                 Ergebnisregister
cec8   e5  6f       sbc   $6f           richtigstellen
ceca   90  0c       bcc   $ced8         verzweige, bei Unterlauf
cecc   e6  8b       inc   $8b           nächsthöheres Register plus 1
cece   d0  06       bne   $ced6         verzweige, wenn noch nicht 0
```

```
ced0    e6  8c       inc   $8c         nächsthöheres Register plus 1
ced2    d0  02       bne   $ced6       verzweige, wenn noch nicht 0
ced4    e6  8d       inc   $8d         $8d nächsthöheres Register wiederum plus 1
ced6    85  90       sta   $90         und Rest der Division merken
ced8    60           rts               Ende
--------------------------------------------------------------------------
ced9                                   Ergebnisspeicher in $8b,8c,8d löschen, d.h. mit 0
                                       auffüllen.
ced9    a9  00       lda   #$00        Null
cedb    85  8b       sta   $8b         als Leerinhalt
cedd    85  8c       sta   $8c         des Ergebnisregisters
cedf    85  8d       sta   $8d         setzen
cee1    60           rts               Ende
--------------------------------------------------------------------------
cee2                                   Rechenregister $90,91,92 mit 4 multiplizieren
cee2    20  e5  ce   jsr   $cee5       Rechenregister mal 2; danach...
--------------------------------------------------------------------------
cee5                                   Rechenregister $90,91,92 mit 2 multiplizieren
cee5    18           clc               Übertrag löschen
cee6    26  90       rol   $90         Register durch einmal
cee8    26  91       rol   $91         Linksverschieben mit
ceea    26  92       rol   $92         2 multiplizieren
ceec    60           rts               Ende
--------------------------------------------------------------------------
ceed                                   Ergebnisspeicher $8b,8c,8d zum Rechenregister $90,91,92
                                       addieren.
ceed    18           clc               Übertrag löschen
ceee    a2  fd       ldx   #$fd        minus 3 als Index
cef0    b5  8e       lda   $8e,x       und alle
cef2    75  93       adc   $93,x       drei Registerplätze
cef4    95  8e       sta   $8e,x       addieren und
cef6    e8           inx               im Ergebnisspeicher
cef7    d0  f7       bne   $cef0       ablegen
cef9    60           rts               Ende
--------------------------------------------------------------------------
cefa                                   Herstellen der Zwischenspeichertabelle.
cefa    a2  00       ldx   #$00        0 als Ausgangswert
cefc    8a           txa               in Akku
cefd    95  fa       sta   $fa,x       Wert in Zwischenspeicher
ceff    e8           inx               Index und Speicherwert plus 1
cf00    e0  04       cpx   #$04        schon Endwert erreicht?
cf02    d0  f8       bne   $cefc       weitermachen, wenn nein
cf04    a9  06       lda   #$06        Wert 6
cf06    95  fa       sta   $fa,x       in $fe abspeichern
cf08    60           rts               Ende
--------------------------------------------------------------------------
cf09                                   Aktualisieren der Zwischenspeichertabelle.
cf09    a0  04       ldy   #$04        4 als Index in Tabelle
cf0b    a6  82       ldx   $82         aktuelle Kanalnummer
cf0d    b9  fa  00   lda   $00fa,y     Kanalnummer aus Tabelle holen
cf10    96  fa       stx   $fa,y       aktuelle Kanalnummer neu übernehmen
cf12    c5  82       cmp   $82         neue Kanalnummer mit Tabellenwert vergleichen
cf14    f0  07       beq   $cf1d       verzweige, wenn gleich; alles in Ordnung
cf16    88           dey               sonst Index setzen und
cf17    30  e1       bmi   $cefa       neue Tabelle erzeugen
cf19    aa           tax               Einsprung mit Kanalnummer in A
cf1a    4c  0d  cf   jmp   $cf0d       und Tabelle prüfen
cf1d    60           rts               Ende
```

```
------------------------------------------------------------------------
cf1e                                    Aktiven Puffer für Diskbetrieb setzen; ggf. neuen Puffer
                                        suchen
cf1e   20  09  cf    jsr   $cf09       Tabelle aktualisieren
cf21   20  b7  df    jsr   $dfb7       gewählter Kanal in Ordnung?
cf24   d0  46        bne   $cf6c       verzweige, wenn nein
cf26   20  d3  d1    jsr   $d1d3       Drivenummer setzen
cf29   20  8e  d2    jsr   $d28e       freien Puffer suchen
cf2c   30  48        bmi   $cf76       Fehler, werm nicht gefunden
cf2e   20  c2  df    jsr   $dfc2       neuen Puffer inaktiv setzen
cf31   a5  80        lda   $80         aktuelle Tracknummer
cf33   48            pha               merken
cf34   a5  81        lda   $81         aktuelle Sektornummer
cf36   48            pha               merken
cf37   a9  01        lda   #$01        Nummer des zu holenden Bytes
cf39   20  f6  d4    jsr   $d4f6       Sektornummer aus Puffer holen
cf3c   85  81        sta   $81         und als aktuellen Sektor setzen
cf3e   a9  00        lda   #$00        Nummer des zu holenden Bytes
cf40   20  f6  d4    jsr   $d4f6       Tracknummer aus Puffer holen
cf43   85  80        sta   $80         und als aktuellen Track setzen
cf45   f0  1f        beq   $cf66       verzweige, wenn 0 (letzter Block)
cf47   20  25  d1    jsr   $d125       Filetyp holen
cf4a   f0  0b        beq   $cf57       verzweige, wenn relative Datei
cf4c   20  ab  dd    jsr   $ddab       auf Befehlscode 'Schreiben' prüfen
cf4f   d0  06        bne   $cf57       verzweige, wenn kein Schreibjob
cf51   20  8c  cf    jsr   $cf8c       Puffer wechseln
cf54   4c  5d  cf    jmp   $cf5d       und weitermachen
cf57   20  8c  cf    jsr   $cf8c       Puffer wechseln
cf5a   20  57  de    jsr   $de57       Befehlscode 'Lesen' prüfen und an DC übergeben
cf5d   68            pla               Sektornummer zurückholen
cf5e   85  81        sta   $81         und wieder setzen
cf60   68            pla               Tracknummer zurückholen
cf61   85  80        sta   $80         und ebenfalls wieder setzen
cf63   4c  6f  cf    jmp   $cf6f       weitermachen
cf66   68            pla               Sektornummer zurückholen
cf67   85  81        sta   $81         und wieder setzen
cf69   68            pla               Tracknummer zurückholen
cf6a   85  80        sta   $80         und wieder setzen
cf6c   20  8c  cf    jsr   $cf8c       Puffer wechseln
cf6f   20  93  df    jsr   $df93       Puffernummer holen
cf72   aa            tax               und nach X
cf73   4c  99  d5    jmp   $d599       Ausführung des Jobs abwarten und prüfen
------------------------------------------------------------------------
cf76                                    '70, NO CHANNEL' ausgeben.
cf76   a9  70        lda   #$70        Nummer der Fehlermeldung
cf78   4c  c8  c1    jmp   $c1c8       '70, NO CHANNEL' ausgeben; Ende
------------------------------------------------------------------------
cf7b                                    Sucht nach freiem Puffer.
cf7b   20  09  cf    jsr   $cf09       Tabelle aktualisieren
cf7e   20  b7  df    jsr   $dfb7       gewählter Puffer frei?
cf81   d0  08        bne   $cf8b       verzweige, wenn ja; Ende
cf83   20  8e  d2    jsr   $d28e       anderen Puffer suchen
cf86   30  ee        bmi   $cf76       Fehler, wenn nicht gefunden
cf88   20  c2  df    jsr   $dfc2       gefundenen Puffer inaktiv setzen
cf8b   60            rts               Ende
```

```
------------------------------------------------------------------------
cf8c                                    Wechseln des Betriebszustandes eines Puffers von aktiv nach
                                        inaktiv und umgekehrt.
cf8c   a6  82       ldx   $82           ldx $82 aktuelle Kanalnummer
cf8e   b5  a7       lda   $a7,x         Pufferstatustabelle
cf90   49  80       eor   #$80          Pufferstatus wechseln
cf92   95  a7       sta   $a7,x         und wieder abspeichern
cf94   b5  ae       lda   $ae,x         zweite Statustabelle
cf96   49  80       eor   #$80          Status ebenfalls wechseln
cf98   95  ae       sta   $ae,x         und wieder abspeichern
cf9a   60           rts                 Ende
------------------------------------------------------------------------
cf9b                                    Schreiben eines Bytes über den internen Schreibkanal in
                                        einen Puffer.
cf9b   a2  12       ldx   #$12          18 (interner Schreibkanal)
cf9d   86  83       stx   $83           als aktuelle Sekundäradresse setzen
cf9f   20  07  d1   jsr   $d107         Schreibkanal suchen und eröffnen
cfa2   20  00  c1   jsr   $c100         LED am Laufwerk einschalten
cfa5   20  25  d1   jsr   $d125         Dateityp holen
cfa8   90  05       bcc   $cfaf         verzweige, wenn kein relatives File
cfaa   a9  20       lda   #$20          Kanalstatus in der
cfac   20  9d  dd   jsr   $dd9d         Tabelle umdrehen
cfaf   a5  83       lda   $83           aktuelle Sekundäradresse
cfb1   c9  0f       cmp   #$0f          gleich 15 (Kommandokanal)?
cfb3   f0  23       beq   $cfd8         verzweige, wenn ja
cfb5   d0  08       bne   $cfbf         unbedingter Sprung, wenn nein
cfb7   a5  84       lda   $84           Sekundäradresse
cfb9   29  8f       and   #$8f          isolieren und
cfbb   c9  0f       cmp   #$0f          ebenfalls mit 15 vergleichen
cfbd   b0  19       bcs   $cfd8         verzweige, wenn gröBer oder gleich 15
cfbf   20  25  d1   jsr   $d125         Dateityp holen
cfc2   b0  05       bcs   $cfc9         verzweige, wenn kein sequentielles File
cfc4   a5  85       lda   $85           aktuelles Datenbyte für Kanal
cfc6   4c  9d  d1   jmp   $d19d         in zugehörigen Puffer schreiben
cfc9   d0  03       bne   $cfce         verzweige, wenn USR-File
cfcb   4c  ab  e0   jmp   $e0ab         Datenbyte in relative Datei schreiben
cfce   a5  85       lda   $85           aktuelles Datenbyte
cfd0   20  f1  cf   jsr   $cff1         in Puffer schreiben
cfd3   a4  82       ldy   $82           aktuelle Kanalnummer in Y
cfd5   4c  ee  d3   jmp   $d3ee         nächstes Byte für Ausgabe bereitstellen
cfd8   a9  04       lda   #$04          Kanalnummer 4 für Kommandokanal
cfda   85  82       sta   $82           setzen
cfdc   20  e8  d4   jsr   $d4e8         Pufferzeiger in Befehlspuffer
cfdf   c9  2a       cmp   #$2a          holen und testen, ob Puffer voll
cfe1   f0  05       beq   $cfe8         verzweige, wenn ja
cfe3   a5  85       lda   $85           aktuelles Datenbyte
cfe5   20  f1  cf   jsr   $cff1         in Puffer schreiben
cfe8   a5  f8       lda   $f8           auf EOI testen
cfea   f0  01       beq   $cfed         verzweige, wenn kein EOI
cfec   60           rts                 Ende; alles fertig
cfed   ee  55  02   inc   $0255         weiterer Befehl ist auszuführen
cff0   60           rts                 Ende für diesen Durchgang
```

```
--------------------------------------------------------------------------
cff1                                    Datenbyte in den aktuellen Puffer schreiben.
cff1    48          pha                 Datenbyte merken
cff2    20  93  df    jsr   $df93       Puffernummer holen
cff5    10  06        bpl   $cffd       verzweige, wenn Puffer ok
cff7    68            pla               sonst Stack wiederherstellen
cff8    a9  61        lda   #$61        Numer für Fehlermeldung
cffa    4c  c8  c1    jmp   $c1c8       '61, FILE NOT OPEN' ausgeben; Ende
cffd    0a            asl               Puffernummer mal 2
cffe    aa            tax               als Index setzen
cfff    68            pla               Datenbyte zurückholen
d000    81  99        sta   ($99,x)     und in den Puffer schreiben
d002    f6  99        inc   $99,x       Pufferzeiger erhöhen
d004    60            rts               Ende
--------------------------------------------------------------------------
d005                                    INITIALIZE-Befehl.
d005    20  d1  c1    jsr   $c1d1       Parameter für den Befehl prüfen
d008    20  42  d0    jsr   $d042       Diskette initialisieren (BAM laden)
d00b    4c  94  c1    jmp   $c194       Diskstatus bereitstellen: Ende
--------------------------------------------------------------------------
d00e                                    Initialisieren des Laufwerks, dessen Nummer in $7f abgelegt
                                        ist (hier immer 0).
d00e    20  0f  f1    jsr   $f10f       BAM-Pointer holen
d011    a8            tay               als Index in Tabelle
d012    b6  a7        ldx   $a7,y       entsprechende Kanalnummer aus Tabelle holen
d014    e0  ff        cpx   #$ff        Puffer reserviert?
d016    d0  14        bne   $d02c       verzweige, wenn ja
d018    48            pha               Puffernummer merken
d019    20  8e  d2    jsr   $d28e       Puffer für BAM suchen
d01c    aa            tax               Nummer testen
d01d    10  05        bpl   $d024       verzweige, wenn Puffer gefunden
d01f    a9  70        lda   #$70        Nummer für Fehlermeldung
d021    20  48  e6    jsr   $e648       '70, NO CHANNEL' ausgeben
d024    68            pla               Puffernummer zurückholen
d025    a8            tay               und als Index nehmen
d026    8a            txa               neue Puffernummer in A
d027    09  80        ora   #$80        Puffer als belegt kennzeichnen
d029    99  a7  00    sta   $00a7,y     und in Tabelle abspeichern
d02c    8a            txa               Puffernummer wiederum in A
d02d    29  0f        and   #$0f        Nummer isolieren
d02f    85  f9        sta   $f9         und als aktuelle Puffernummer setzen
d031    a2  00        ldx   #$00        Sektornummer 0
d033    86  81        stx   $81         für Diskbetrieb setzen
d035    ae  85  fe    ldx   $fe85       Wert 18 (für BAM)
d038    86  80        stx   $80         als Tracknummer für Disk setzen
d03a    20  d3  d6    jsr   $d6d3       Parameter an DC für Jobcode zum Lesen
d03d    a9  b0        lda   #$b0        Befehlscode 'Suchen eines Sektors'
*1 d03f 4c  e5  a6    jmp   $a6e5       an DC und Job ausführen; Ende
*0 d03f 4c  8c  d5    jmp   $d58c       an DC und Job ausführen; Ende
--------------------------------------------------------------------------
d042                                    BAM lesen und im Speicher aktualisieren.
d042    20  d1  f0    jsr   $f0d1       Tracknummer der BAM löschen
d045    20  13  d3    jsr   $d313       Kanal belegen
d048    20  0e  d0    jsr   $d00e       Puffer belegen; Header suchen
d04b    a6  7f        ldx   $7f         aktuelle Drivenummer
d04d    a9  00        lda   #$00        'BAM dirty flag'
d04f    9d  51  02    sta   $0251,x     löschen
d052    8a            txa               Drivenummer in A
```

```
d053   0a           asl             2
d054   aa           tax             als Index
d055   a5  16       lda   $16       ID1; erstes Zeichen der ID
d057   95  12       sta   $12,x     übernehmen
d059   a5  17       lda   $17       ID2; zweites Zeichen der ID
d05b   95  13       sta   $13,x     ebenfalls übernehmen
*1 d05d 20  67  a6   jsr   $a667     Diskseite feststellen; Block lesen
*0 d05d 20  86  d5   jsr   $d586     Block von Diskette lesen
d060   a5  f9       lda   $f9       aktuelle Puffernummer
d062   0a           asl             mal 2
d063   aa           tax             als Index in Puffer
d064   a9  02       lda   #$02      Pufferzeiger Lo
d066   95  99       sta   $99,x     auf $02 setzen
d068   a1  99       lda   ($99,x)   Formatkennzeichen aus Puffer holen
d06a   a6  7f       ldx   $7f       aktuelle Drivenummer nach X
d06c   9d  01  01   sta   $0101,x   und Formatkennzeichen merken
d06f   a9  00       lda   #$00      Flags für Laufwerk setzen:
d071   4c  1d  aa   jmp   $aa1d     Drive inaktiv; kein Diskettenwechsel außerdem
d074   ea           nop             wird die Anzahl der freien Blöcke berechnet
-------------------------------------------------------------------------------
d075                                Anzahl der freien Blöcke auf der Diskette berechnen.
d075   20  3a  ef   jsr   $ef3a     Pufferadresse nach $6d/6e holen
d078   a0  04       ldy   #$04      Zeiger auf 4. Byte; Beginn der BAM
d07a   a9  00       lda   #$00      erster Additionswert
d07c   aa           tax             Hi-Byte der Blockanzahl in X
d07d   18           clc             Addition vorbereiten
d07e   71  6d       adc   ($6d),y   plus Anzahl der freien Blöcke eines Tracks
d080   90  01       bcc   $d083     verzweige, wenn kein Überlauf
d082   e8           inx             sonst Anzahl Hi ebenfalls erhöhen
d083   c8           iny             ndex auf nächsten
d084   c8           iny             Wert für die Addition
d085   c8           iny             einstellen;
d086   c8           iny             alle 4 Bytes ein neuer Wert
d087   c0  48       cpy   #$48      Indexwert für Track 18 erreicht?
d089   f0  f8       beq   $d083     verzweige, wenn ja; Track überspringen
d08b   c0  90       cpy   #$90      Indexwert für letzten Track erreicht?
d08d   d0  ee       bne   $d07d     weitermachen, wenn nein
d08f   48           pha             Lo-Byte der Anzahl merken
d090   8a           txa             Hi-Byte der Anzahl in A
d091   a6  7f       ldx   $7f       Drivenummer als Index
d093   9d  fc  02   sta   $02fc,x   und Hi-Byte abspeichern
d096   68           pla             Lo-Byte zurückholen
*1 d097 4c  51  a9   jmp   $a951     und abspeichern; zweite Diskseite addieren
*0 d097 9d  fa  02   sta   $02fa,x   und abspeichern
d09a   60           rts             unbenutzter Programmrest des 1541 DOS 2.6
-------------------------------------------------------------------------------
d09b                                Lesen eines Blocks und Übernehmen der Parameter
d09b   20  d0  d6   jsr   $d6d0     Parameter für Lesen an DC übergeben
d09e   20  c3  d0   jsr   $d0c3     Befehl (Block lesen) an OC übergeben
d0a1   20  99  d5   jsr   $d599     Ausführung des Jobs prüfen und abwarten
d0a4   20  37  d1   jsr   $d137     erstes Byte aus dem Puffer holen
d0a7   85  80       sta   $80       und als Tracknummer speichern
d0a9   20  37  d1   jsr   $d137     zweites Byte aus Puffer
d0ac   85  81       sta   $81       und als Sektornummer nehmen
d0ae   60           rts             Ende
```

```
-------------------------------------------------------------------------
d0af                                    Anhand der Linker des ersten Blocks wird auch der
                                        nachfolgende Block in einen Puffer geholt.
d0af    20  9b  d0    jsr   $d09b       Lesen eines Blocks; Holen der Linkbytes
d0b2    a5  80        lda   $80         Tracknummer gleich 0 (letzter Block)?
d0b4    d0  01        bne   $d0b7       weitermachen, wenn nein
d0b6    60            rts               sonst Ende
d0b7    20  1e  cf    jsr   $cf1e       auf Zweipufferbetrieb umschalten
d0ba    20  d0  d6    jsr   $d6d0       Parameter zum Lesen an DC übergeben
d0bd    20  c3  d0    jsr   $d0c3       Block in einen anderen Puffer lesen
d0c0    4c  1e  cf    jmp   $cf1e       Pufferbetrieb wieder auf ersten umschalten
-------------------------------------------------------------------------
d0c3                                    Einstieg für Job: Block lesen bei $d0c3; für Block
                                        schreiben muß bei $d0c7 eingesprungen werden.
d0c3    a9  80        lda   #$80        $80 Jobcode für Lesen
d0c5    d0  02        bne   $d0c9       unbedingter Sprung zur Ausführung

d0c7    a9  90        lda   #$90        $90 Jobcode für Schreiben
d0c9    8d  4d  02    sta   $024d       Jobcode in Jobspeicher
d0cc    20  93  df    jsr   $df93       Puffernummer für Job holen
d0cf    aa            tax               Nummer nach X
d0d0    20  06  d5    jsr   $d506       Werte für Job prüfen und an DC
d0d3    8a            txa               Puffernummer wieder nach A
d0d4    48            pha               und merken
d0d5    0a            asl               Puffernummer mal 2
d0d6    aa            tax               und nach X aLs Index
d0d7    a9  00        lda   #$00        Wert für Pufferzeiger Lo
d0d9    95  99        sta   $99,x       setzen
d0db    20  25  d1    jsr   $d125       Filetyp holen
d0de    c9  04        cmp   #$04        und mit sequentieLlem File vergleichen
d0e0    b0  06        bcs   $d0e8       verzweige, wenn kein sequentielles File
d0e2    f6  b5        inc   $b5,x       Recordnummer Lo
d0e4    d0  02        bne   $d0e8       verzweige, wenn ungleich Null
d0e6    f6  bb        inc   $bb,x       Recordnummer Hi
d0e8    68            pla               Puffernummer zurückholen
d0e9    aa            tax               und nach X
d0ea    60            rts               Ende
-------------------------------------------------------------------------
d0eb                                    Suchen und Eröffnen eines Kanals zum Lesen.
d0eb    a5  83        lda   $83         aktuelle Sekundäradresse
d0ed    c9  13        cmp   #$13        mit 19 (Maximum) vergleichen
d0ef    90  02        bcc   $d0f3       verzweige, wenn kleiner
d0f1    29  0f        and   #$0f        sonst Kanlanummer isolieren
d0f3    c9  0f        cmp   #$0f        und mit 15 (Kommandokanal) vergleichen
d0f5    d0  02        bne   $d0f9       verzweige, wenn ungleich 15
d0f7    a9  10        lda   #$10        16
d0f9    aa            tax               als Index nach X
d0fa    38            sec               Flag für Kanal belegt setzen
d0fb    bd  2b  02    lda   $022b,x     in Kanaltabelle suchen
d0fe    30  06        bmi   $d106       verzweige, wenn Kanal belegt; Ende mit SEC
d100    29  0f        and   #$0f        sonst Kanalnummer isolieren
d102    85  82        sta   $82         und als aktuelle Nummer speichern
d104    aa            tax               Nummer außerdem nach X
d105    18            clc               Flag für ok setzen
d106    60            rts               Ende
```

```
------------------------------------------------------------------------
d107                                    Suchen und Eröffnen eines Kanals zum Schreiben.
d107   a5  83       lda   $83          aktuelle Sekundäradresse
d109   c9  13       cmp   #$13         mit 19 (Maximum) vergleichen
d10b   90  02       bcc   $d10f        verzweige, wenn kleiner
d10d   29  0f       and   #$0f         auf 15 begrenzen
d10f   aa           tax                und als Index in Tabelle
d110   bd  2b  02    lda   $022b,x      Kanalstatus holen
d113   a8           tay                und nach Y
d114   0a           asl                mal 2
d115   90  0a       bcc   $d121        verzweige, wenn Bit 7=0 war
d117   30  0a       bmi   $d123        verzweige unbedingt
d119   98           tya                Kanalstatus zurückholen
d11a   29  0f       and   #$0f         und Kanalnummer isolieren
d11c   85  82       sta   $82          Nummer setzen
d11e   aa           tax                und außerdem nach X
d11f   18           clc                Flag für ok setzen
d120   60           rts                Ende
d121   30  f6       bmi   $d119        verzweige, wenn Schreib.fLesekanal
d123   38           sec                Flag für Fehler setzen; Kanal belegt
d124   60           rts                Ende
------------------------------------------------------------------------
d125                                    Aktuellen Filetyp holen und auf REL.File prüfen
d125   a6  82       ldx   $82          aktuelle Kanalnummer
d127   b5  ec       lda   $ec,x        Kanalfiletyp holen
d129   4a           lsr                Wert halbieren und
d12a   29  07       and   #$07         Bits 0 bis 2 isolieren
d12c   c9  04       cmp   #$04         vergleiche auf relatives File
d12e   60           rts                Ende mit gesetztem Status
------------------------------------------------------------------------
d12f                                    Puffer. und Kanalnummer holen und setzen.
d12f   20  93  df    jsr   $df93        Puffernummer holen
d132   0a           asl                mal 2
d133   aa           tax                und als Index nehmen
d134   a4  82       ldy   $82          aktuelle Kanalnummer
d136   60           rts                Ende
------------------------------------------------------------------------
d137                                    Holen eines Bytes aus dem aktiven Puffer.
d137   20  2f  d1    jsr   $d12f        Puffer. und Kanalnummer holen
d13a   b9  44  02    lda   $0244,y      Zeiger auf das letzte Zeichen im Puffer
d13d   f0  12       beq   $d151        verzweige, wenn Null
d13f   a1  99       lda   ($99,x)      Zeichen aus Puffer holen
d141   48           pha                und merken
d142   b5  99       lda   $99,x        Pufferzeiger Lo
d144   d9  44  02    cmp   $0244,y      zeigt er schon auf das letzte Zeichen?
d147   d0  04       bne   $d14d        verzweige, wenn nein
d149   a9  ff       lda   #$ff         255
d14b   95  99       sta   $99,x        als Pufferzeiger Lo setzen
d14d   68           pla                Byte zurückholen
d14e   f6  99       inc   $99,x        Pufferzeiger Lo erhöhen
d150   60           rts                Ende
d151   a1  99       lda   ($99,x)      Byte aus Puffer holen
d153   f6  99       inc   $99,x        Pufferzeiger Lo erhöhen
d155   60           rts                Ende
```

```
------------------------------------------------------------------------
d156                                    Holen eines Bytes aus einer Datei. Wenn nötig, wird der
                                        nächste Block einer Datei gelesen. Es wird ein EOI-Signal
                                        in $f2 übergeben, falls das letzte Byte gelesen wurde.
d156    20  37  d1    jsr   $d137       Byte aus Puffer holen
d159    d0  36        bne   $d191       verzweige, wenn Pufferzeiger Lo ungleich 0
d15b    85  85        sta   $85         Byte merken
d15d    b9  44  02    lda   $0244,y     Endezeiger in Puffer
d160    f0  08        beq   $d16a       verzweige, wenn Null
d162    a9  80        lda   #$80        Wert für Lesen
d164    99  f2  00    sta   $00f2,y     als Kanalstatus speichern
d167    a5  85        lda   $85         Datenbyte zurückholen
d169    60            rts               Ende
d16a    20  1e  cf    jsr   $cf1e       Folgeblock der Datei lesen
d16d    a9  00        lda   #$00        Pufferzeiger auf 0
d16f    20  c8  d4    jsr   $d4c8       setzen
d172    20  37  d1    jsr   $d137       erstes Byte aus dem Puffer holen
d175    c9  00        cmp   #$00        gleich 0?
d177    f0  19        beq   $d192       verzweige, wenn ja; letzter Block im Puffer
d179    85  80        sta   $80         sonst Tracknummer übernehmen
d17b    20  37  d1    jsr   $d137       zweites Byte aus Puffer holen
d17e    85  81        sta   $81         als Sektornummer speichern
d180    20  1e  cf    jsr   $cf1e       Folgeblock in anderen Puffer lesen
d183    20  d3  d1    jsr   $d1d3       Puffer- und Drivenummer setzen
d186    20  d0  d6    jsr   $d6d0       Parameter zum Lesen an DC übergeben
d189    20  c3  d0    jsr   $d0c3       Befehl für Lesen an DC übergeben
d18c    20  1e  cf    jsr   $cf1e       Puffer wechseln; Block lesen.
d18f    a5  85        lda   $85         Datenbyte nach A holen
d191    60            rts               Ende
d192    20  37  d1    jsr   $d137       nächstes Byte aus Puffer holen
d195    a4  82        ldy   $82         aktuelle Kanalnummer als Index
d197    99  44  02    sta   $0244,y     Byte als Endezeiger merken (Anzahl der Bytes)
d19a    a5  85        lda   $85         aktuelles Datenbyte nach A holen
d19c    60            rts               Ende
------------------------------------------------------------------------
d19d                                    Schreiben eines Bytes in den aktiven Puffer Wird der Puffer
                                        dadurch gefüllt, so wird er auf Diskette geschrieben.
d19d    20  f1  cf    jsr   $cff1       Byte in Puffer schreiben
d1a0    f0  01        beq   $d1a3       verzweige, wenn Puffer voll
d1a2    60            rts               Ende, wenn noch nicht voll
d1a3    20  d3  d1    jsr   $d1d3       Drive- und Puffernummer holen
d1a6    20  1e  f1    jsr   $f11e       nächsten freien Block in der BAM suchen
d1a9    a9  00        lda   #$00        Wert (0) für Pufferzeiger holen
d1ab    20  c8  d4    jsr   $d4c8       und Pufferzeiger auf Null (A) setzen
d1ae    a5  80        lda   $80         Tracknummer
d1b0    20  f1  cf    jsr   $cff1       in Puffer schreiben
d1b3    a5  81        lda   $81         Sektornummer
d1b5    20  f1  cf    jsr   $cff1       in Puffer schreiben
d1b8    20  c7  d0    jsr   $d0c7       Block auf Diskette schreiben
d1bb    20  1e  cf    jsr   $cf1e       auf Zweitpuffer umschalten
d1be    20  d0  d6    jsr   $d6d0       Parameter für nächsten Block an DC schicken
d1c1    a9  02        lda   #$02        Wert (2) für Pufferzeiger holen
d1c3    4c  c8  d4    jmp   $d4c8       und Pufferzeiger auf 2 (A) setzen
```

```
------------------------------------------------------------------------
d1c6                                    Erhöhen des aktuellen Pufferzeigers.
d1c6   85  6f       sta   $6f          Wert zwischenspeichern
d1c8   20  e8  d4   jsr   $d4e8        aktiven Pufferzeiger holen
d1cb   18           clc               zu gespeichertem Wert
d1cc   65  6f       adc   $6f          addieren
d1ce   95  99       sta   $99,x       und Pufferzeiger neu setzen
d1d0   85  94       sta   $94          Directorypufferzeiger Lo setzer
d1d2   60           rts               Ende
------------------------------------------------------------------------
d1d3                                    Holen und Setzen der Laufwerksnr.
d1d3   20  93  df   jsr   $df93        Puffernummer holen
d1d6   aa           tax               und nach X geben
d1d7   bd  5b  02   lda   $025b,x      aus Tabelle mit Pufferbefehlscodes
d1da   29  01       and   #$01         Drivenummer isolieren
d1dc   85  7f       sta   $7f          und als aktuelle Nummer speichern
d1de   60           rts               Ende
------------------------------------------------------------------------
d1df                                    Routine zum Suchen eines Kanals und des zuge- hörigen
                                        Puffers. Erfolgt der Einsprung bei $d1df, so wird ein
                                        Schreibkanal gesucht. Bei Einsprung ab $d1e2 wird ein
                                        Lesekanal gesucht.
d1df   38           sec               Flag für Schreibkanal setzen
d1e0   b0  01       bcs   $d1e3       unbedingter Sprung

d1e2   18           clc               Flag für Lesekanal setzen
d1e3   08           php               und merken
d1e4   85  6f       sta   $6f          A muß die Anzahl der Puffer enthalten
d1e6   20  27  d2   jsr   $d227        Kanal schließen
d1e9   20  7f  d3   jsr   $d37f        nächsten freien Kanal belegen
d1ec   85  82       sta   $82          Kanalnummer übernehmen
d1ee   a6  83       ldx   $83          Sekundäradresse als Index
d1f0   28           plp               Flag für Schreib- oder Lesekanal zurückholen
d1f1   90  02       bcc   $d1f5       verzweige, wenn Lesekanal gewünscht
d1f3   09  80       ora   #$80         Schreibstatus
d1f5   9d  2b  02   sta   $022b,x      in Tabelle setzen
d1f8   29  3f       and   #$3f         Kanalnummer wieder isolieren
d1fa   a8           tay               als Index für Tabelle
d1fb   a9  ff       lda   #$ff         Code für unbenutzen Puffer
d1fd   99  a7  00   sta   $00a7,y      in Pufferbelegungstabelle
d200   99  ae  00   sta   $00ae,y      in Pufferstatustabelle
d203   99  cd  00   sta   $00cd,y      in Side-Sektor-Tabelle schreiben
d206   c6  6f       dec   $6f          Pufferanzahl erniedrigen
d208   30  1c       bmi   $d226        verzweige, wenn Anzahl kleiner Null
d20a   20  8e  d2   jsr   $d28e        Puffer suchen
d20d   10  08       bpl   $d217        verzweige, wenn gefunden
d20f   20  5a  d2   jsr   $d25a        Belegung in Tabellen löschen
d212   a9  70       lda   #$70         Nummer der Fehlermeldung
d214   4c  c8  c1   jmp   $c1c8        '70, NO CHANNEL' ausgeben
d217   99  a7  00   sta   $00a7,y      Puffernummer in Belegungstabelle
d21a   c6  6f       dec   $6f          Anzahl der benötigten Puffer erniedrigen
d21c   30  08       bmi   $d226        verzweige, wenn kleiner Null
d21e   20  8e  d2   jsr   $d28e        Puffer suchen
d221   30  ec       bmi   $d20f        verzweige, wenn nicht gefunden
d223   99  ae  00   sta   $00ae,y      Puffernummer in Belegungstabelle
d226   60           rts               Ende
```

```
--------------------------------------------------------------------------
d227                                    Freigeben aller Schreib- und Lesekanäle außer dem
                                        Kommandokanal durch Löschen der Belegungen in den Tabellen.
d227   a5  83       lda   $83          Sekundäradresse
d229   c9  0f       cmp   #$0f         mit 15 vergleichen
d22b   d0  01       bne   $d22e        verzweige, wenn nein
d22d   60           rts                Ende
d22e   a6  83       ldx   $83          Sekundäradresse als Index
d230   bd  2b  02    lda   $022b,x      Kanalstatus holen
d233   c9  ff       cmp   #$ff         ist Kanal unbenutzt?
d235   f0  22       beq   $d259        verzweige, wenn ja
d237   29  3f       and   #$3f         Kanalnummer isolieren
d239   85  82       sta   $82          und abspeichern
d23b   a9  ff       lda   #$ff         Code für unbenutzten Puffer
d23d   9d  2b  02    sta   $022b,x      in Tabelle eintragen
d240   a6  82       ldx   $82          aktuelle Kanalnummer
d242   a9  00       lda   #$00         Flags in Kanalstatustabelle
d244   95  f2       sta   $f2,x        löschen
d246   20  5a  d2    jsr   $d25a        Puffer wieder freigeben
d249   a6  82       ldx   $82          aktuelle Kanalnummer
d24b   a9  01       lda   #$01         Bitflag für
d24d   ca           dex                die aktuelle Kanalnummer
d24e   30  03       bmi   $d253        an die ihr entsprechende
d250   0a           asl                Position
d251   d0  fa       bne   $d24d        schieben
d253   0d  56  02    ora   $0256        und in das Register
d256   8d  56  02    sta   $0256        eintragen
d259   60           rts                Ende
--------------------------------------------------------------------------
d25a                                    Puffer und dessen Kanalzuordnung freigeben.
d25a   a6  82       ldx   $82          Kanalnummer als Index
d25c   b5  a7       lda   $a7,x        zugehörige Puffernummer holen
d25e   c9  ff       cmp   #$ff         ist Puffer diesem Kanal zugeordnet?
d260   f0  09       beq   $d26b        verzweige, wenn nein
d262   48           pha                Puffernummer merken
d263   a9  ff       lda   #$ff         Code für unbenutzten Puffer
d265   95  a7       sta   $a7,x        Pufferzuordnung löschen
d267   68           pla                Puffernummer zurückholen
d268   20  f3  d2    jsr   $d2f3        und Puffer freigeben
d26b   a6  82       ldx   $82          Kanalnummer als Index
d26d   b5  ae       lda   $ae,x        zugehörige Puffernummer holen
d26f   c9  ff       cmp   #$ff         ist Puffer belegt?
d271   f0  09       beq   $d27c        verzweige, wenn nein
d273   48           pha                Puffernummer merken
d274   a9  ff       lda   #$ff         Code für unbenutzten Puffer
d276   95  ae       sta   $ae,x        Pufferzuordnung löschen
d278   68           pla                Puffernummer zurückholen
d279   20  f3  d2    jsr   $d2f3        Puffer freigeben
d27c   a6  82       ldx   $82          Kanalnummer als Index
d27e   b5  cd       lda   $cd,x        zugehörige Puffernummer holen
d280   c9  ff       cmp   #$ff         ist Puffer belegt?
d282   f0  09       beq   $d28d        verzweige, wenn nein
d284   48           pha                Puffernummer merken
d285   a9  ff       lda   #$ff         Code für unbenutzten Puffer
d287   95  cd       sta   $cd,x        Pufferzuordnung löschen
d289   68           pla                Puffernummer zurückholen
d28a   20  f3  d2    jsr   $d2f3        Puffer freigeben
d28d   60           rts                Ende
```

```
------------------------------------------------------------------------
d28e                                    Suchen eines Puffers.
d28e    98          tya                 Puffernummer in Y
d28f    48          pha                 merken
d290    a0  01      ldy   #$01          Puffer 8-15
d292    20  ba  d2  jsr   $d2ba         nach freiem Puffer durchsuchen
d295    10  0c      bpl   $d2a3         verzweige, wenn gefunden
d297    88          dey                 Puffer 0- 7
d298    20  ba  d2  jsr   $d2ba         nach freiem Puffer durchsuchen
d29b    10  06      bpl   $d2a3         verzweige, wenn gefunden
d29d    20  39  d3  jsr   $d339         inaktiven Puffer 'stehlen'
d2a0    aa          tax                 Puffernummer nach X
d2a1    30  13      bmi   $d2b6         verzweige, wenn kein Puffer
d2a3    b5  00      lda   $00,x         Jobcode holen
d2a5    30  fc      bmi   $d2a3         und auf Ende des Jobs warten
d2a7    a5  7f      lda   $7f           aktuelte Drivenummer
d2a9    95  00      sta   $00,x         Jobcode in Jobspeicher
d2ab    9d  5b  02  sta   $025b,x       und Befehlscodetabelle löschen
d2ae    8a          txa                 Puffernummer zurückholen
d2af    0a          asl                 mal 2
d2b0    a8          tay                 als Index in Tabelle
d2b1    a9  02      lda   #$02          Wert für Pufferzeiger Lo
d2b3    99  99  00  sta   $0099,y       Pufferzeiger auf 3.Byte
d2b6    68          pla                 Puffernummer zurückholen
d2b7    a8          tay                 und nach Y
d2b8    8a          txa                 Nummer des gefundenen Puffers
d2b9    60          rts                 Ende
------------------------------------------------------------------------
d2ba                                    Suchen eines freien Puffers. Y bestimmt die Puffernummern:
Y=0 heißt Puffer 0-7;
Y=1 heißt Puffer 8-15.
War die Suche erfolgreich, so enthält X die Puffernummer, sonst enthält X den Wert $ff.
d2ba    a2  07      ldx   #$07          Wert für Bit-Test
d2bc    b9  4f  02  lda   $024f,y       Pufferbelegungsspeicher
d2bf    3d  e9  ef  and   $efe9,x       auf Belegung testen
d2c2    f0  04      beq   $d2c8         verzweige, wenn Puffer frei
d2c4    ca          dex                 nächsten Puffer
d2c5    10  f5      bpl   $d2bc         und weitermachen
d2c7    60          rts                 Ende, wenn kein freier Puffer vorhanden ist
d2c8    b9  4f  02  lda   $024f,y       Pufferbelegungstabelle
d2cb    5d  e9  ef  eor   $efe9,x       Puffer belegen
d2ce    99  4f  02  sta   $024f,y       und in Tabelle abspeichern
d2d1    8a          txa                 Puffernummer
d2d2    88          dey                 welcher Pufferbereich?
d2d3    30  03      bmi   $d2d8         verzweige, wenn Puffer 0-7
d2d5    18          clc                 Puffer 8-15
d2d6    69  08      adc   #$08          waren gewünscht
d2d8    aa          tax                 Puffernummer
d2d9    60          rts                 Ende
d2da    a6  82      ldx   $82           Kanalnummer als Index
d2dc    b5  a7      lda   $a7,x         zugehörige Puffernummer holen
d2de    30  09      bmi   $d2e9         verzweige, wenn Puffer frei
d2e0    8a          txa                 Kanalnummer
d2e1    18          clc                 wird zu Zeiger
```

```
d2e2   69  07       adc   #$07        auf Alternativtabelle
d2e4   aa           tax               erhöht
d2e5   b5  a7       lda   $a7,x        alternative Puffernummer holen
d2e7   10  f0       bpl   $d2d9       Ende, wenn auch belegt
d2e9   c9  ff       cmp   #$ff        ist Puffer frei ?
d2eb   f0  ec       beq   $d2d9       verzweige, wenn ja
d2ed   48           pha               Puffernummer merken
d2ee   a9  ff       lda   #$ff        Code für unbenutzten Puffer
d2f0   95  a7       sta   $a7,x       Puffer freigeben
d2f2   68           pla               Puffernummer zurückholen
d2f3   29  0f       and   #$0f        und auf 15 begrenzen
d2f5   a8           tay               nach Y
d2f6   c8           iny               die neue
d2f7   a2  10       ldx   #$10        Belegung in das
d2f9   6e  50  02   ror   $0250       Pufferbelegungsregister
d2fc   6e  4f  02   ror   $024f       eintragen
d2ff   88           dey               nächste Puffernummer
d300   d0  01       bne   $d303       verzweige, wenn richtiger Puffer aktualisiert
d302   18           clc               mit
d303   ca           dex               nächstem Puffer
d304   10  f3       bpl   $d2f9       weitermaehen
d306   60           rts               Ende
```

```
--------------------------------------------------------------------------
d307                                    Alle Kanäle außer dem Kommandokanal löschen und freigeben.
d307   a9  0e       lda   #$0e          14
d309   85  83       sta   $83           als Sekundäradresse
d30b   20  27  d2    jsr   $d227         Kanal schließen
d30e   c6  83       dec   $83           nächste Sekundäradresse
d310   d0  f9       bne   $d30b         und Kanal ebenfalls schließen
d312   60           rts                 Ende
--------------------------------------------------------------------------
d313                                    Alle Kanäle des anderen Laufwerks schließen; außer dem
                                        Kommandokanal.
d313   a9  0e       lda   #$0e          14
d315   85  83       sta   $83           als Sekundäradresse
d317   a6  83       ldx   $83           Sekundär adresse als Index in Tabelle
d319   bd  2b  02    lda   $022b,x       Kanalstatus holen
d31c   c9  ff       cmp   #$ff          Kanal in Betrieb?
d31e   f0  14       beq   $d334         verzweige, wenn nein
d320   29  3f       and   #$3f          Kanalnummer isolieren
d322   85  82       sta   $82           und übernehmen
d324   20  93  df    jsr   $df93         Puffernummer holen
d327   aa           tax                 nach X
d328   bd  5b  02    lda   $025b,x       Jobtabelle
d32b   29  01       and   #$01          Drivenummer isolieren
d32d   c5  7f       cmp   $7f           mit aktueller Nummer vergleichen
d32f   d0  03       bne   $d334         verzweige, wenn ungleich
d331   20  27  d2    jsr   $d227         Kanal freigeben
d334   c6  83       dec   $83           nächste Sekundäradresse
d336   10  df       bpl   $d317         und weitermachen
d338   60           rts                 Ende
--------------------------------------------------------------------------
d339                                    'Stehlen' eines inaktiven Puffers, der anhand der Tabelle
                                        gefunden wird. Die Nummer dieses Puffers wird in A
                                        übergeben.
d339   a5  6f       lda   $6f          Kanalnummer
d33b   48           pha                 merken
d33c   a0  00       ldy   #$00          Index für Tabelle
d33e   b6  fa       ldx   $fa,y         Kanalnummer holen
d340   b5  a7       lda   $a7,x         zugehörige Puffernummer holen
d342   10  04       bpl   $d348         verzweige, wenn Puffer belegt
d344   c9  ff       cmp   #$ff          Puffer frei?
d346   d0  16       bne   $d35e         verzweige, wenn nein
d348   8a           txa                 Kanalnummer
d349   18           clc                 maximale Anzahl an Kanälen
d34a   69  07       adc   #$07          addieren
d34c   aa           tax                 um alternative Puffernummer zu
d34d   b5  a7       lda   $a7,x         holen
d34f   10  04       bpl   $d355         verzweige, wenn ebenfalls belegt
d351   c9  ff       cmp   #$ff          Puffer frei?
d353   d0  09       bne   $d35e         verzweige, wenn nein
d355   c8           iny                 nächste Kanalnummer
d356   c0  05       cpy   #$05          und
d358   90  e4       bcc   $d33e         weitermachen
d35a   a2  ff       ldx   #$ff          Code für Fehler
d35c   d0  1c       bne   $d37a         unbedingter Sprung; Ende
d35e   86  6f       stx   $6f          Kanalnummer merken
d360   29  3f       and   #$3f          Puffernummer isolieren
```

```
d362   aa          tax             als Index
d363   b5  00      lda   $00,x     zugehörigen Jobcode holen
d365   30  fc      bmi   $d363     auf Ende des Jobs warten
*1 d367c9  02      cmp   #$02      Fehler aufgetreten?
*0 d367 4c  3f  aa  jmp   $aa3f     Rückmeldung prüfen
d36a   ea          nop             $d373 verzweige, wenn nein
d36b   a6  6f      ldx   $6f       Kanalnummer
d36d   e0  07      cpx   #$07      mit Maximum vergleichen
d36f   90  d7      bcc   $d348     weitermachen, wenn kleiner
d371   b0  e2      bcs   $d355     unbedingter Sprung
d373   a4  6f      ldy   $6f       Kanalnummer als Index
d375   a9  ff      lda   #$ff      Code für unbenutzten Puffer
d377   99  a7  00  sta   $00a7,y   Puffer freigeben
d37a   68          pla             alte Kanalnummer zurückholen
d37b   85  6f      sta   $6f       und wieder abspeichern
d37d   8a          txa             Puffernummer
d37e   60          rts             Ende
-------------------------------------------------------------------------
d37f                               Freien Kanal suchen und belegen
d37f   a0  00      ldy   #$00      Kanalnummer
d381   a9  01      lda   #$01      Bitflag für Kanalbelegung
d383   2c  56  02  bit   $0256     Kanal frei?
d386   d0  09      bne   $d391     verzweige, wenn ja
d388   c8          iny             nächste Kanalnummer
d389   0a          asl             Flag für nächsten Kanal
d38a   d0  f7      bne   $d383     und weitermachen
d38c   a9  70      lda   #$70      Nummer der Fehlermeldung
d38e   4c  c8  c1  jmp   $c1c8     '70, NO CHANNEL' ausgeben
d391   49  ff      eor   #$ff      Kanal belegen
d393   2d  56  02  and   $0256     und in Tabelle
d396   8d  56  02  sta   $0256     anzeigen
d399   98          tya             Kanalnummer
d39a   60          rts             Ende
-------------------------------------------------------------------------
d39b                               Nächstes Byte eines Kanals holen.
d39b   20  eb  d0  jsr   $d0eb     Kanal zum Lesen öffnen
d39e   20  00  c1  jsr   $c100     LED am Laufwerk einschalten
d3a1   20  aa  d3  jsr   $d3aa     Byte über Kanal holen
d3a4   a6  82      ldx   $82       Kanalnummer
d3a6   bd  3e  02  lda   $023e,x   aktuelles Byte in A
d3a9   60          rts             Ende
-------------------------------------------------------------------------
d3aa                               Nächstes Byte irgendeines Files holen
d3aa   a6  82      ldx   $82       Kanalummer
d3ac   20  25  d1  jsr   $d125     Filetyp holen
d3af   d0  03      bne   $d3b4     verzweige, wenn keine relative Datei
d3b1   4c  20  e1  jmp   $e120     relative Datei bearbeiten
d3b4   a5  83      lda   $83       aktuelle Sekundäradresse
d3b6   c9  0f      cmp   #$0f      auf Kommandokanal prüfen
d3b8   f0  5a      beq   $d414     verzweige, wenn Kommandokanal
d3ba   b5  f2      lda   $f2,x     Kanalstatus holen
d3bc   29  08      and   #$08      auf EOI prüfen
d3be   d0  13      bne   $d3d3     verzweige bei keinem EOI
d3c0   20  25  d1  jsr   $d125     Filetyp holen
d3c3   c9  07      cmp   #$07      Direktzugriffsdatei?
d3c5   d0  07      bne   $d3ce     verzweige, wenn nein
d3c7   a9  89      lda   #$89      Flags für Direktzugriff
```

```
d3c9   95  f2       sta   $f2,x       in Tabelle setzen
d3cb   4c  de  d3    jmp   $d3de       Byte für Direktzugriff holen
d3ce   a9  00        lda   #$00        Kanalstatus
d3d0   95  f2        sta   $f2,x       löschen; EOI
d3d2   60            rts               Ende
d3d3   a5  83        lda   $83         Sekundäradresse
d3d5   f0  32        beq   $d409       verzweige, wenn LOAD
d3d7   20  25  d1    jsr   $d125       Filetyp holen
d3da   c9  04        cmp   #$04        Direktzugriff?
d3dc   90  22        bcc   $d400       verzweige, wenn nein
d3de   20  2f  d1    jsr   $d12f       Puffer- und Kanalnummer holen
d3e1   b5  99        lda   $99,x       Pufferzeiger Lo
d3e3   d9  44  02    cmp   $0244,y     gleich Endezeiger?
d3e6   d0  04        bne   $d3ec       verzweigen, wenn nein
d3e8   a9  00        lda   #$00        Pufferzeiger Lo
d3ea   95  99        sta   $99,x       auf Null setzen
d3ec   f6  99        inc   $99,x       Pufferzeiger Lo erhöhen
d3ee   a1  99        lda   ($99,x)     Byte aus Puffer holen
d3f0   99  3e  02    sta   $023e,y     in Tabelle für Ausgabe
d3f3   b5  99        lda   $99,x       Pufferzeiger Lo
d3f5   d9  44  02    cmp   $0244,y     gleich Endezeiger?
d3f8   d0  05        bne   $d3ff       verzweige, wenn nein
d3fa   a9  81        lda   #$81        letztes Zeichen
d3fc   99  f2  00    sta   $00f2,y     EOI setzen
d3ff   60            rts               Ende
d400   20  56  d1    jsr   $d156       Byte aus Puffer holen
d403   a6  82        ldx   $82         Kanalnummer als Index
d405   9d  3e  02    sta   $023e,x     Byte in Tabelle für Ausgabe
d408   60            rts               Ende
d409   ad  54  02    lda   $0254       Flag für Directory
d40c   f0  f2        beq   $d400       verzweige, wenn nicht gesetzt
d40e   20  67  ed    jsr   $ed67       Bytes aus Directory holen
d411   4c  03  d4    jmp   $d403       Ende
--------------------------------------------------------------------------
d414                                   Lesen eines Bytes aus dem Fehlerkanal.
d414   20  e8  d4    jsr   $d4e8       aktiven Pufferzeiger holen
d417   c9  d4        cmp   #$d4        ERROR - Puffer?
d419   d0  18        bne   $d433       verzweige, wenn nein
d41b   a5  95        lda   $95         Zeiger auf Directorypuffer Hi
d41d   c9  02        cmp   #$02        zeigt auf Fehlerpuffer?
d41f   d0  12        bne   $d433       verzweige, wenn nein
d421   a9  0d        lda   #$0d        'RETURN'
d423   85  85        sta   $85         ins Ausgaberegister
d425   20  23  c1    jsr   $c123       Fehlerflags löschen
d428   a9  00        lda   #$00        Code für 'OK'-Meldung
d42a   20  c1  e6    jsr   $e6c1       Meldung bereitstellen
d42d   c6  a5        dec   $a5         Zeiger in ERROR-Puffer Lo minus 1
d42f   a9  80        lda   #$80        Flag für EOI löschen
d431   d0  12        bne   $d445       unbedingter Sprung
d433   20  37  d1    jsr   $d137       Byte holen
d436   85  85        sta   $85         ins Ausgaberegister
d438   d0  09        bne   $d443       verzweige, wenn Byte ungleich Null
d43a   a9  d4        lda   #$d4        ert für ERROR-Puffer-Zeiger Lo
d43c   20  c8  d4    jsr   $d4c8       Pufferzeiger setzen
d43f   a9  02        lda   #$02        Pufferzeiger Hi
d441   95  9a        sta   $9a,x       setzen
d443   a9  88        lda   #$88        READ-Flag
d445   85  f7        sta   $f7         setzen
```

```
d447   a5  85       lda   $85         Datenbyte
d449   8d  43  02   sta   $0243       für Ausgabe bereitstellen
d44c   60           rts               Ende
-------------------------------------------------------------------------
d44d                                  Lesen des nächsten Blocks eines Files und Setzen eines EOF-
                                      Signales, falls der Linker der Spur dieses Blocks gleich
                                      $00 ist.
d44d   20  93  df   jsr   $df93       Puffernummer holen
d450   0a          asl               mal 2
d451   aa          tax               als Index fur Tabelle
d452   a9  00       lda   #$00        Pufferzeiger Lo
d454   95  99       sta   $99,x       auf Null setzen
d456   a1  99       lda   ($99,x)     Tracknummer aus Puffer holen
d458   f0  05       beq   $d45f       verzweige, wenn kein Folgeblock
d45a   d6  99       dec   $99,x       Pufferzeiger Lo inaktiv setzen
d45c   4c  56  d1   jmp   $d156       Folgeblock lesen
d45f   60           rts               Ende
-------------------------------------------------------------------------
d460                                  Routine zur Übergabe der Parameter an den DC. Bei Einsprung
                                      ab $d460 wird der Code zum Lesen ($80), bei Einsprung ab
                                      $d464 der Code zum Schreiben ($90) übergeben. Anschließend
                                      wird das Kommando ausgeführt.
d460   a9  80       lda   #$80        Jobcode fur Lesen
d462   d0  02       bne   $d466       unbedingter Sprung
d464   a9  90       lda   #$90        Jobcode fur Schreiben
d466   05  7f       ora   $7f         mit Drivenummer verknüpfen
d468   8d  4d  02   sta   $024d       in Jobtabelle eintragen
d46b   a5  f9       lda   $f9         aktuelle Puffernummer
d46d   20  d3  d6   jsr   $d6d3       Parameter an Diskcontroller
d470   a6  f9       ldx   $f9         aktuelle Puffernummer
d472   4c  93  d5   jmp   $d593       Job ausführen
-------------------------------------------------------------------------
d475                                  Öffnen eines Kanals zum Lesen. Der Filetyp wird auf SEQ
                                      gesetzet. Anschließend wird der Block gelesen.
d475   a9  01       lda   #$01        Nummer für SEQ
d477   8d  4a  02   sta   $024a       setzen
d47a   a9  11       lda   #$11        17 (READ)
d47c   85  83       sta   $83         als Sekundäradresse setzen
d47e   20  46  dc   jsr   $dc46       Puffer belegen; Block lesen
d481   a9  02       lda   #$02        Wert für Pufferzeiger
d483   4c  c8  d4   jmp   $d4c8       Pufferzeiger setzen; Ende
-------------------------------------------------------------------------
d486                                  Öffnen eines internen Schreibkanals.
d486   a9  12       lda   #$12        18 (WRITE)
d488   85  83       sta   $83         als Sekundäradresse setzen
d48a   4c  da  dc   jmp   $dcda       Schreibkanal öffnen
-------------------------------------------------------------------------
d48d                                  Belegen und Schreiben des nächsten Directory' Blocks.
d48d   20  3b  de   jsr   $de3b       Track- und Sektornummer holen
d490   a9  01       lda   #$01        ein
d492   85  6f       sta   $6f         Block
d494   a5  69       lda   $69         Abstand der Sektoren (10)
d496   48          pha               merken
d497   a9  03       lda   #$03        durch 3 bei Directory
d499   85  69       sta   $69         ersetzen
```

```
d49b   20  2d  f1    jsr   $f12d       nächsten freien Block hole
d49e   68            pla               Schrittweite zurückholen
d49f   85  69        sta   $69         und abspeichern
d4a1   a9  00        lda   #$00        Pufferzeiger auf
d4a3   20  c8  d4    jsr   $d4c8       Null setzen
d4a6   a5  80        lda   $80         Tracknummer
d4a8   20  f1  cf    jsr   $cff1       in Puffer schreiben
d4ab   a5  81        lda   $81         Sektornummer
d4ad   20  f1  cf    jsr   $cff1       in Puffer schreiben
d4b0   20  c7  d0    jsr   $d0c7       Block auf Diskette schreiben
d4b3   20  99  d5    jsr   $d599       Ende des Jobs abwarten
d4b6   a9  00        lda   #$00        Pufferzeiger auf
d4b8   20  c8  d4    jsr   $d4c8       Null setzen
d4bb   20  f1  cf    jsr   $cff1       Puffer mit $00
d4be   d0  fb        bne   $d4bb       füllen
d4c0   20  f1  cf    jsr   $cff1       Null als Tracknummer in Puffer
d4c3   a9  ff        lda   #$ff        256 Bytes als Anzahl
d4c5   4c  f1  cf    jmp   $cff1       in Puffer schreiben; Ende
--------------------------------------------------------------------------
d4c8                                   Setzen des aktuellen Pufferzeigers. A enthält den neuen
                                       Wert, der gesetzt werden soll-
d4c8   85  6f        sta   $6f         Wert merken
d4ca   20  93  df    jsr   $df93       aktuelle Puffernummer holen
d4cd   0a            asl               mal 2
d4ce   aa            tax               als Index für Tabelle
d4cf   b5  9a        lda   $9a,x       Pufferzeiger Hi
d4d1   85  95        sta   $95         übernehmen
d4d3   a5  6f        lda   $6f         neuen Wert
d4d5   95  99        sta   $99,x       für Pufferzeiger Lo
d4d7   85  94        sta   $94         übernehmen
d4d9   60            rts               Ende
--------------------------------------------------------------------------
d4da                                   Beide internen Kanäle (Lese- und Schreibkanal) schließen.
d4da   a9  11        lda   #$11        17 (READ)
d4dc   85  83        sta   $83         als Sekundäradresse setzen
d4de   20  27  d2    jsr   $d227       Kanal schließen
d4e1   a9  12        lda   #$12        18 (WRITE)
d4e3   85  83        sta   $83         als Sekundäradresse setzen
d4e5   4c  27  d2    jmp   $d227       Kanal schließen; Ende
--------------------------------------------------------------------------
d4e8                                   Aktuellen Pufferzeiger setzen.
d4e8   20  93  df    jsr   $df93       Puffernummer holen
d4eb   0a            asl               mal 2
d4ec   aa            tax               als Index für Tabelle
d4ed   b5  9a        lda   $9a,x       Pufferzeiger Hi
d4ef   85  95        sta   $95         übernehmen
d4f1   b5  99        lda   $99,x       Pufferzeiger Lo
d4f3   85  94        sta   $94         übernehmen
d4f5   60            rts               Ende
--------------------------------------------------------------------------
d4f6                                   Lesen eines Bytes aus dem aktuellen Puffer. A enthält bei
                                       Einsprung die Nummer des gewünschten Bytes und bei der
                                       Rückkehr aus der Routine das gelesene Byte.
d4f6   85  71        sta   $71         A als Zeiger Lo merken
d4f8   20  93  df    jsr   $df93       Puffernummer holen
d4fb   aa            tax               nach X
```

```
d4fc    bd  e0  fe    lda   $fee0,x     Pufferadresse Hi holen
d4ff    85  72        sta   $72         als Zeiger Hi merken
d501    a0  00        ldy   #$00        Byte aus
d503    b1  71        lda   ($71),y     dem Puffer holen
d505    60            rts               Ende
-------------------------------------------------------------------------
d506                                    Überprüfen von Track- und Sektornummer für die Jobschleife.
                                        X enthält die Puffernummer und $024d den Code für den Job.
d506    bd  5b  02    lda   $025b,x     Befehlscode aus Tabelle holen
d509    29  01        and   #$01        Drivenummer isolieren
d50b    0d  4d  02    ora   $024d       mit Jobcode verknüpfen
d50e    48            pha               und merken
d50f    86  f9        stx   $f9         Puffernummer setzen
d511    8a            txa               und
d512    0a            asl               mal 2 nehmen
d513    aa            tax               als Index für Tabelle
d514    b5  07        lda   $07,x       Sektornummer
d516    8d  4d  02    sta   $024d       merken
d519    b5  06        lda   $06,x       Tracknummer
d51b    f0  2d        beq   $d54a       Fehler, wenn Null
*1 d51d cd  ac  02    cmp   $02ac       vergleiche mit Maximum
*0 d51d cd  d7  fe    cmp   $fed7       mit Maximum vergleichen
d520    b0  28        bcs   $d54a       Fehler, wenn größer gleich
d522    aa            tax               Tracknummer merken
d523    68            pla               Jobcode zurückholen
d524    48            pha               Stack wiederherstellen
d525    29  f0        and   #$f0        Jobcode isolieren
d527    c9  90        cmp   #$90        Code für Schreiben?
d529    d0  4f        bne   $d57a       verzweige, wenn nein
d52b    68            pla               Jobcode zurückholen
d52c    48            pha               Stack wiederherstellen
d52d    4a            lsr               auf Drive 1 prüfen
d52e    b0  05        bcs   $d535       verzweige, wenn Drive 1
d530    ad  01  01    lda   $0101       Formatkennzeichen Drive 0
d533    90  03        bcc   $d538       unbedingter Sprung
d535    ad  02  01    lda   $0102       Formatkennzeichen Drive 1
d538    f0  05        beq   $d53f       verzweige, wenn Null
d53a    cd  d5  fe    cmp   $fed5       vergleiche mit 'A'
d53d    d0  33        bne   $d572       verzweige, wenn ungleich
d53f    8a            txa               Tracknummer
d540    20  4b  f2    jsr   $f24b       maximale Sektornummer holen
d543    cd  4d  02    cmp   $024d       mit Sektornummer vergleichen
d546    f0  02        beq   $d54a       Fehler, wenn gleich
d548    b0  30        bcs   $d57a       verzweige, wenn ok
d54a    20  52  d5    jsr   $d552       Track' und Sektornummer holen
d54d    a9  66        lda   #$66        Nummer der Fehlermeldung
d54f    4c  45  e6    jmp   $e645       '66, ILLEGAL TRACK OR SECTOR' ausgeben
-------------------------------------------------------------------------
d552                                    Track- und Sektornummer für Job holen.
d552    a5  f9        lda   $f9         aktuelle Puffernummer
d554    0a            asl               mal 2
d555    aa            tax               als Index für Tabelle
d556    b5  06        lda   $06,x       Tracknummer aus Jobspeicher
d558    85  80        sta   $80         übernehmen
d55a    b5  07        lda   $07,x       Sektornummer aus Jobspeicher
d55c    85  81        sta   $81         übernehmen
d55e    60            rts               Ende
```

```
------------------------------------------------------------------------
d55f                                    Prüft auf legale Track- und Sektornummern.
d55f    a5  80      lda   $80          Tracknummer
d561    f0  ea      beq   $d54d        Fehler, wenn Null
*1 d563 cd  ac  02  cmp   $02ac        mit Maximum vergleichen
*0 d563 cd  d7  fe  cmp   $fed7        mit Maximum vergleichen (30)
d566    b0  e5      bcs   $d54d        Fehler, wenn größer gleich
d568    20  4b  f2  jsr   $f24b        maximale Sektornummer holen
d56b    c5  81      cmp   $81          mit aktueller Nummer vergleichen
d56d    f0  de      beq   $d54d        Fehler, wenn gleich
d56f    90  dc      bcc   $d54d        Fehler, wenn kleiner
d571    60          rts                Ende
------------------------------------------------------------------------
d572                                    Track, und Sektornummer für die DOS-Mismatch- Meldung
                                        holen,
d572    20  52  d5  jsr   $d552        Track, und Sektornummer holen
d575    a9  73      lda   #$73         Fehlernummer in A
d577    4c  45  e6  jmp   $e645        '73, CBM DOS V3.0 1571' ausgeben
------------------------------------------------------------------------
d57a                                    Setzen des Jobs für einen Puffer.
d57a    a6  f9      ldx   $f9          aktuelle Puffernummer
d57c    68          pla                Befehlscode zurückholen
d57d    8d  4d  02  sta   $024d        als Jobcode speichern
d580    95  00      sta   $00,x        in Jobspeicher und
d582    9d  5b  02  sta   $025b,x      in Befehlscodetabelle
d585    60          rts                Ende
------------------------------------------------------------------------
d586                                    Jobcodes an DC übergeben. Bei Einsprung ab $d586 wird der
                                        Code für lesen ($80), bei Einsprung ab $d58a der Code für
                                        Schreiben übergeben.
d586    a9  80      lda   #$80         Code für lesen
d588    d0  02      bne   $d58c        unbedingter Sprung

d58a    a9  90      lda   #$90         Code für Schreiben
d58c    05  7f      ora   $7f          mit Drivenummer verknüpfen
d58e    a6  f9      ldx   $f9          aktuelle Puffernummer
d590    8d  4d  02  sta   $024d        Jobcode speichern
d593    ad  4d  02  lda   $024d        Jobcode in A
d596    20  0e  d5  jsr   $d50e        Parameter prüfen und an DCi Ende
------------------------------------------------------------------------
d599                                    Warten nach der Übergabe der Jobcodes auf die Beendigung
                                        des Jobs.
d599    20  a6  d5  jsr   $d5a6        Ausführung des Jobs prüfen
d59c    b0  fb      bcs   $d599        Warten auf Ende des Jobs
d59e    48          pha                Rückmeldung merken
d59f    a9  00      lda   #$00         Fehlerflag
d5a1    8d  98  02  sta   $0298        löschen
d5a4    68          pla                Rückmeldung zurückholen
d5a5    60          rts                Ende
------------------------------------------------------------------------
d5a6                                    Überprüft auf fehlerfreie Durchführung des Jobs. Wird ein
                                        Fehler auf Diskette außer 'WRITE PROTECT' oder 'ID
                                        MISMATCH' entdeckt, so wird versucht, durch Kopfjustierung
                                        erneut Daten von Diskette zu lesen.
d5a6    b5  00      lda   $00,x        Jobspeicher prüfen
d5a8    30  1a      bmi   $d5c4        verzweige, wenn DC noch beschäftigt
```

```
d5aa   c9  02       cmp   #$02        auf fehlerfreie Durchführung prüfen
d5ac   90  14       bcc   $d5c2       verzweige, wenn Durchführung ok
d5ae   c9  08       cmp   #$08        8 (WRITE PROTECT ON)?
d5b0   f0  08       beq   $d5ba       verzweige, wenn ja
d5b2   c9  0b       cmp   #$0b        11 (DISK ID MISMATCH)?
d5b4   f0  04       beq   $d5ba       verzweige, wenn ja
d5b6   c9  0f       cmp   #$0f        1S (DRIVE NOT READY)?
d5b8   d0  0c       bne   $d5c6       weiter versuchen, wenn nein
d5ba   2c  98  02   bit   $0298       wurde Fehler schon angezeigt?
d5bd   30  03       bmi   $d5c2       verzweige, wenn ja
d5bf   4c  3f  d6   jmp   $d63f       Fehlermeldung ausgeben
d5c2   18           clc               Flag für Ausführung beendet
d5c3   60           rts               Ende; alles ok
d5c4   38           sec               Flag für DC noch in Aktion
d5c5   60           rts               Ende
d5c6   98           tya               Index
d5c7   48           pha               merken
d5c8   a5  7f       lda   $7f         aktuelle Drivenummer
d5ca   48           pha               merken
d5cb   bd  5b  02   lda   $025b,x     Tabelle der Jobs
d5ce   29  01       and   #$01        Drivenummer dieses Jobs isolieren
d5d0   85  7f       sta   $7f         und übernehmen
d5d2   a8           tay               Drivenummer als Index
d5d3   b9  ca  fe   lda   $feca,y     Bitmaske fur LED holen
d5d6   8d  6d  02   sta   $026d       und abspeichern
d5d9   20  a6  d6   jsr   $d6a6       Anzahl Leseversuche (in $6a (5)) durchführen
d5dc   c9  02       cmp   #$02        Rückmeldung des letzten Versuchs
d5de   b0  03       bcs   $d5e3       verzweige, wenn Fehler
d5e0   4c  6d  d6   jmp   $d66d       Werte zurückholen; Ende
d5e3   bd  5b  02   lda   $025b,x     Tabelle der Jobcodes
d5e6   29  f0       and   #$f0        Code dieses Jobs isolieren
d5e8   48           pha               und merken
d5e9   c9  90       cmp   #$90        Code für Schreiben?
d5eb   d0  07       bne   $d5f4       verzweige, wenn nein
d5ed   a5  7f       lda   $7f         Drivenummer holen und mit Jobcode
d5ef   09  b8       ora   #$b8        für SUCHEN EINES SEKTORS verknüpfen
d5f1   9d  5b  02   sta   $025b,x     als neuen Jobcode merken
d5f4   24  6a       bit   $6a         Bit 6 (Kopf auf Track) prüfen
d5f6   70  39       bvs   $d631       verzweige, wenn Kopf auf Track
d5f8   a9  00       lda   #$00        Suche neben dem Track generieren
d5fa   8d  99  02   sta   $0299       Zeiger auf Werte für Kopfdejustage
d5fd   8d  9a  02   sta   $029a       Byte für Kopfpositionierung
d600   ac  99  02   ldy   $0299       Index für Tabelle holen
d603   ad  9a  02   lda   $029a       Byte für Kopfpositionierung holen
d606   38           sec               Wert für
d607   f9  db  fe   sbc   $fedb,y     Kopf justage bilden
d60a   8d  9a  02   sta   $029a       und merken
d60d   b9  db  fe   lda   $fedb,y     Wert für Positionierung holen
d610   20  a1  ff   jsr   $ffa1       Kopf neben den Track positionieren
d613   ee  99  02   inc   $0299       Index erhöhen
d616   20  a6  d6   jsr   $d6a6       wieder 5 Leseversuche
d619   c9  02       cmp   #$02        Rückmeldung prüfen
d61b   90  08       bcc   $d625       verzweige, wenn ok
d61d   ac  99  02   ldy   $0299       Index holen
d620   b9  db  fe   lda   $fedb,y     nächsten Wert aus Tabelle
d623   d0  db       bne   $d600       verzweige, wenn noch nicht zu Ende
d625   ad  9a  02   lda   $029a       Wert für Wiederherstellung holen
d628   20  a6  ff   jsr   $ffa6       Kopf wieder auf Track positionieren
```

```
d62b   b5  00       lda   $00,x       Code aus Jobspeicher
d62d   c9  02       cmp   #$02        Rückmeldung prüfen
d62f   90  2b       bcc   $d65c       verzweige, wenn ok
d631   24  6a       bit   $6a         Bit 7 (BUMP) prüfen
d633   10  0f       bpl   $d644       verzweige, wenn BUMP erfolger s:,l
d635   68           pla               Jobcode zurückholen
d636   c9  90       cmp   #$90        Schreiben?
d638   d0  05       bne   $d63f       verzweige, wenn nein
d63a   05  7f       ora   $7f         Drivenummer übernehmen
d63c   9d  5b  02    sta   $025b,x     und wieder in Tabelle schreiben
d63f   b5  00       lda   $00,x       Rückmeldung aus Jobspeicher
d641   20  0a  e6    jsr   $e60a       und Fehlermeldung ausgeben; Ende
d644   68           pla               Jobcode zurückholen
d645   2c  98  02    bit   $0298       auf Fehler prüfen
d648   30  23       bmi   $d66d       verzweige, wenn Fehler vorhanden
d64a   48           pha               Jobcode wieder merken
d64b   a9  c0       lda   #$c0        Jobcode für BUMP
d64d   05  7f       ora   $7f         mit Drivenummer verknüpfen
d64f   95  00       sta   $00,x       und in Jobspeicher eintragen
d651   20  b6  9f    jsr   $9fb6       Jobausführung erzwingen und
d654   ea           nop               Rückmeldung abwarten
d655   20  a6  d6    jsr   $d6a6       aktuellen Job abermals versuchen
d658   c9  02       cmp   #$02        Rückmeldung prüfen
d65a   b0  d9       bcs   $d635       verzweige, wenn Fehler
d65c   68           pla               Jobcode zurückholen
d65d   c9  90       cmp   #$90        Schreiben?
d65f   d0  0c       bne   $d66d       verzweige, wenn nein
d661   05  7f       ora   $7f         mit Drivenummer verknüpfen
d663   9d  5b  02    sta   $025b,x     und in Tabelle schreiben
d666   20  a6  d6    jsr   $d6a6       Ausführung nochmals versuchen
d669   c9  02       cmp   #$02        Rückmeldung prüfen
d66b   b0  d2       bcs   $d63f       verzweige, wenn Fehler
d66d   68           pla               aktuelle Drivenummer zurückholen
d66e   85  7f       sta   $7f         und setzen
d670   68           pla               Index zurückholen
d671   a8           tay               nach Y
d672   b5  00       lda   $00,x       Rückmeldung aus Jobspeicher
d674   18           clc               Flag für Ausführung beendet
d675   60           rts               Ende
-------------------------------------------------------------------------
d676                                  Routine zum Positionieren des Lese- Schreibkopfes relativ
                                      zur jetzigen Position. A enthält die Anzahl der zu
                                      fahrenden Steps; bei A kleiner 128 bewegt sich der Kopf
                                      nach außen; bei A größer gleich 128 bewegt sich der Kopf
                                      nach innen.
d676   c9  00       cmp   #$00        Null?
d678   f0  18       beq   $d692       Ende, wenn ja
d67a   30  0c       bmi   $d688       wenn negativ, dann Bewegung nach innen
d67c   a0  01       ldy   #$01        einen Step nach außen
d67e   20  93  d6    jsr   $d693       Kopf bewegen
d681   38           sec               Anzahl der
d682   e9  01       sbc   #$01        Steps vermindern
d684   d0  f6       bne   $d67c       weitermachen
d686   f0  0a       beq   $d692       unbedingter Sprung; Ende
d688   a0  ff       ldy   #$ff        eine Step nach innen
d68a   20  93  d6    jsr   $d693       Kopf bewegen
d68d   18           clc               Anzahl der
```

```
d68e   69  01       adc   #$01        Steps erhöhen
d690   d0  f6       bne   $d688       weitermachen
d692   60           rts               Ende
--------------------------------------------------------------------------
d693                                  Bewegen des Kopfes um einen Track nach innen oder außen.
d693   48           pha               Wert merken
d694   98           tya               Wert für Kopfbewegung
d695   a4  7f       ldy   $7f         Drivenummer als Index
d697   99  fe  02    sta   $02fe,y     Wert für Kopfbewegung an Diskcontroller
d69a   d9  fe  02    cmp   $02fe,y     Rückmeldung des Diskcontroller
d69d   f0  fb       beq   $d69a       abwarten
d69f   a9  00       lda   #$00        Parameter
d6a1   99  fe  02    sta   $02fe,y     wieder löschen
d6a4   68           pla               Wert wieder zurückholen
d6a5   60           rts               Ende
--------------------------------------------------------------------------
d6a6                                  Steuerung der Anzahl der Leseversuche bei Auftreten von
                                      Fehlern
d6a6   a5  6a       lda   $6a         Anzahl der Versuche (normalerweise 5)
d6a8   29  3f       and   #$3f        begrenzen
d6aa   a8           tay               und als Zähler merken
d6ab   ad  6d  02    lda   $026d       Maske für LED am Laufwerk
d6ae   4d  00  1c    eor   $1c00       LED umschalten; wird als das
d6b1   8d  00  1c    sta   $1c00       charakteristische Flackern deutlich
d6b4   bd  5b  02    lda   $025b,x     Jobcode holen
d6b7   95  00       sta   $00,x       und an DC übergeben
d6b9   20  b6  9f    jsr   $9fb6       Jobausführung erzwingen
d6bc   ea           nop               und Rückmeldung abwarten
d6bd   c9  02       cmp   #$02        Fehler aufgetreten?
d6bf   90  03       bcc   $d6c4       verzweige, wenn nein
d6c1   88           dey               Zähler vermindern
d6c2   d0  e7       bne   $d6ab       und noch einmal versuchen
d6c4   48           pha               Rückmeldung merken
d6c5   ad  6d  02    lda   $026d       Maske für LED am Laufwerk
d6c8   0d  00  1c    ora   $1c00       LED
d6cb   8d  00  1c    sta   $1c00       einschalten
d6ce   68           pla               Rückmeldung zurückholen
d6cf   60           rts               Ende
--------------------------------------------------------------------------
d6d0                                  Parameter für nächsten Job an den Diskcontrol- ler
                                      übergeben. Übergabe der Spur- und Sektornummer an den
                                      Jobspeicher.
d6d0   20  93  df    jsr   $df93       Puffernummer holen
d6d3   0a           asl               mal 2
d6d4   a8           tay               als Index in Tabelle
d6d5   a5  80       lda   $80         aktuelle Tracknummer
d6d7   99  06  00    sta   $0006,y     in Jobspeicher
d6da   a5  81       lda   $81         aktuelle Sektornummer
d6dc   99  07  00    sta   $0007,y     ebenfalls in Jobspeicher
d6df   a5  7f       lda   $7f         Drivenummer
d6e1   0a           asl               mal 2
d6e2   aa           tax               als Index vormerken
d6e3   60           rts               Ende
--------------------------------------------------------------------------
d6e4                                  Neue Datei im Directory eintragen.
d6e4   a5  83       lda   $83         aktuelle Sekundäradresse
d6e6   48           pha               merken
```

```
d6e7   a5 82       lda  $82         aktuelle Kanalnummer
d6e9   48          pha              ebenfalls merken
d6ea   a5 81       lda  $81         aktuelle Sektornummer
d6ec   48          pha              merken
d6ed   a5 80       lda  $80         aktuelle Tracknummer
d6ef   48          pha              ebenfalls merken
d6f0   a9 11       lda  #$11        17 (SA für lesen)
d6f2   85 83       sta  $83         internen lesekanal setzen
d6f4   20 3b de    jsr  $de3b       Track- und Sektornummer holen
d6f7   ad 4a 02    lda  $024a       aktuellen Filetyp
d6fa   48          pha              merken
d6fb   a5 e2       lda  $e2         Drivenummer für neues File
d6fd   29 01       and  #$01        isolieren und
d6ff   85 7f       sta  $7f         als aktuelle Drivenummer übernehmen
d701   a6 f9       ldx  $f9         aktuelle Puffernummer
d703   5d 5b 02    eor  $025b,x     zugehörigen Jobcode prüfen
d706   4a          lsr              Drivenummern identisch?
d707   90 0c       bcc  $d715       verzweige, wenn ja
d709   a2 01       ldx  #$01        Zeiger für
d70b   8e 92 02    stx  $0292       Directory setzen
d70e   20 ac c5    jsr  $c5ac       freien Platz für Eintrag suchen
d711   f0 1d       beq  $d730       verzweige, wenn alles voll; neuen Block!
d713   d0 28       bne  $d73d       unbedingter Sprung; Eintrag schreiben
d715   ad 91 02    lda  $0291       Sektornummer des ersten Eintrags
d718   f0 0c       beq  $d726       verzweige, wenn Null
d71a   c5 81       cmp  $81         gleich der des neuen Eintrags?
d71c   f0 1f       beq  $d73d       verzweige, wenn ja
d71e   85 81       sta  $81         sonst neuen Sektor anlegen
d720   20 60 d4    jsr  $d460       Block lesen
d723   4c 3d d7    jmp  $d73d       Eintrag in Directory schreiben
d726   a9 01       lda  #$01        Suche nach freiem Platz für neuen
d728   8d 92 02    sta  $0292       Eintrag generieren
d72b   20 17 c6    jsr  $c617       z. B. gelöschten Eintrag suchen
d72e   d0 0d       bne  $d73d       verzweige, wenn Platz gefunden
d730   20 8d d4    jsr  $d48d       neuen Directorysektor anlegen
d733   a5 81       lda  $81         Nummer des neuen Sektors
d735   8d 91 02    sta  $0291       als Directorysektor setzen
d738   a9 02       lda  #$02        Zeiger in Directory
d73a   8d 92 02    sta  $0292       auf 2 setzen
d73d   ad 92 02    lda  $0292       Zeiger auf Eintrag
d740   20 c8 d4    jsr  $d4c8       Pointer für diesen Eintrag setzen
d743   68          pla              Filetyp zurückholen
d744   8d 4a 02    sta  $024a       und für Eintrag setzen
d747   c9 04       cmp  #$04        relative Datei?
d749   d0 02       bne  $d74d       verzweige, wenn nein
d74b   09 80       ora  #$80        Bit 7 (File gültig) setzen
d74d   20 f1 cf    jsr  $cff1       Filetyp in Puffer schreiben
d750   68          pla              Tracknummer zurückholen
d751   8d 80 02    sta  $0280       $0280 und für Directory merken
d754   20 f1 cf    jsr  $cff1       und ebenfalls in Puffer schreiben
d757   68          pla              Sektornummer zurückholen
d758   8d 85 02    sta  $0285       und für Directory merken
d75b   20 f1 cf    jsr  $cff1       und ebenfalls in Puffer schreiben
d75e   20 93 df    jsr  $df93       Puffernummer holen
d761   a8          tay              als Index nehmen
d762   ad 7a 02    lda  $027a       Zeiger in Filetabelle
d765   aa          tax              nach X
d766   a9 10       lda  #$10        16, max. länge des Filenamens
```

```
d768   20  6e  c6    jsr   $c66e       Filename in Puffer schreiben
d76b   a0  10       ldy   #$10        Startposition 16
d76d   a9  00       lda   #$00        Füllwert 0
d76f   91  94       sta   ($94),y     restlichen Platz im
d771   c8           iny               Directoryeintrag (bis Position 27)
d772   c0  1b       cpy   #$1b        mit Nullen
d774   90  f9       bcc   $d76f       auffüllen
d776   ad  4a  02    lda   $024a       Filetyp
d779   c9  04       cmp   #$04        relative Datei
d77b   d0  13       bne   $d790       verzweige, wenn nein
d77d   a0  10       ldy   #$10        16 als Index
d77f   ad  59  02    lda   $0259       Track für Side-Sektor-Block
d782   91  94       sta   ($94),y     in Puffer schreiben
d784   c8           iny               17 als Index
d785   ad  5a  02    lda   $025a       Sektor für Side-Sektor-Block
d788   91  94       sta   ($94),y     in Puffer schreiben
d78a   c8           iny               18 als Index
d78b   ad  58  02    lda   $0258       Recordlänge
d78e   91  94       sta   ($94),y     in Puffer schreiben
d790   20  64  d4    jsr   $d464       Block auf Diskette schreiben
d793   68           pla               Kanalnummer zurückholen
d794   85  82       sta   $82         und abspeichern
d796   aa           tax               als Index
d797   68           pla               Sekundäradresse zurückholen
d798   85  83       sta   $83         und wieder übernehmen
d79a   ad  91  02    lda   $0291       Sektornummer des
d79d   85  d8       sta   $d8         Fileeintrags im
d79f   9d  60  02    sta   $0260,x     Directory vermerken
d7a2   ad  92  02    lda   $0292       Zeiger auf
d7a5   85  dd       sta   $dd         Fileeintrag
d7a7   9d  66  02    sta   $0266,x     im Directory merken
d7aa   ad  4a  02    lda   $024a       Filetyp
d7ad   85  e7       sta   $e7         in Tabelle
d7af   a5  7f       lda   $7f         zugehörige Drivenummer
d7b1   85  e2       sta   $e2         ebenfalls in Tabelle schreiben
d7b3   60           rts               Ende
-------------------------------------------------------------------------
d7b4                                  OPEN-Befehl vom seriellen Bus erhalten. Die Sekundäradresse
                                      bewegt sich zwischen 0 und 14, das heißt, öffnen einer
                                      Datei für LOAD, SAVE oder als Datendatei (SEQ/USR/REL).
d7b4   a5  83       lda   $83         Sekundäradresse
d7b6   8d  4c  02    sta   $024c       zwischenspeichern
d7b9   20  b3  c2    jsr   $c2b3       Werte für Befehlsstring setzen
d7bc   8e  2a  02    stx   $022a       X=0 abspeichern
d7bf   ae  00  02    ldx   $0200       erstes Zeichen aus Befehlsstring holen
d7c2   ad  4c  02    lda   $024c       Sekundäradresse
d7c5   d0  2c       bne   $d7f3       verzweige, wenn ungleich 0 (LOAD)
d7c7   e0  2a       cpx   #$2a        ASCII-Code für '*'?
d7c9   d0  28       bne   $d7f3       verzweige, wenn nein
d7cb   a5  7e       lda   $7e         gleich letztem Zugriff?
d7cd   f0  4d       beq   $d81c       verzweige, wenn nein
d7cf   85  80       sta   $80         'alte' Tracknummer übernehmen
d7d1   ad  6e  02    lda   $026e       'alte' Drivenummer
d7d4   85  7f       sta   $7f         übernehmen und für neuen Job
d7d6   85  e2       sta   $e2         setzen
d7d8   a9  02       lda   #$02        Code für Filetyp PRG
d7da   85  e7       sta   $e7         setzen
```

```
d7dc   ad  6f  02    lda   $026f       'alte' Sektornummer
d7df   85  81        sta   $81         übernehmen
d7e1   20  00  c1    jsr   $c100       LED am Laufwerk einschalten
d7e4   20  46  dc    jsr   $dc46       Puffer belegen; Block lesen
d7e7   a9  04        lda   #$04        #$04 Filetyp PRG 'mal 2'
d7e9   05  7f        ora   $7f         mit Drivenummer verknüpfen
d7eb   a6  82        ldx   $82         zugehörige Kanalnummer holen
d7ed   99  ec  00    sta   $00ec,y     und Filetyp in Tabelle eintragen
d7f0   4c  94  c1    jmp   $c194       Diskstatus bereitstellen; Ende

d7f3   e0  24        cpx   #$24        ASCII-Code für '$'?
d7f5   d0  1e        bne   $d815       verzweige, wenn nein
d7f7   ad  4c  02    lda   $024c       Sekundäradresse
d7fa   d0  03        bne   $d7ff       verzweige, wenn ungleich 0 (LOAD)
d7fc   4c  55  da    jmp   $da55       Directory laden; Ende

d7ff   20  d1  c1    jsr   $c1d1       Befehlsstring zu Ende analysieren
d802   ad  85  fe    lda   $fe85       18, Directorytrack
d805   85  80        sta   $80         als aktuelle Tracknummer setzen
d807   a9  00        lda   #$00        0 als Sektornummer für
d809   85  81        sta   $81         die BAM setzen
d80b   20  46  dc    jsr   $dc46       Puffer belegen; Block lesen
d80e   a5  7f        lda   $7f         aktuelle Drivenummer
d810   09  02        ora   #$02        mit Filetyp SEC 'mal 2' verknüpfen
d812   4c  eb  d7    jmp   $d7eb       in Tabelle und - Ende

d815   e0  23        cpx   #$23        ASCII-Code für '#'?
d817   d0  12        bne   $d82b       verzweige, wenn nein
d819   4c  84  cb    jmp   $cb84       Direktzugriffsdatei öffnen; Ende

d81c   a9  02        lda   #$02        Code für Filetyp PRG
d81e   8d  96  02    sta   $0296       setzen und
d821   a9  00        lda   #$00        Drive 0
d823   85  7f        sta   $7f         ebenfalls setzen
d825   8d  8e  02    sta   $028e       Drive für Job setzen
d828   20  42  d0    jsr   $d042       gesetztes Drive initialisieren
d82b   20  e5  c1    jsr   $c1e5       Befehlsstring nach ':' durchsuchen
d82e   d0  04        bne   $d834       verzweige, wenn nicht gefunden
d830   a2  00        ldx   #$00        Index laden
d832   f0  0c        beq   $d840       unbedingter Sprung zur weiteren Analyse
d834   8a            txa               Komma ',' gefunden?
d835   f0  05        beq   $d83c       verzweige, wenn nein
d837   a9  30        lda   #$30        Nummer für Fehlermeldung
d839   4c  c8  c1    jmp   $c1c8       '30, SYNTAX ERROR' ausgeben

d83c   88            dey               Y zeigt jetzt auf den ':' im Befehlsstring
d83d   f0  01        beq   $d840       verzweige, wenn ':' am Anfang steht
d83f   88            dey               sonst y wieder minus 1
d840   8c  7a  02    sty   $027a       y zeigt jetzt auf Drivenummer; merken
d843   a9  8d        lda   #$8d        'SHIFT RETURN'
d845   20  68  c2    jsr   $c268       Befehlsstring bis zum Ende analysieren
d848   e8            inx               Anzahl der enthaltenen Kommata plus 1
d849   8e  78  02    stx   $0278       als Kommazähler merken
d84c   20  12  c3    jsr   $c312       Driveparameter setzen
d84f   20  ca  c3    jsr   $c3ca       Drivenummer(n) prüfen
d852   20  9d  c4    jsr   $c49d       Eintrag in Directory suchen
d855   a2  00        ldx   #$00        Parameter löschen:
d857   8e  58  02    stx   $0258       Recordlänge
```

```
d85a   8e 97 02   stx  $0297      Betriebsart des Files
d85d   8e 4a 02   stx  $024a      Filetyp
d860   e8         inx            X ist jetzt 1
d861   ec 77 02   cpx  $0277      'Sonderzeichen' (*,?,...) im String?
d864   b0 10      bcs  $d876      verzweige, wenn nein
d866   20 09 da   jsr  $da09      Filetyp und -betriebsart holen
d869   e8         inx            X ist jetzt 2
d86a   ec 77 02   cpx  $0277      'Sonderzeichen' im String?
d86d   b0 07      bcs  $d876      verzweige, wenn nur ein Zeichen
d86f   c0 04      cpy  #$04       relatives File?
d871   f0 3e      beq  $d8b1      verzweige, wenn ja
d873   20 09 da   jsr  $da09      Filetyp und -betriebsart holen
d876   ae 4c 02   ldx  $024c      Sekundäradresse zurückholen
d879   86 83      stx  $83        und wieder abspeichern
d87b   e0 02      cpx  #$02       größer gleich 2?
d87d   b0 12      bcs  $d891      verzweige, wenn ja
d87f   8e 97 02   stx  $0297      0 oder 1 (LOAD oder SAVE) setzen
d882   a9 40      lda  #$40       'BAM dirty' Zustand anzeigen
d884   8d f9 02   sta  $02f9      BAM wurde geändert
d887   ad 4a 02   lda  $024a      Filetyp
d88a   d0 1b      bne  $d8a7      verzweige, wenn nicht DEL-File
d88c   a9 02      lda  #$02       Code für PRG-File
d88e   8d 4a 02   sta  $024a      setzen
d891   ad 4a 02   lda  $024a      Filetyp
d894   d0 11      bne  $d8a7      verzweige, wenn nicht DEL-File
d896   a5 e7      lda  $e7        Filetyp aus Tabelle holen
d898   29 07      and  #$07       isolieren, und
d89a   8d 4a 02   sta  $024a      als aktuell übernehmen
d89d   ad 80 02   lda  $0280      lda $0280 Tracknummer dieses Files
d8a0   d0 05      bne  $d8a7      verzweige, wenn vorhanden
d8a2   a9 01      lda  #$01       Code für Filetyp SEQ
d8a4   8d 4a 02   sta  $024a      übernehmen
d8a7   ad 97 02   lda  $0297      Filebetriebsart
d8aa   c9 01      cmp  #$01       SCHREIBEN?
d8ac   f0 18      beq  $d8c6      verzweige, wenn ja
d8ae   4c 40 d9   jmp  $d940      Kanal für LOAD oder READ öffnen; Ende

d8b1   bc 7a 02   ldy  $027a,x    Zeiger in Befehlsstring hinter zweites Komma
d8b4   b9 00 02   lda  $0200,y    Recordlänge aus String holen
d8b7   8d 58 02   sta  $0258      und übernehmen
d8ba   ad 80 02   lda  $0280      Tracknummer des Files
d8bd   d0 b7      bne  $d876      verzweige, wenn vorhanden
d8bf   a9 01      lda  #$01       Code für WRITE
d8c1   8d 97 02   sta  $0297      als Filebetriebsart setzen
d8c4   d0 b0      bne  $d876      unbedingter Sprung
d8c6   a5 e7      lda  $e7        Filetypparameter aus Tabelle holen
d8c8   29 80      and  #$80       prüfen, ob Datei vorhanden
d8ca   aa         tax            Ergebnis merken
d8cb   d0 14      bne  $d8e1      verzweige, wenn ja
d8cd   a9 20      lda  #$20       Filenamen auf Joker
d8cf   24 e7      bit  $e7        testen
d8d1   f0 06      beq  $d8d9      verzweige, wenn Joker (?,*) vorhanden
d8d3   20 b6 c8   jsr  $c8b6      Eintrag löschen; Block schreiben
d8d6   4c e3 d9   jmp  $d9e3      neuen Block anlegen; Ende

d8d9   ad 80 02   lda  $0280      Tracknummer des Fileeintrags
d8dc   d0 03      bne  $d8e1      verzweige, wenn File vorhanden
d8de   4c e3 d9   jmp  $d9e3      neuen Block anlegen; Ende
```

```
d8e1   ad  00  02   lda   $0200       erstes Zeichen aus Befehlsstring holen
d8e4   c9  40       cmp   #$40        ASCII-Code für '@' (Klammeraffe=REPLACE)
d8e6   f0  0d       beq   $d8f5       beq $dBf5 verzweige, wenn ja
d8e8   8a           txa               vorhanden?
d8e9   d0  05       bne   $d8f0       verzweige, wenn nein
d8eb   a9  63       lda   #$63        Nummer der Fehlermeldung
d8ed   4c  c8  c1   jmp   $c1c8       '63, FILE EXISTS' ausgeben
d8f0   a9  33       lda   #$33        Nummer der Fehlermeldung
d8f2   4c  c8  c1   jmp   $c1c8       '33, SYNTAX ERROR' ausgeben
-------------------------------------------------------------------------
d8f5                                  Öffnen eines Files mit Überschreiben.
d8f5   a5  e7       lda   $e7         Filetyp aus Tabelle
d8f7   29  07       and   #$07        isolieren
d8f9   cd  4a  02   cmp   $024a       gleich aktuellem Filetyp?
d8fc   d0  67       bne   $d965       verzweige, wenn nein
d8fe   c9  04       cmp   #$04        relatives File?
d900   f0  63       beq   $d965       verzweige, wenn ja
d902   20  da  dc   jsr   $dcda       neuen Sektor anlegen
d905   a5  82       lda   $82         aktuelle Kanalnummer
d907   8d  70  02   sta   $0270       als aktuellen Schreibkanal merken
d90a   a9  11       lda   #$11        17 (READ)
d90c   85  83       sta   $83         internen Lesekanal setzen
d90e   20  eb  d0   jsr   $d0eb       lesekanal suchen und öffnen
d911   ad  94  02   lda   $0294       aktueller Pufferzeiger
d914   20  c8  d4   jsr   $d4c8       Zeiger für Directory setzen
d917   a0  00       ldy   #$00        Pufferindex auf 0
d919   b1  94       lda   ($94),y     Filetyp aus Puffer holen
d91b   09  20       ora   #$20        File als 'offen' deklarieren
d91d   91  94       sta   ($94),y     Filetyp wieder schreiben
d91f   a0  1a       ldy   #$1a        26 als Position im Fileeintrag
d921   a5  80       lda   $80         aktuelle Tracknummer
d923   91  94       sta   ($94),y     setzen
d925   c8           iny               27
d926   a5  81       lda   $81         aktuelle Sektornummer
d928   91  94       sta   ($94),y     ebenfalls eintragen
d92a   ae  70  02   ldx   $0270       Kanalnummer
d92d   a5  d8       lda   $d8         Sektor des Fileeintrags
d92f   9d  60  02   sta   $0260,x     übernehmen
d932   a5  dd       lda   $dd         Zeiger in Fileeintrag
d934   9d  66  02   sta   $0266,x     ebenfalls übernehmen
d937   20  3b  de   jsr   $de3b       Track und Sektor setzen
d93a   20  64  d4   jsr   $d464       Block auf Diskette schreiben
d93d   4c  ef  d9   jmp   $d9ef       Datei ist jetzt geöffnet; Ende
d940   ad  80  02   lda   $0280       Tracknummer des Files
d943   d0  05       bne   $d94a       verzweige, wenn File vorhanden
d945   a9  62       lda   #$62        Nummer der Fehlermeldung
d947   4c  c8  c1   jmp   $c1c8       '62, FILE NOT FOUND' ausgeben
d94a   ad  97  02   lda   $0297       Filebetriebsart
d94d   c9  03       cmp   #$03        MODIFY?
d94f   f0  0b       beq   $d95c       verzweige, wenn ja
d951   a9  20       lda   #$20        Datei auf bereits geöffnet
d953   24  e7       bit   $e7         testen
d955   f0  05       beq   $d95c       verzwge, wenn File nicht offen
d957   a9  60       lda   #$60        Nummer der Fehlermeldung
d959   4c  c8  c1   jmp   $c1c8       '60, WRITE FILE OPEN' ausgeben
d95c   a5  e7       lda   $e7         Filetyp aus Tabelle
d95e   29  07       and   #$07        isolieren
d960   cd  4a  02   cmp   $024a       gleich dem Filetyp aus dem Befehl?
```

```
d963   f0  05       beq   $d96a       verzweige, wenn ja
d965   a9  64       lda   #$64        Nummer der Fehlermeldung
d967   4c  c8  c1    jmp   $c1c8       '64, FILE TYPE MISMATCH' ausgeben
d96a   a0  00       ldy   #$00        Zeiger
d96c   8c  79  02    sty   $0279       löschen
d96f   ae  97  02    ldx   $0297       Filebetriebsart
d972   e0  02       cpx   #$02        APPEND?
d974   d0  1a       bne   $d990       Fehler, wenn ja
d976   c9  04       cmp   #$04        relative Datei?
d978   f0  eb       beq   $d965       Fehler, wenn ja
d97a   b1  94       lda   ($94),y     Filetyp aus Puffer holen
d97c   29  4f       and   #$4f        File als offen deklarieren
d97e   91  94       sta   ($94),y     Filetyp wieder abspeichern
d980   a5  83       lda   $83         Sekundäradresse
d982   48          pha               merken
d983   a9  11       lda   #$11        17 (READ)
d985   85  83       sta   $83         als neue Sekundäradresse setzen
d987   20  3b  de    jsr   $de3b       Track- und Sektor für Lesekanal holen
d98a   20  64  d4    jsr   $d464       Block auf Diskette schreiben
d98d   68          pla               Sekundäradresse zurückholen
d98e   85  83       sta   $83         und wieder setzen
d990   20  a0  d9    jsr   $d9a0       File zum Lesen öffnen
d993   ad  97  02    lda   $0297       Filebetriebsart
d996   c9  02       cmp   #$02        APPEND?
d998   d0  55       bne   $d9ef       verzweige, wenn nein
d99a   20  2a  da    jsr   $da2a       Ende des Files für APPEND suchen
d99d   4c  94  c1    jmp   $c194       Diskstatus bereitstellen; Ende
-------------------------------------------------------------------------------
d9a0                                   File zum Lesen öffnen.
d9a0   a0  13       ldy   #$13        19 als Pufferposition
d9a2   b1  94       lda   ($94),y     Track des Side-Sektor-Blocks
d9a4   8d  59  02    sta   $0259       merken
d9a7   c8          iny               20
d9a8   b1  94       lda   ($94),y     Sektor des Side-Sektor-Blocks
d9aa   8d  5a  02    sta   $025a       merken
d9ad   c8          iny               21
d9ae   b1  94       lda   ($94),y     Recordlänge
d9b0   ae  58  02    ldx   $0258       letzte Recordlänge retten
d9b3   8d  58  02    sta   $0258       neue Recordlänge übernehmen
d9b6   8a          txa               letzte Recordlänge
d9b7   f0  0a       beq   $d9c3       verzweige, wenn Null
d9b9   cd  58  02    cmp   $0258       mit neuer Länge vergleichen
d9bc   f0  05       beq   $d9c3       verzweige, wenn beide Längen gleich
d9be   a9  50       lda   #$50        Nummer der Fehlermeldung
d9c0   20  c8  c1    jsr   $c1c8       '50, RECORD NOT PRESENT' ausgeben
d9c3   ae  79  02    ldx   $0279       Zeiger auf Fileparameter
d9c6   bd  80  02    lda   $0280,x     Tracknummer des Files
d9c9   85  80       sta   $80         übernehmen
d9cb   bd  85  02    lda   $0285,x     Sektornummer des Files
d9ce   85  81       sta   $81         übernehmen
d9d0   20  46  dc    jsr   $dc46       Kanal öffnen; Block lesen
d9d3   a4  82       ldy   $82         aktuelle Kanalnummer
d9d5   ae  79  02    ldx   $0279       Zeiger auf Fileparameter
d9d8   b5  d8       lda   $d8,x       Sektornummer des Fileeintrags
d9da   99  60  02    sta   $0260,y     übernehmen
d9dd   b5  dd       lda   $dd,x       Zeiger in Fileeintrag
d9df   99  66  02    sta   $0266,y     übernehmen
d9e2   60          rts               Ende
```

```
------------------------------------------------------------------------
d9e3                                    File zum Schreiben öffnen.
d9e3   a5  e2        lda   $e2          Drivenummer
d9e5   29  01        and   #$01         isolieren und
d9e7   85  7f        sta   $7f          als aktuell übernehmen
d9e9   20  da  dc    jsr   $dcda        $dcda Block anlegen
d9ec   20  e4  d6    jsr   $d6e4        $d6e4 File im Directory eintragen
d9ef   a5  83        lda   $83          Sekundäradresse
d9f1   c9  02        cmp   #$02         größer gleich 0 oder 11
d9f3   b0  11        bcs   $da06        verzweige, wenn ja
d9f5   20  3e  de    jsr   $de3e        Track und Sektor holen
d9f8   a5  80        lda   $80          aktuelle Tracknummer
d9fa   85  7e        sta   $7e          für weitere Aktionen festlegen
d9fc   a5  7f        lda   $7f          Drivenummer
d9fe   8d  6e  02    sta   $026e        für weitere Aktionen festlegen
da01   a5  81        lda   $81          Sektornummer
da03   8d  6f  02    sta   $026f        für weitere Aktionen festlegen
da06   4c  99  c1    jmp   $c199        Diskstatus bereitstellen; Ende
------------------------------------------------------------------------
da09                                    Filebetriebsarten prüfen und Parameter setzen
da09   bc  7a  02    ldy   $027a,x      Zeiger in Befehlszeile
da0c   b9  00  02    lda   $0200,y      Zeichen aus Befehlsstring holen
da0f   a0  04        ldy   #$04         Anzahl der möglichen Betriebsarten
da11   88            dey                minus 1; alle testen
da12   30  08        bmi   $da1c        weiter, wenn alle geprüft
da14   d9  b2  fe    cmp   $feb2,y      Filemodi 'R','W','A' und 'M'
da17   d0  f8        bne   $da11        nicht gefunden; weitersuchen
da19   8c  97  02    sty   $0297        Nummer des Modus merken
da1c   a0  05        ldy   #$05         maximale Anzahl der Filetypen
da1e   88            dey                minus 1; alle testen
da1f   30  08        bmi   $da29        weiter, wenn alle geprüft
da21   d9  b6  fe    cmp   $feb6,y      Filetypen 'D','S','P', 'U' und 'L'
da24   d0  f8        bne   $da1e        nicht gefunden; weitersuchen
da26   8c  4a  02    sty   $024a        Nummer des Filetyps merken
da29   60            rts                Ende
------------------------------------------------------------------------
da2a                                    File für APPEND vorbereiten und vorherigen Datenteil
                                        überlesen.
da2a   20  39  ca    jsr   $ca39        Byte über den Lesekanal holen
da2d   a9  80        lda   #$80         Test, ob letztes
da2f   20  a6  dd    jsr   $dda6        Byte gelesen wurde
da32   f0  f6        beq   $da2a        verzweige, wenn nein
da34   20  95  de    jsr   $de95        Track und Sektor holen
da37   a6  81        ldx   $81          Sektornummer
da39   e8            inx                auf $ff (Maximalwert)
da3a   8a            txa                prüfen
da3b   d0  05        bne   $da42        verzweige, wenn nicht $ff
da3d   20  a3  d1    jsr   $d1a3        nächsten Sektor holen
da40   a9  02        lda   #$02         Wert für Pufferzeiger
da42   20  c8  d4    jsr   $d4c8        Pufferzeiger auf 2 setzen (1. Datenbyte)
da45   a6  82        ldx   $82          Kanalnummer
da47   a9  01        lda   #$01         Flag für WRITE
da49   95  f2        sta   $f2,x        in Tabelle eintragen
da4b   a9  80        lda   #$80         Flag für WRITE
da4d   05  82        ora   $82          verbunclen mit Kanalnummer
da4f   a6  83        ldx   $83          und Sekundäradresse als Index
da51   9d  2b  02    sta   $022b,x      in Statustabelle eintragen
da54   60            rts                Ende
------------------------------------------------------------------------
da55                                    Directory von Diskette laden
```

```
da55   a9  0c       lda   #$0c        12 als Befehlsnummer
da57   8d  2a  02    sta   $022a       setzen
da5a   a9  00       lda   #$00        Null als Default für Drivenummer
da5c   ae  74  02    ldx   $0274       Länge des Befehlsstrings
da5f   ca           dex               gleich 1?
da60   f0  0b       beq   $da6d       verzweige, wenn ja
da62   ca           dex               gleich 2?
da63   d0  21       bne   $da86       verzweige, wenn nein
da65   ad  01  02    lda   $0201       zweites Zeichen aus Befehlsstring
da68   20  bd  c3    jsr   $c3bd       auf Drivenummer prüfen
da6b   30  19       bmi   $da86       verzweige, wenn Drivenummer unklar
da6d   85  e2       sta   $e2         Drivenummer in Tabelle
da6f   ee  77  02    inc   $0277       Zeiger in Befehlsstring
da72   ee  78  02    inc   $0278       allesamt
da75   ee  7a  02    inc   $027a       erhöhen
da78   a9  80       lda   #$80        gültigen Filetyp
da7a   85  e7       sta   $e7         setzen
da7c   a9  2a       lda   #$2a        ASCII-Coce für '*'
da7e   8d  00  02    sta   $0200       als Filename
da81   8d  01  02    sta   $0201       in Befehlsstring
da84   d0  18       bne   $da9e       unbedingter Sprung
da86   20  e5  c1    jsr   $c1e5       ':' in Befehlsstring suchen
da89   d0  05       bne   $da90       verzweige, wenn nicht gefunden
da8b   20  dc  c2    jsr   $c2dc       Parameter für Befehlsstring löschen
da8e   a0  03       ldy   #$03        Y-Wert 3 durch zweimaliges Abziehen
da90   88           dey               auf 1 setzen;
da91   88           dey               anderer Einsprung von $da89
da92   8c  7a  02    sty   $027a       Befehlsstringzeiger ist jetzt 1!
da95   20  00  c2    jsr   $c200       Befehlsstring analysieren
da98   20  98  c3    jsr   $c398       Filetyp und -parameter setzen
da9b   20  20  c3    jsr   $c320       Drivenummer(n) holen
da9e   20  ca  c3    jsr   $c3ca       Drive(s) initialisieren
daa1   20  b7  c7    jsr   $c7b7       erste Zeile für Directory erzeugen
daa4   20  9d  c4    jsr   $c49d       Fileeinträge holen
daa7   20  9e  ec    jsr   $ec9e       Format der Einträge herstellen
daaa   20  37  d1    jsr   $d137       Byte aus Puffer holen
daad   a6  82       ldx   $82         aktuelle Kanalnummer
daaf   9d  3e  02    sta   $023e,x     Byte aus Tabelle für Ausgabe bereitstellen
dab2   a5  7f       lda   $7f         aktuelle Drivenummer
dab4   8d  8e  02    sta   $028e       als letzte Drivenummer merken
dab7   09  04       ora   #$04        Wert für PRG-File ergänzen und
dab9   95  ec       sta   $ec,x       Filetyp in Tabelle setzen
dabb   a9  00       lda   #$00        Zeiger in INPUT-Puffer
dabd   85  a3       sta   $a3         löschen
dabf   60           rts               Ende
-------------------------------------------------------------------------
dac0                                  File mit gegebener Sekundäradresse schließen.
dac0   a9  00       lda   #$00        Flag für 'BAM nicht schreiben'
dac2   8d  f9  02    sta   $02f9       setzen
dac5   a5  83       lda   $83         aktuelle Sekundäradresse
dac7   d0  0b       bne   $dad4       verzweige, wenn ungleich 0 (LOAD)
dac9   a9  00       lda   #$00        Flag für Directory-Listing
dacb   8d  54  02    sta   $0254       löschen
dace   20  27  d2    jsr   $d227       Kanal mit gegebener Nummer schließen
dad1   4c  da  d4    jmp   $d4da       interne Kanäle schließen; Ende
dad4   c9  0f       cmp   #$0f        15 (Kommandokanal)?
```

```
dad6   f0  14       beq   $daec       beqx $daec verzweige, wenn ja
dad8   20  02  db   jsr   $db02       Datei schließen
dadb   a5  83       lda   $83         Sekundäradresse
dadd   c9  02       cmp   #$02        größer gleich 0 oder 1?
dadf   90  f0       bcc   $dad1       verzweige, wenn nein
dae1   ad  6c  02   lda   $026c       Fehler beim letzten Befehl?
dae4   d0  03       bne   $dae9       verzweige, wenn ja
dae6   4c  94  c1   jmp   $c194       Diskstatus bereitstellen; Ende
dae9   4c  ad  c1   jmp   $c1ad       Fehler anzeigen; Ende
-------------------------------------------------------------------------
daec                                  Alle Files schließen, wenn der Kommandokanal geschlossen
                                      werden soll.
daec   a9  0e       lda   #$0e        14 als höchste Sekundäradresse
daee   85  83       sta   $83         setzen
daf0   20  02  db   jsr   $db02       zugehörige Datei schließen
daf3   c6  83       dec   $83         nächste Sekundäradresse
daf5   10  f9       bpl   $daf0       weitermachen, wenn vorhanden (bis 0)
daf7   ad  6c  02   lda   $026c       Fehler beim letzten Befehl?
dafa   d0  03       bne   $daff       verzweige, wenn ja
dafc   4c  94  c1   jmp   $c194       Diskstatus bereitstellen; Ende
daff   4c  ad  c1   jmp   $c1ad       Fehler anzeigen; Ende
-------------------------------------------------------------------------
db02                                  File mit bestimmter Sekundäradresse schließen; Datei
                                      schließen.
db02   a6  83       ldx   $83         Sekundäradresse
db04   bd  2b  02   lda   $022b,x     zugehörigen Kanalstatus holen
db07   c9  ff       cmp   #$ff        Kanal benutzt?
db09   d0  01       bne   $db0c       verzweige, wenn ja
db0b   60           rts               sonst Ende
db0c   29  0f       and   #$0f        Kanalnummer isolieren
db0e   85  82       sta   $82         und abspeichern
db10   20  25  d1   jsr   $d125       Filetyp prüfen
db13   c9  07       cmp   #$07        Direktzugriffsdatei ('#')?
db15   f0  0f       beq   $db26       verzweige, wenn ja
db17   c9  04       cmp   #$04        relative Datei?
db19   f0  11       beq   $db2c       verzweige, wenn ja
db1b   20  07  d1   jsr   $d107       Kanal zum Schreiben öffnen
db1e   b0  09       bcs   $db29       verzweige, wenn kein Schreibkanal verfügbar
db20   20  62  db   jsr   $db62       Schreibvorgang abschließen
db23   20  a5  db   jsr   $dba5       Eintrag im Directory abschließen
db26   20  f4  ee   jsr   $eef4       BAM auf die Diskette schreiben
db29   4c  27  d2   jmp   $d227       Kanal schließen; Ende
db2c   20  f1  dd   jsr   $ddf1       BAM schreiben, wenn 'dirty', d.h. geändert
db2f   20  1e  cf   jsr   $cf1e       Zweipufferbetrieb einrichten
db32   20  cb  e1   jsr   $e1cb       Zeiger auf letzten Record stellen
db35   a6  d5       ldx   $d5         Nummer des Side-Sektor-Blocks
db37   86  73       stx   $73         merken
db39   e6  73       inc   $73         und plus 1
db3b   a9  00       lda   #$00        Zwischenspeicher für weitere
db3d   85  70       sta   $70         Bearbeitung
db3f   85  71       sta   $71         löschen
db41   a5  d6       lda   $d6         Zeiger in Side-Sektor
db43   38           sec               Subtraktion vorbereiten
db44   e9  0e       sbc   #$0e        und 14 vom Zeiger abziehen
db46   85  72       sta   $72         Ergebnis speichern
db48   20  51  df   jsr   $df51       Anzahl der Blöcke berechnen
db4b   a6  82       ldx   $82         Kanalnummer holen
db4d   a5  70       lda   $70         Rechenergebnis ist gleich
```

```
db4f   95  b5       sta   $b5,x       Recordnummer Lo
db51   a5  71       lda   $71         Rechenergebnis ist gleich
db53   95  bb       sta   $bb,x       Recordnummer Hi
db55   a9  40       lda   #$40        Dirty-Flag für Record prüfen
db57   20  a6  dd   jsr   $dda6       gesetzt?
db5a   f0  03       beq   $db5f       verzweige, wenn nein
db5c   20  a5  db   jsr   $dba5       Directoryeintrag abschließen
db5f   4c  27  d2   jmp   $d227       Kanal schließen; Ende
-------------------------------------------------------------------------
db62                                  Letzten BLock einer Datei schreiben
db62   a6  82       ldx   $82         Kanalnummer
db64   b5  b5       lda   $b5,x       Recordnummer Lo
db66   15  bb       ora   $bb,x       mit Recordnummer Hi verknüpfen
db68   d0  0c       bne   $db76       verzweige, wenn ungleich 0
db6a   20  e8  d4   jsr   $d4e8       Pufferzeiger holen
db6d   c9  02       cmp   #$02        2 (noch kein Byte im Block)?
db6f   d0  05       bne   $db76       verzweige, wenn nein
db71   a9  0d       lda   #$0d        sonst 'RETURN'
db73   20  f1  cf   jsr   $cff1       in den Puffer schreiben
db76   20  e8  d4   jsr   $d4e8       und Pufferzeiger holen
db79   c9  02       cmp   #$02        2 (noch kein Byte im Block)?
db7b   d0  0f       bne   $db8c       verzweige, wenn nein
db7d   20  1e  cf   jsr   $cf1e       Puffer wechseln
db80   a6  82       ldx   $82         Kanalnummer
db82   b5  b5       lda   $b5,x       Recordnummer La
db84   d0  02       bne   $db88       verzweige, wenn ungleich Null
db86   d6  bb       dec   $bb,x       Recordnummer Hi und
db88   d6  b5       dec   $b5,x       Recordnummer Lo vermindern
db8a   a9  00       lda   #$00        Default für Anzahl der Bytes im Block
db8c   38           sec               sonst Pufferzeiger
db8d   e9  01       sbc   #$01        minus 1
db8f   48           pha               als Anzahl der Bytes merken
db90   a9  00       lda   #$00        Pufferzeiger auf Null
db92   20  c8  d4   jsr   $d4c8       setzen
db95   20  f1  cf   jsr   $cff1       $00 als Folgetrack in Puffer schreiben
db98   68           pla               Anzahl der Byte zurückholen
db99   20  f1  cf   jsr   $cff1       und ebenfalls in Puffer schreiben
db9c   20  c7  d0   jsr   $d0c7       Pufferinhalt als Block auf Diskette
db9f   20  99  d5   jsr   $d599       Schreibjob abwarten und auf Fehler testen
dba2   4c  1e  cf   jmp   $cf1e       Puffer wechseln; Ende
-------------------------------------------------------------------------
dba5                                  Directory nach dem Schreiben eines Files aktualisieren.
dba5   a6  82       ldx   $82         Kanalnummer
dba7   8e  70  02   stx   $0270       merken
dbaa   a5  83       lda   $83         Sekundäradresse
dbac   48           pha               merken
dbad   bd  60  02   lda   $0260,x     Sektornummer im Directory
dbb0   85  81       sta   $81         übernehmen
dbb2   bd  66  02   lda   $0266,x     Zeiger in Directory
dbb5   8d  94  02   sta   $0294       übernehmen
dbb8   b5  ec       lda   $ec,x       Filetyp für Kanalnummer holen
dbba   29  01       and   #$01        Drivenummer isolieren
dbbc   85  7f       sta   $7f         und übernehmen
dbbe   ad  85  fe   lda   $fe85       18; Track für Directory
dbc1   85  80       sta   $80         übernehmen
dbc3   20  93  df   jsr   $df93       Puffernummer holen
dbc6   48           pha               und merken
```

```
dbc7   85  f9       sta   $f9         und als aktuell übernehmen
dbc9   20  60  d4    jsr   $d460       Directorysektor von Diskette .5
dbcc   a0  00       ldy   #$00        Pufferindex auf 0
dbce   bd  e0  fe    lda   $fee0,x     Pufferadresse Hi holen und
dbd1   85  87       sta   $87         übernehmen
dbd3   ad  94  02    lda   $0294       Pufferzeiger als Adresse Lo
dbd6   85  86       sta   $86         übernehmen
dbd8   b1  86       lda   ($86),y     Byte aus Puffer (Filetyp) holen
dbda   29  20       and   #$20        Datei geschlossen?
dbdc   f0  43       beq   $dc21       verzweige, wenn nein
dbde   20  25  d1    jsr   $d125       sonst Filetyp prüfen
dbe1   c9  04       cmp   #$04        relative Datei?
dbe3   f0  44       beq   $dc29       verzweige, wenn ja
dbe5   b1  86       lda   ($86),y     Filetyp nochmal aus Puffer holen
dbe7   29  8f       and   #$8f        Bits 4,5 und 6 löschen
dbe9   91  86       sta   ($86),y     und Filetyp 'neutral' wieder abspeichern
dbeb   c8           iny               Zeiger erhöhen
dbec   b1  86       lda   ($86),y     Tracknummer des Files holen
dbee   85  80       sta   $80         und übernehmen
dbf0   84  71       sty   $71         Zeiger in Puffer merken
dbf2   a0  1b       ldy   #$1b        27
dbf4   b1  86       lda   ($86),y     Sektornummer für REPLACE-Befehl
dbf6   48           pha               merken
dbf7   88           dey               26
dbf8   b1  86       lda   ($86),y     Tracknummer für REPLACE-Befehl
dbfa   d0  0a       bne   $dc06       verzweige, wenn vorhanden
dbfc   85  80       sta   $80         sonst (0) setzen
dbfe   68           pla               Sektornummer zurückholen
dbff   85  81       sta   $81         und ebenfalls übernehmen
dc01   a9  67       lda   #$67        Nummer der Fehlermeldung
dc03   20  45  e6    jsr   $e645       '64, ILLEGAL TRACK OR SECTOR' ausgeben
dc06   48           pha               Tracknummer für REPLACE-Befehl
dc07   a9  00       lda   #$00        Tracknummer
dc09   91  86       sta   ($86),y     löschen
dc0b   c8           iny               27
dc0c   91  86       sta   ($86),y     Sektornummer löschen
dc0e   68           pla               Tracknummer zurückholen
dc0f   a4  71       ldy   $71         Zeiger zurückholen
dc11   91  86       sta   ($86),y     Tracknummer bei REPLACE setzen
dc13   c8           iny               Zeiger erhöhen
dc14   b1  86       lda   ($86),y     Sektornummer holen
dc16   85  81       sta   $81         und übernehmen
dc18   68           pla               Sektornummer für REPLACE
dc19   91  86       sta   ($86),y     setzen
dc1b   20  7d  c8    jsr   $c87d       alte Datei in der BAM löschen
dc1e   4c  29  dc    jmp   $dc29       Datei schließen; Ende
dc21   b1  86       lda   ($86),y     Filetyp holen
dc23   29  0f       and   #$0f        Bits 4,5 und 6 löschen
dc25   09  80       ora   #$80        Bit 7 setzen (Kennzeichen für
dc27   91  86       sta   ($86),y     ein geschlossenes File)
dc29   ae  70  02    ldx   $0270       Kanalnummer holen
dc2c   a0  1c       ldy   #$1c        28
dc2e   b5  b5       lda   $b5,x       Blockzahl Lo
dc30   91  86       sta   ($86),y     in Directory schreiben
dc32   c8           iny               29
dc33   b5  bb       lda   $bb,x       Blockzahl Hi
dc35   91  86       sta   ($86),y     in Directory schreiben
dc37   68           pla               Puffernummer zurückholen
```

```
dc38   aa          tax             als Index
dc39   a9  90      lda   #$90      Code für Schreiben
dc3b   05  7f      ora   $7f       mit Drivenummer verknüpfen
dc3d   20  90  d5  jsr   $d590     und an Diskcontroller
dc40   68          pla             Sekundäradresse zurückholen
dc41   85  83      sta   $83       und wieder abspeichern
dc43   4c  07  d1  jmp   $d107     Kanal zum Schreiben öffnen; Ende
--------------------------------------------------------------------------------
dc46                               Kanal zum Lesen mit zwei Puffern öffnen und Block lesen.
dc46   a9  01      lda   #$01      einen Kanal
dc48   20  e2  d1  jsr   $d1e2     zum Lesen öffnen
dc4b   20  b6  dc  jsr   $dcb6     Zeiger zurücksetzen
dc4e   ad  4a  02  lda   $024a     Filetyp
dc51   48          pha             merken
dc52   0a          asl             Filetyp mit
dc53   05  7f      ora   $7f       Drivenummer verknüpfen
dc55   95  ec      sta   $ec,x     und in Tabelle eintragen
dc57   20  9b  d0  jsr   $d09b     Block von Diskette lesen
dc5a   a6  82      ldx   $82       Kanalnummer
dc5c   a5  80      lda   $80       und Tracknummer
dc5e   d0  05      bne   $dc65     verzweige, wenn kein Folgetrack
dc60   a5  81      lda   $81       kein Folgetrack; Anzahl der Bytes
dc62   9d  44  02  sta   $0244,x   als Endekennzeichen merken
dc65   68          pla             Filetyp zurückholen
dc66   c9  04      cmp   #$04      relative Datei?
dc68   d0  3f      bne   $dca9     verzweige, wenn nein
dc6a   a4  83      ldy   $83       Sekundäradresse
dc6c   b9  2b  02  lda   $022b,y   zugehörige Kanalnummer
dc6f   09  40      ora   #$40      READ/WRITE-Modus setzen
dc71   99  2b  02  sta   $022b,y   und wieder abspeichern
dc74   ad  58  02  lda   $0258     Recordlänge
dc77   95  c7      sta   $c7,x     in Tabelle schreiben
dc79   20  8e  d2  jsr   $d28e     Puffer für Side-Sektor-Block suchen
dc7c   10  03      bpl   $dc81     verzweige, wenn gefunden
dc7e   4c  0f  d2  jmp   $d20f     '70, NO CHANNEl' ausgeben
dc81   a6  82      ldx   $82       Kanalnummer
dc83   95  cd      sta   $cd,x     Puffernummer für Side-Sektor merken
dc85   ac  59  02  ldy   $0259     Tracknummer des Side-Sektor-Blocks
dc88   84  80      sty   $80       übernehmen
dc8a   ac  5a  02  ldy   $025a     Sektor des Side-Sektor-Blocks
dc8d   84  81      sty   $81       übernehmen
dc8f   20  d3  d6  jsr   $d6d3     Parameter an Diskcontroller übergeben
dc92   20  73  de  jsr   $de73     Side-Sektor-Block lesen
dc95   20  99  d5  jsr   $d599     Jobausführung abwarten und prüfen
dc98   a6  82      ldx   $82       Kanalnummer
dc9a   a9  02      lda   #$02      nächsten zu bearbeitenden Record
dc9c   95  c1      sta   $c1,x     setzen
dc9e   a9  00      lda   #$00      Pufferzeiger
dca0   20  c8  d4  jsr   $d4c8     auf Null setzen
dca3   20  53  e1  jsr   $e153     ersten Record suchen
dca6   4c  3e  de  jmp   $de3e     Track und Sektor holen; Ende
dca9   20  56  d1  jsr   $d156     Byte aus Puffer holen
dcac   a6  82      ldx   $82       Kanalnummer als Index
dcae   9d  3e  02  sta   $023e,x   Byte für Ausgabe bereitstellen
dcb1   a9  88      lda   #$88      Flag für 'READY TO TALK' setzen
dcb3   95  f2      sta   $f2,x     und als Kanalstatus übernehmen
dcb5   60          rts             Ende
```

```
------------------------------------------------------------------------
dcb6                                    Parameter zum Öffnen eines Kanals setzen.
dcb6    a6  82      ldx   $82          Kanalnummer
dcb8    b5  a7      lda   $a7,x        zugehörige Puffernummer holen
dcba    0a          asl               mal zwei rechnen und
dcbb    30  06      bmi   $dcc3       verzweigen, wenn größer 127
dcbd    a8          tay               sonst als Index nehmen
dcbe    a9  02      lda   #$02        Pufferzeiger Lo
dcc0    99  99  00  sta   $0099,y     auf 2
dcc3    b5  ae      lda   $ae,x        Puffernummer aus zweiter Tabelle
dcc5    09  80      ora   #$80        Bit 7
dcc7    95  ae      sta   $ae,x        Puffer frei setzen
dcc9    0a          asl               Puffernummer mal zwei
dcca    30  06      bmi   $dcd2       verzweige, wenn größer 127
dccc    a8          tay               sonst als Index
dccd    a9  02      lda   #$02        Pufferzeiger Lo
dccf    99  99  00  sta   $0099,y     auf 2 setzen
dcd2    a9  00      lda   #$00        Löschen:
dcd4    95  b5      sta   $b5,x        Blockzahl Lo
dcd6    4c  7f  a9  jmp   $a97f       Blockzahl Hi und Endezeiger; Ende
dcd9    ea          nop               Programmrest
------------------------------------------------------------------------
dcda                                    Kanal zum Schreiben mit zwei Puffern öffnen und Block
                                        anlegen.
dcda    20  a9  f1  jsr   $f1a9       freien Block in BAM suchen
dcdd    a9  01      lda   #$01        einen
dcdf    20  df  d1  jsr   $d1df       Puffer zum Schreiben suchen
dce2    20  d0  d6  jsr   $d6d0       Parameter an Diskcontroller übergeben
dce5    20  b6  dc  jsr   $dcb6       Kanalparameter setzen
dce8    a6  82      ldx   $82         Kanalnummer
dcea    ad  4a  02  lda   $024a       Filetyp
dced    48          pha               merken
dcee    0a          asl               und mal 2
dcef    05  7f      ora   $7f         mit Drivenummer verknüpfen
dcf1    95  ec      sta   $ec,x       und in Tabelle eintragen
dcf3    68          pla               Filetyp zurückholen
dcf4    c9  04      cmp   #$04        relative Datei?
dcf6    f0  05      beq   $dcfd       verzweige, wenn ja
dcf8    a9  01      lda   #$01        Flag für 'ACTIVE LISTENER' setzen
dcfa    95  f2      sta   $f2,x       und als Kanalstatus übernehmen
dcfc    60          rts               Ende
dcfd    a4  83      ldy   $83         Sekundäradresse
dcff    b9  2b  02  lda   $022b,y     zugehörigen Kanalstatus holen
dd02    29  3f      and   #$3f        und READ/WRITE-Modus setzen
dd04    09  40      ora   #$40        Kanalstatus
dd06    99  2b  02  sta   $022b,y     wieder abspeichern
dd09    ad  58  02  lda   $0258       Recordlänge
dd0c    95  c7      sta   $c7,x       in Tabelle übernehmen
dd0e    20  8e  d2  jsr   $d28e       Puffer für Side-Sektoren suchen
dd11    10  03      bpl   $dd16       verzweige, wenn gefunden
dd13    4c  0f  d2  jmp   $d20f       '70, NO CHANNEL' ausgeben
dd16    a6  82      ldx   $82         Kanalnummer
dd18    95  cd      sta   $cd,x       Puffernummer für Side-Sektor merken
dd1a    20  c1  de  jsr   $dec1       Puffer löschen
dd1d    20  1e  f1  jsr   $f11e       nächsten freien Block in BAM suchen
dd20    a5  80      lda   $80         Tracknummer
dd22    8d  59  02  sta   $0259       für ersten Side.Sektor'Block
dd25    a5  81      lda   $81         Sektornummer
```

```
dd27   8d  5a  02    sta   $025a       für ersten Side-Sektor-Block
dd2a   a6  82        ldx   $82         Kanalnummer
dd2c   b5  cd        lda   $cd,x       zugehörige Puffernummer holen
dd2e   20  d3  d6    jsr   $d6d3       und Parameter an Diskcontroller übergeben
dd31   a9  00        lda   #$00        Pufferzeiger
dd33   20  e9  de    jsr   $dee9       auf Null setzen
dd36   a9  00        lda   #$00        0 als Zeiger für nächsten Track
dd38   20  8d  dd    jsr   $dd8d       in Puffer
dd3b   a9  11        lda   #$11        17 als Anzahl der Bytes
dd3d   20  8d  dd    jsr   $dd8d       in Puffer
dd40   a9  00        lda   #$00        0 als Side-Sektor-Nummer
dd42   20  8d  dd    jsr   $dd8d       in Puffer
dd45   ad  58  02    lda   $0258       Recordlänge
dd48   20  8d  dd    jsr   $dd8d       in Puffer
dd4b   a5  80        lda   $80         aktuelle Tracknummer
dd4d   20  8d  dd    jsr   $dd8d       in Puffer
dd50   a5  81        lda   $81         aktuelle Sektornummer
dd52   20  8d  dd    jsr   $dd8d       in Puffer
dd55   a9  10        lda   #$10        Pufferzeiger auf 16
dd57   20  e9  de    jsr   $dee9       setzen
dd5a   20  3e  de    jsr   $de3e       Track und Sektor holen
dd5d   a5  80        lda   $80         Tracknummer des ersten Datenblocks
dd5f   20  8d  dd    jsr   $dd8d       in Puffer
dd62   a5  81        lda   $81         Sektornummer des ersten Datenblocks
dd64   20  8d  dd    jsr   $dd8d       in Puffer
dd67   20  6c  de    jsr   $de6c       Side-Sektor-Block auf Diskette schreiben
dd6a   20  99  d5    jsr   $d599       Jobausführung abwarten und prüfen
dd6d   a9  02        lda   #$02        Pufferzeiger auf 2
dd6f   20  c8  d4    jsr   $d4c8       setzen
dd72   a6  82        ldx   $82         Kanalnummer
dd74   38            sec               Subtraktion vorbereiten
dd75   a9  00        lda   #$00        Recordlänge durch
dd77   f5  c7        sbc   $c7,x       Subtraktion von 0
dd79   95  c1        sta   $c1,x       als nächste zu bearbeitende Nummer invertieren
dd7b   20  e2  e2    jsr   $e2e2       Records auf Null setzen
dd7e   20  19  de    jsr   $de19       linker des Folgeblocks in Puffer
dd81   20  5e  de    jsr   $de5e       Block auf Diskette schreiben
dd84   20  99  d5    jsr   $d599       Jobausführung abwarten und prüfen
dd87   20  f4  ee    jsr   $eef4       BAM auf die Diskette schreiben
dd8a   4c  98  dc    jmp   $dc98       Diskstatus bereitstellen; Ende
-------------------------------------------------------------------------
dd8d                                   Byte in einen Side-Sektor schreiben.
dd8d   48            pha               Byte merken
dd8e   a6  82        ldx   $82         Kanalnummer
dd90   b5  cd        lda   $cd,x       zugehörige Puffernummer holen
dd92   4c  fd  cf    jmp   $cffd       Byte in Puffer schreiben
-------------------------------------------------------------------------
dd95                                   Setzen bzw. Löschen aller wichtigen Flags. Das Carry-Flag
                                       dient hierbei als Zeiger: C=0 bedeutet Flags löschen C=1
                                       bedeutet Flags setzen.
dd95   90  06        bcc   $dd9d       verzweige, wenn Flags gelöscht werden sollen
dd97   a6  82        ldx   $82         Kanalnummer
dd99   15  ec        ora   $ec,x       Filetyp-Flags setzen
dd9b   d0  06        bne   $dda3       verzweige, wenn ungleich 0
dd9d   a6  82        ldx   $82         Kanalnummer
dd9f   49  ff        eor   #$ff        invertieren
dda1   35  ec        and   $ec,x       und Filetyp-Flags löschen
```

```
dda3   95  ec       sta   $ec,x       Wert abspeichern
dda5   60           rts               Ende
--------------------------------------------------------------------------
dda6                                  Filetyp-Flags testen.
dda6   a6  82       ldx   $82         Kanalnummer
dda8   35  ec       and   $ec,x       mit Flags verknüpfen
ddaa   60           rts               Ende
--------------------------------------------------------------------------
ddab                                  Auf Jobcode für Schreiben prüfen.
ddab   20  93  df   jsr   $df93       Puffernummer holen
ddae   aa           tax               als Index
ddaf   bd  5b  02   lda   $025b,x     Jobcode aus Tabelle holen
ddb2   29  f0       and   #$f0        und isolieren
ddb4   c9  90       cmp   #$90        Schreiben?
ddb6   60           rts               Ende
--------------------------------------------------------------------------
ddb7                                  Testet auf aktives File in Tabelle.
ddb7   a2  00       ldx   #$00        Zähler für Files
ddb9   86  71       stx   $71         setzen
ddbb   bd  2b  02   lda   $022b,x     Kanalstatus holen
ddbe   c9  ff       cmp   #$ff        Kanal belegt?
ddc0   d0  08       bne   $ddca       verzweige, wenn ja
ddc2   a6  71       ldx   $71         Zähler
ddc4   e8           inx               erhöhen
ddc5   e0  10       cpx   #$10        16; höchste Sekundäradresse plus 1?
ddc7   90  f0       bcc   $ddb9       verzweige, wenn kleiner; weitermachen
ddc9   60           rts               Ende; alles geprüft
ddca   86  71       stx   $71         Sekundäradresse merken
ddcc   29  3f       and   #$3f        Kanalnummer isolieren
ddce   a8           tay               und als Index benutzen
ddcf   b9  ec  00   lda   $00ec,y     Kanalfiletyp holen
ddd2   29  01       and   #$01        Drivenummer isolieren
ddd4   85  70       sta   $70         und merken
ddd6   ae  53  02   ldx   $0253       Kanalnummer
ddd9   b5  e2       lda   $e2,x       Standard für Drivenummer (0)
dddb   29  01       and   #$01        Drivenummer bereinigen
dddd   c5  70       cmp   $70         gleich dem Tabellenwert?
dddf   d0  e1       bne   $ddc2       verzweige, wenn nein
dde1   b9  60  02   lda   $0260,y     Sektornummer im Directory
dde4   d5  d8       cmp   $d8,x       gleich Sektor der Datei?
dde6   d0  da       bne   $ddc2       verzweige, wenn nein
dde8   b9  66  02   lda   $0266,y     $0266,y Zeiger auf Directoryeintrag
ddeb   d5  dd       cmp   $dd,x       identisch?
dded   d0  d3       bne   $ddc2       verzweige, wenn nein
ddef   18           clc               Flag für aktives File setzen
ddf0   60           rts               Ende
--------------------------------------------------------------------------
ddf1                                  Block schreiben, falls durch Änderung im Puffer
                                      erforderlich geworden (Puffer ist 'dirty').
ddf1   20  9e  df   jsr   $df9e       Nummer des aktiven Puffers holen
ddf4   50  06       bvc   $ddfc       Ende, wenn Puffer nicht 'dirty'
ddf6   20  5e  de   jsr   $de5e       sonst Block neu auf Diskette schreiben
ddf9   20  99  d5   jsr   $d599       Jobausführung abwarten und prüfen
ddfc   60           rts               Ende
```

```
------------------------------------------------------------------------
ddfd                                    Linker für folgenden Sektor schreiben; Puffer 'dirty'
                                        setzen.
ddfd    20  2b  de    jsr   $de2b       Pufferzeiger 'dirty' setzen
de00    a5  80        lda   $80         Tracknummer
de02    91  94        sta   ($94),y     in Puffer schreiben
de04    c8            iny               Zeiger plus 1 (Y=1)
de05    a5  81        lda   $81         Sektornummer
de07    91  94        sta   ($94),y     in Puffer schreiben
de09    4c  05  e1    jmp   $e105       Puffer 'dirty' setzen; Ende
------------------------------------------------------------------------
de0c                                    Linker auf nächsten Sektor aus dem aktuellen Puffer holen.
de0c    20  2b  de    jsr   $de2b       Pufferzeiger setzen
de0f    b1  94        lda   ($94),y     Linker für Folgetrack
de11    85  80        sta   $80         übernehmen (neue Tracknummer)
de13    c8            iny               Zeiger plus 1 (Y=1)
de14    b1  94        lda   ($94),y     Linker für Folgesektor
de16    85  81        sta   $81         übernehmen (neue Sektornummer)
de18    60            rts               Ende
------------------------------------------------------------------------
de19                                    Linker für letzten Block einer Datei setzen; d.h.
                                        Tracknummer ist $00.
de19    20  2b  de    jsr   $de2b       Pufferzeiger setzen
de1c    a9  00        lda   #$00        Tracknummer gleich 0
de1e    91  94        sta   ($94),y     setzen
de20    c8            iny               Zeiger plus 1 (Y=1)
de21    a6  82        ldx   $82         Kanalnummer
de23    b5  c1        lda   $c1,x       Endekennzeichen holen
de25    aa            tax               und
de26    ca            dex               minus 1 als
de27    8a            txa               Anzahl der Bytes
de28    91  94        sta   ($94),y     setzen
de2a    60            rts               Ende
------------------------------------------------------------------------
de2b                                    Aktuellen Pufferzeiger setzen-
de2b    20  93  df    jsr   $df93       Puffernummer holen
de2e    0a            asl               mal 2
de2f    aa            tax               als Index
de30    b5  9a        lda   $9a,x       Pufferzeiger Hi
de32    85  95        sta   $95         setzen
de34    a9  00        lda   #$00        Pufferzeiger Lo
de36    85  94        sta   $94         setzen (0)
de38    a0  00        ldy   #$00        Index in Puffer ebenfalls auf 0
de3a    60            rts               Ende
------------------------------------------------------------------------
de3b                                    Spur- und Sektornummer aus Jobspeicher nach $80/81 holen.
de3b    20  eb  d0    jsr   $d0eb       Lesekanal suchen; Nummer holen
de3e    20  93  df    jsr   $df93       Puffernummer holen
de41    85  f9        sta   $f9         und übernehmen
de43    0a            asl               mal 2
de44    a8            tay               und als Index
de45    b9  06  00    lda   $0006,y     Tracknummer für Puffer
de48    85  80        sta   $80         übernehmen
de4a    b9  07  00    lda   $0007,y     Sektornummer für Puffer
de4d    85  81        sta   $81         übernehmen
de4f    60            rts               Ende
```

```
------------------------------------------------------------------------
de50                                    Schreib- und Lesejobs ausführen.
de50   a9  90       lda   #$90         Jobcode für Schreiben eines Blocks
de52   8d  4d  02    sta   $024d        merken
de55   d0  28       bne   $de7f        unbedingter Sprung
de57   a9  80       lda   #$80         Jobcode für Lesen eines Blocks
de59   8d  4d  02    sta   $024d        merken
de5c   d0  21       bne   $de7f        unbedingter Sprung
de5e   a9  90       lda   #$90         Jobcode für Schreiben eines Blocks
de60   8d  4d  02    sta   $024d        merken
de63   d0  26       bne   $de8b        unbedingter Sprung
de65   a9  80       lda   #$80         Jobcode für Lesen eines Blocks
de67   8d  4d  02    sta   $024d        merken
de6a   d0  1f       bne   $de8b        unbedingter Sprung
de6c   a9  90       lda   #$90         Jobcode für Schreiben eines Blocks
de6e   8d  4d  02    sta   $024d        merken
de71   d0  02       bne   $de75        unbedingter Sprung
de73   a9  80       lda   #$80         Jobcode für Lesen eines Blocks
de75   8d  4d  02    sta   $024d        merken
de78   a6  82       ldx   $82          Kanalnummer holen
de7a   b5  cd       lda   $cd,x        Puffernummer für Side-Sektor-Block
de7c   aa           tax                prüfen
de7d   10  13       bpl   $de92        verzweige, wenn Puffer aktiv
de7f   20  d0  d6    jsr   $d6d0        Parameter an Diskcontroller übergeben
de82   20  93  df    jsr   $df93        Puffernummer holen
de85   aa           tax                Puffernummer als Index
de86   a5  7f       lda   $7f          aktuelle Drivenummer
de88   9d  5b  02    sta   $025b,x      in Befehlstabelle
de8b   20  15  e1    jsr   $e115        Puffer 'dirty' Flag löschen
de8e   20  93  df    jsr   $df93        Puffernummer holen
de91   aa           tax                Puffernummer aLs Index
de92   4c  06  d5    jmp   $d506        Befehlscode prüfen und an Diskcontroller; Ende
------------------------------------------------------------------------
de95                                    Nächsten Sektor der Datei anhand des Linkers im aktuellen
                                        Puffer feststellen.
de95   a9  00       lda   #$00         Pufferzeiger auf 0
de97   20  c8  d4    jsr   $d4c8        setzen
de9a   20  37  d1    jsr   $d137        Byte aus Puffer holen (Tracknummer)
de9d   85  80       sta   $80          und übernehmen
de9f   20  37  d1    jsr   $d137        Byte aus Puffer holen (Sektornummer)
dea2   85  81       sta   $81          und ebenfalls übernehmen
dea4   60           rts                Ende
------------------------------------------------------------------------
dea5                                    Pufferinhalte aus einem Puffer in einen anderen übertragen.
                                        A enthält die Anzahl der zu übertragenden Bytes; Y die
                                        Nummer des Puffers, von dem die Daten kommen und X die
                                        Nummer des Puffers, der die Daten aufnehmen soll.
dea5   48           pha                Anzahl der Bytes merken
dea6   a9  00       lda   #$00         Pufferadressen Lo
dea8   85  6f       sta   $6f          setzen
deaa   85  71       sta   $71          setzen
deac   b9  e0  fe    lda   $fee0,y      Adresse des Quellpuffers Hi holen
deaf   85  70       sta   $70          und setzen
deb1   bd  e0  fe    lda   $fee0,x      Adresse des Zielpuffers Hi holen
deb4   85  72       sta   $72          und setzen
deb6   68           pla                Anzahl der Bytes zurückholen
deb7   a8           tay                und als Zähler
```

```
deb8   88           dey              benutzen
deb9   b1  6f       lda   ($6f),y    Byte aus Quellpuffer
debb   91  71       sta   ($71),y    in Zielpuffer übertragen
debd   88           dey              Zähler minus 1
debe   10  f9       bpl   $deb9      und nächstes Byte holen
dec0   60           rts              Ende
--------------------------------------------------------------------------
dec1                                 Puffer löschen, dessen Nummer sich in A befindet
dec1   a8           tay              Puffernummer nach Y
dec2   b9  e0  fe   lda   $fee0,y    Pufferadresse Hi holen
dec5   85  70       sta   $70        und setzen
dec7   a9  00       lda   #$00       Pufferadresse Lo
dec9   85  6f       sta   $6f        gleich 0 setzen
decb   a8           tay              0 nach Y
decc   91  6f       sta   ($6f),y    und in Puffer
dece   c8           iny              nächstes Byte
decf   d0  fb       bne   $decc      und weitermachen
ded1   60           rts              Ende
--------------------------------------------------------------------------
ded2                                 Side-Sektor-Nummer holen.
ded2   a9  00       lda   #$00       Pufferzeiger auf Null
ded4   20  dc  de   jsr   $dedc      setzen
ded7   a0  02       ldy   #$02       Byte 2
ded9   b1  94       lda   ($94),y    Side-Sektor-Nummer nach A
dedb   60           rts              Ende
--------------------------------------------------------------------------
dedc                                 Pufferzeiger auf Side-Sektor setzen, wobei A das Lo-Byte
                                     enthalten muß.
dedc   85  94       sta   $94        Pufferzeiger Lo setzen
dede   a6  82       ldx   $82        Kanalnummer
dee0   b5  cd       lda   $cd,x      zugehörige Puffernummer
dee2   aa           tax              als Index
dee3   bd  e0  fe   lda   $fee0,x    Pufferadresse Hi holen
dee6   85  95       sta   $95        und setzen
dee8   60           rts              Ende
--------------------------------------------------------------------------
dee9                                 Pufferzeiger setzen.
dee9   48           pha              Pufferzeiger Lo merken
deea   20  dc  de   jsr   $dedc      Pufferzeiger setzen
deed   48           pha              Pufferzeiger Hi merken
deee   8a           txa              Puffernummer
deef   0a           asl              mal 2
def0   aa           tax              als Index
def1   68           pla              Pufferzeiger Hi zurückholen
def2   95  9a       sta   $9a,x      und setzen
def4   68           pla              Pufferzeiger Lo zurückholen
def5   95  99       sta   $99,x      und setzen
def7   60           rts              Ende
--------------------------------------------------------------------------
def8                                 Side-Sektor und Pufferzeiger setzen. V-Flag = 0: alles in
                                     Ordnung V-Flag = 1: kein Side-Sektor!
def8   20  66  df   jsr   $df66      Side-Sektor in Puffer und ok?
defb   30  0e       bmi   $df0b      verzweige, wenn nein
defd   50  13       bvc   $df12      verzweige, wenn alles in Ordnung
deff   a6  82       ldx   $82        Kanalnummer
df01   b5  cd       lda   $cd,x      zugehörige Puffernummer
```

```
df03   20  1b  df    jsr   $df1b       Side-Sektor in Puffer lesen
df06   20  66  df    jsr   $df66       nochmals prüfen, ob Side-Sektor ok
df09   10  07        bpl   $df12       verzweige, wenn ja
df0b   20  cb  e1    jsr   $e1cb       vorherigen Side-Sektor lesen
df0e   2c  ce  fe    bit   $fece       V-Flag setzen
df11   60            rts               Ende; Fehlerstatus
df12   a5  d6        lda   $d6         Zeiger in Side-Sektor
df14   20  e9  de    jsr   $dee9       Pufferzeiger in Side-Sektor setze
df17   2c  cd  fe    bit   $fecd       V-Flag löschen
df1a   60            rts               Ende; Status ok
--------------------------------------------------------------------------
df1b                                   Routine zur Übergabe von Jobs an den Diskcontroller. Bei
                                       Einsprung in $df1b wird gelesen; bei Einsprung über $df21
                                       wird geschrieben. A muß die Nummer des Puffers zum
                                       Schreiben/Lesen und X die Nummer des aktiven Puffers
                                       enthalten.
df1b   85  f9        sta   $f9         Puffernummer merken
df1d   a9  80        lda   #$80        Jobcode für lesen eines Blocks
df1f   d0  04        bne   $df25       unbedingter Sprung zur übergabe

df21   85  f9        sta   $f9         Puffernummer merken
df23   a9  90        lda   #$90        Jobcode für Schreiben eines Blocks
df25   48            pha               merken
df26   b5  ec        lda   $ec,x       Kanalfiletyp holen
df28   29  01        and   #$01        Drivenummer isolieren
df2a   85  7f        sta   $7f         und übernehmen
df2c   68            pla               Jobcode zurückholen
df2d   05  7f        ora   $7f         mit Drivenummer verknüpfen
df2f   8d  4d  02    sta   $024d       und merken
df32   b1  94        lda   ($94),y     Folgetrack aus Puffer holen
df34   85  80        sta   $80         und übernehmen
df36   c8            iny               Zeiger plus 1
df37   b1  94        lda   ($94),y     Folgesektor aus Puffer holen
df39   85  81        sta   $81         und ebenfalls übernehmen
df3b   a5  f9        lda   $f9         aktuelle Puffernummer
df3d   20  d3  d6    jsr   $d6d3       Parameter an Diskcontroller übergeben
df40   a6  f9        ldx   $f9         Puffernummer
df42   4c  93  d5    jmp   $d593       Jobcode an Diskcontroller übergeben; Ende
--------------------------------------------------------------------------
df45                                   Werte für Side-Sektor setzen.
df45   a6  82        ldx   $82         Kanalnummer
df47   b5  cd        lda   $cd,x       zugehörige Puffernummer
df49   4c  eb  d4    jmp   $d4eb       Pufferzeiger setzen; Ende
--------------------------------------------------------------------------
df4c                                   Gesamtzahl der Blöcke einer relativen Datei berechnen.
df4c   a9  78        lda   #$78        120 (Anzahl der Blockzeiger pro Side-Sektor)
df4e   20  5c  df    jsr   $df5c       zu $70/71 addieren
df51   ca            dex               Side-Sektor-Nummer
df52   10  f8        bpl   $df4c       nächster Side-Sektor
df54   a5  72        lda   $72         Anzahl der linkbytes
df56   4a            lsr               geteilt durch 2 (da 2 Bytes pro linker)
df57   20  5c  df    jsr   $df5c       Anzahl wieder addieren
df5a   a5  73        lda   $73         Zahl der belegten Side-Sektoren
df5c   18            clc               Addition vorbereiten
df5d   65  70        adc   $70         addieren und
df5f   85  70        sta   $70         Ergebnis abspeichern
```

```
df61   90  02       bcc   $df65       verzweige, wenn kein Übertrag
df63   e6  71       inc   $71         sonst Anzahl Hi erhöhen
df65   60           rts               Ende
-------------------------------------------------------------------------
df66                                  Zustand eines Side-Sektor-Blocks im Puffer prüfen.
df66   20  d2  de    jsr   $ded2       Nummer des Side-Sektors holen
df69   c5  d5       cmp   $d5         identisch mit Side-Sektor im Puffer?
df6b   d0  0e       bne   $df7b       verzweige, wenn nein
df6d   a4  d6       ldy   $d6         Zeiger in Side-Sektor
df6f   b1  94       lda   ($94),y     Tracknummer aus Puffer holen
df71   f0  04       beq   $df77       verzweige, wenn kein weiterer Sektor
df73   2c  cd  fe    bit   $fecd       Fehlerflags löschen
df76   60           rts               Ende; alles ok
df77   2c  cf  fe    bit   $fecf       N-Flag setzen
df7a   60           rts               Ende; kein Side-Sektor
df7b   a5  d5       lda   $d5         Side-Sektor-Nummer
df7d   c9  06       cmp   #$06        größer gleich 6?
df7f   b0  0a       bcs   $df8b       verzweige, wenn ja
df81   0a           asl               mal 2
df82   a8           tay               als Zeiger in Puffer
df83   a9  04       lda   #$04        Pufferzeiger Lo
df85   85  94       sta   $94         auf 4 setzen
df87   b1  94       lda   ($94),y     Tracknummer holen
df89   d0  04       bne   $df8f       verzeige, wenn vorhanden
df8b   2c  d0  fe    bit   $fed0       sonst N- und V-Flag setzen
df8e   60           rts               Ende; Fehlerstatus
df8f   2c  ce  fe    bit   $fece       V-Flag setzen
df92   60           rts               Ende; Side-Sektor nicht vorhanden
-------------------------------------------------------------------------
df93                                  Nummer des aktiven Puffers holen.
df93   a6  82       ldx   $82         Kanalnummer
df95   b5  a7       lda   $a7,x       zugehörige Puffernummer aus Tabelle
df97   10  02       bpl   $df9b       verzweige, wenn Puffer belegt
df99   b5  ae       lda   $ae,x       Puffer aktiv oder nur belegt?
df9b   29  bf       and   #$bf        testen...
df9d   60           rts               Ende; bei Rückkehr Status abfragen
-------------------------------------------------------------------------
df9e                                  Aktiven Puffer prüfen.
df9e   a6  82       ldx   $82         Kanalnummer
dfa0   8e  57  02    stx   $0257       merken
dfa3   b5  a7       lda   $a7,x       Puffernummer holen
dfa5   10  09       bpl   $dfb0       verzweige, wenn Puffer belegt
dfa7   8a           txa               Puffernummer
dfa8   18           clc               7
dfa9   69  07       adc   #$07        addieren
dfab   8d  57  02    sta   $0257       und merken
dfae   b5  ae       lda   $ae,x       Puffer aktiv oder nur belegt?
dfb0   85  70       sta   $70         Status merken
dfb2   29  1f       and   #$1f        Puffer auf Aktivität
dfb4   24  70       bit   $70         prüfen
dfb6   60           rts               Ende; nach Rückkehr Status abfragen
-------------------------------------------------------------------------
dfb7                                  Nummer eines inaktiven Puffers und eines Kanals holen.
dfb7   a6  82       ldx   $82         Kanalnummer
dfb9   b5  a7       lda   $a7,x       zugehörige Puffernummer
dfbb   30  02       bmi   $dfbf       verzweige, wenn Puffer frei
```

```
dfbd   b5  ae       lda   $ae,x       belegter Puffer aktiv?
dfbf   c9  ff       cmp   #$ff        cmp #$ff prüfen...
dfc1   60           rts               Ende; nach Rückkehr Status abfragen
-------------------------------------------------------------------------
dfc2                                  Puffer freigeben oder inaktivieren; Puffernummer in A.
dfc2   a6  82       ldx   $82         Kanalnummr
dfc4   09  80       ora   #$80        Flag für Puffer inaktiv setzen
dfc6   b4  a7       ldy   $a7,x       Puffer belegt?
dfc8   10  03       bpl   $dfcd       verzweige, wenn ja
dfca   95  a7       sta   $a7,x       Puffer inaktiv setzen
dfcc   60           rts               Ende
dfcd   95  ae       sta   $ae,x       Puffer unbelegt setzen
dfcf   60           rts               Ende
-------------------------------------------------------------------------
dfd0                                  Nächsten Record einer relativen Datei erstellen.
dfd0   a9  20       lda   #$20        Bit 5
dfd2   20  9d  dd   jsr   $dd9d       Kanalfiletyp setzen
dfd5   a9  80       lda   #$80        Bit 7
dfd7   20  a6  dd   jsr   $dda6       bei Kanalfiletyp gesetzt?
dfda   d0  41       bne   $e01d       verzweige, wenn ja
dfdc   a6  82       ldx   $82         KanalnUlllll1er
dfde   f6  b5       inc   $b5,x       Recordnummer Lo plus 1
dfe0   d0  02       bne   $dfe4       verzweige, wenn ungleich 0
dfe2   f6  bb       inc   $bb,x       RecordnUlllll1er Hi plus 1
dfe4   a6  82       ldx   $82         Kanalnummer
dfe6   b5  c1       lda   $c1,x       Zeiger auf nächsten Record
dfe8   f0  2e       beq   $e018       verzweige, wenn Null
dfea   20  e8  d4   jsr   $d4e8       Pufferzeiger entsprechend A setzen
dfed   a6  82       ldx   $82         Kanalnummer
dfef   d5  c1       cmp   $c1,x       Pufferzeiger kleiner Recordzeiger?
dff1   90  03       bcc   $dff6       verzweige, wenn ja
dff3   20  3c  e0   jsr   $e03c       Record schreiben; nächsten lesen
dff6   a6  82       ldx   $82         KanalnUlllll1er
dff8   b5  c1       lda   $c1,x       Zeiger auf nächsten Record
dffa   20  c8  d4   jsr   $d4c8       Pufferzeiger entsprechend A setzen
dffd   a1  99       lda   ($99,x)     Byte aus Puffer holen
dfff   85  85       sta   $85         und für Ausgabe bereitstellen
e001   a9  20       lda   #$20        Bit 5
e003   20  9d  dd   jsr   $dd9d       Kanalfiletyp wiederherstellen
e006   20  04  e3   jsr   $e304       Recordlänge zu Zeiger addieren
e009   48           pha               Ergebnis merken
e00a   90  28       bcc   $e034       verzweige, wenn Record gefunden
e00c   a9  00       lda   #$00        Pufferzeiger 0 auf Folgetrack
e00e   20  f6  d4   jsr   $d4f6       Linkbyte des Folgetracks aus Puffer holen
e011   d0  21       bne   $e034       verzweige, wenn Block vorhanden
e013   68           pla               Zeiger zurückholen
e014   c9  02       cmp   #$02        Ergebnis gleich 2?
e016   f0  12       beq   $e02a       verzweige, wenn ja
e018   a9  80       lda   #$80        Bit 7
e01a   20  97  dd   jsr   $dd97       testen
e01d   20  2f  d1   jsr   $d12f       Kanal, und Puffernummer holen
e020   b5  99       lda   $99,x       Pufferzeiger Lo
e022   99  44  02   sta   $0244,y     als Endezeiger merken
e025   a9  0d       lda   #$0d        'RETURN'
e027   85  85       sta   $85         für Ausgabe bereitstellen
e029   60           rts               Ende
e02a   20  35  e0   jsr   $e035       Zeiger auf letztes Zeichen setzen
```

```
e02d   a6  82       ldx   $82         Kanalnummer
e02f   a9  00       lda   #$00        Zeiger auf nächsten Record gleich 0
e031   95  c1       sta   $c1,x       setzen
e033   60           rts               Ende
e034   68           pla               Zeiger zurückholen
e035   a6  82       ldx   $82         Kanalnummer
e037   95  c1       sta   $c1,x       als Zeiger auf nächsten Record speichern
e039   4c  6e  e1    jmp   $e16e       Zeiger auf letztes Zeichen; Ende
--------------------------------------------------------------------------------
e03c                                  Nächsten Record in Puffer generierieren; vorherigen Block
                                      schreiben.
e03c   20  d3  d1    jsr   $d1d3       Drivenummer setzen
e03f   20  95  de    jsr   $de95       Track und Sektor setzen
e042   20  9e  df    jsr   $df9e       Puffer 'dirty' (Inhalt geändert)?
e045   50  16       bvc   $e05d       verzweige, wenn nein
e047   20  5e  de    jsr   $de5e       Pufferinhalt auf Diskette schreiben
e04a   20  1e  cf    jsr   $cf1e       Puffer wechseln
e04d   a9  02       lda   #$02        Pufferzeiger auf 2
e04f   20  c8  d4    jsr   $d4c8       setzen
e052   20  ab  dd    jsr   $ddab       war letzter Job ein Schreibjob?
e055   d0  24       bne   $e07b       verzweige, wenn nein
e057   20  57  de    jsr   $de57       Block von Diskette lesen
e05a   4c  99  d5    jmp   $d599       Jobausführung abwarten und prüfen
e05d   20  1e  cf    jsr   $cf1e       Puffer wechseln
e060   20  ab  dd    jsr   $ddab       war letzter Job ein Schreibjob?
e063   d0  06       bne   $e06b       verzweige, wenn nein
e065   20  57  de    jsr   $de57       Block von Diskette lesen
e068   20  99  d5    jsr   $d599       Jobausführung abwarten und prüfen
e06b   20  95  de    jsr   $de95       Track und Sektor setzen
e06e   a5  80       lda   $80         Tracknummer; Folgeblock vorhanden?
e070   f0  09       beq   $e07b       verzweige, wenn nein
e072   20  1e  cf    jsr   $cf1e       Puffer wechseln
e075   20  57  de    jsr   $de57       Block von Diskette lesen
e078   20  1e  cf    jsr   $cf1e       Puffer wechseln
e07b   60           rts               Ende
--------------------------------------------------------------------------------
e07c                                  Ein Byte in den Recordpuffer schreiben.
e07c   20  05  e1    jsr   $e105       Puffer 'dirty' setzen, da Inhalt geändert
e07f   20  93  df    jsr   $df93       Puffernummer holen
e082   0a           asl               mal 2
e083   aa           tax               als Index
e084   a5  85       lda   $85         Byte von seriellem Bus
e086   81  99       sta   ($99,x)     in den Puffer schreiben
e088   b4  99       ldy   $99,x       Pufferzeiger
e08a   c8           iny               plus 1
e08b   d0  09       bne   $e096       verzweige, wenn noch nicht Null
e08d   a4  82       ldy   $82         Kanalnummer
e08f   b9  c1  00    lda   $00c1,y     Zeiger auf nächsten Record
e092   f0  0a       beq   $e09e       verzweige, wenn gleich Null
e094   a0  02       ldy   #$02        Puferzeiger neu setzen
e096   98           tya               und nach A
e097   a4  82       ldy   $82         Kanalnummer
e099   d9  c1  00    cmp   $00c1,y     gleich Zeiger auf Record?
e09c   d0  05       bne   $e0a3       verzweige, wenn nein
e09e   a9  20       lda   #$20        Bit 5
e0a0   4c  97  dd    jmp   $dd97       Kanalfiletyp ändern; Ende
e0a3   f6  99       inc   $99,x       Pufferzeiger erhöhen
e0a5   d0  03       bne   $e0aa       verzweige, wenn noch nicht Null
```

```
e0a7   20  3c  e0    jsr   $e03c       Record schreiben; nächsten lesen
e0aa   60            rts               Ende
--------------------------------------------------------------------------
e0ab                                   Schreiben der Records.
e0ab   a9  a0        lda   #$a0        Bit 7 und 5
e0ad   20  a6  dd    jsr   $dda6       $dda6 testen
e0b0   d0  27        bne   $e0d9       verzweige, wenn eines gesetzt
e0b2   a5  85        lda   $85         Byte von seriellem Bus
e0b4   20  7c  e0    jsr   $e07c       in Recordpuffer schreiben
e0b7   a5  f8        lda   $f8         EOI erhalten?
e0b9   f0  0d        beq   $e0c8       verzweige, wenn nein
e0bb   60            rts               Ende
e0bc   a9  20        lda   #$20        Bit 5
e0be   20  a6  dd    jsr   $dda6       testen
e0c1   f0  05        beq   $e0c8       verzweige, wenn nicht gesetzt
e0c3   a9  51        lda   #$51        Nummer der Fehlermeldung
e0c5   8d  6c  02    sta   $026c       Fehlerflag setzen
e0c8   20  f3  e0    jsr   $e0f3       Rest des Records mit Null füllen
e0cb   20  53  e1    jsr   $e153       nächsten Record suchen
e0ce   ad  6c  02    lda   $026c       Fehler aufgetreten?
e0d1   f0  03        beq   $e0d6       verzweige, wenn nein
e0d3   4c  c8  c1    jmp   $c1c8       '51, OVERFLOY IN RECORD' ausgeben
e0d6   4c  bc  e6    jmp   $e6bc       Ende; alles ok
e0d9   29  80        and   #$80        Bit 7 testen
e0db   d0  05        bne   $e0e2       verzweige, wenn gesetzt
e0dd   a5  f8        lda   $f8         EOI erhalten?
e0df   f0  db        beq   $e0bc       verzweige, wenn nein
e0e1   60            rts               Ende
e0e2   a5  85        lda   $85         Byte von seriellem Bus
e0e4   48            pha               merken
e0e5   20  1c  e3    jsr   $e31c       mehr 'Platz in der Datei schaffen
e0e8   68            pla               Datenbyte zurückholen
e0e9   85  85        sta   $85         und wieder abspeichern
e0eb   a9  80        lda   #$80        Bit 7
e0ed   20  9d  dd    jsr   $dd9d       Kanalfiletyp wiederherstellen
e0f0   4c  b2  e0    jmp   $e0b2       Byte in Datei schreiben; Ende
--------------------------------------------------------------------------
e0f3                                   Füllt den Rest des Records mit Nullen auf.
e0f3   a9  20        lda   #$20        Bit 5
e0f5   20  a6  dd    jsr   $dda6       testen
e0f8   d0  0a        bne   $e104       verzweige, wenn gesetzt
e0fa   a9  00        lda   #$00        Datenbyte gleich $00
e0fc   85  85        sta   $85         setzen
e0fe   20  7c  e0    jsr   $e07c       Byte in Recordpuffer schreiben
e101   4c  f3  e0    jmp   $e0f3       weitermachen, bis Record voll
e104   60            rts               Ende
--------------------------------------------------------------------------
e105                                   Puffernummer in Tabelle registrieren und anzei. gen, daß
                                       der Pufferinhalt aktualisiert wurde, so daß er mit dem
                                       Block auf der Diskette nicht mehr übereinstimmt ('dirty
                                       butter').
e105   a9  40        lda   #$40        Bit 6
e107   20  97  dd    jsr   $dd97       als Flag setzen
e10a   20  9e  df    jsr   $df9e       Puffernummer holen; Flags setzen
e10d   09  40        ora   #$40        Bit 6 setzen
e10f   ae  57  02    ldx   $0257       Kanalnummer plus 7
e112   95  a7        sta   $a7,x       Puffer 'dirty' Flag in Tabelle
e114   60            rts               Ende
```

```
------------------------------------------------------------------------
e115                                    Löschen des Flags in der Puffertabelle, das die Änderungen
                                        an einem Puffer anzeigt ('dirty' Flag).
e115   20  9e  df    jsr   $df9e       Puffernummer holen; Flags setzen
e118   29  bf        and   #$bf        Bit 6 löschen
e11a   ae  57  02    ldx   $0257       Kanalnummer plus 7
e11d   95  a7        sta   $a7,x       Flags in Tabelle eintragen
e11f   60            rts               Ende
------------------------------------------------------------------------
e120                                    Byte aus Recordpuffer holen.
e120   a9  80        lda   #$80        Bit 7
e122   20  a6  dd    jsr   $dda6       testen
e125   d0  37        bne   $e15e       verzweige, wenn gesetzt
e127   20  2f  d1    jsr   $d12f       Byte aus Puffer holen
e12a   b5  99        lda   $99,x       Pufferzeiger
e12c   d9  44  02    cmp   $0244,y     Ende schon erreicht?
e12f   f0  22        beq   $e153       verzweige, wenn ja
e131   f6  99        inc   $99,x       Pufferzeiger erhöhen
e133   d0  06        bne   $e13b       verzweige, wenn ungleich Null
e135   20  3c  e0    jsr   $e03c       Record schreiben; nächste Record lesen
e138   20  2f  d1    jsr   $d12f       Zeiger in Puffer setzen
e13b   a1  99        lda   ($99,x)     Byte aus Puffer holen
e13d   99  3e  02    sta   $023e,y     und für Ausgabe bereitstellen
e140   a9  89        lda   #$89        READ/WRITE Flag setzen und
e142   99  f2  00    sta   $00f2,y     Kanalstatus herstellen
e145   b5  99        lda   $99,x       Pufferzeiger
e147   d9  44  02    cmp   $0244,y     Ende schon erreicht?
e14a   f0  01        beq   $e14d       verzweige, wenn ja
e14c   60            rts               Ende
e14d   a9  81        lda   #$81        Kanalstatus zurücksetzen, da
e14f   99  f2  00    sta   $00f2,y     Ende des Blocks erreicht
e152   60            rts               Ende
------------------------------------------------------------------------
e153                                    Nächsten Record lesen und Byte für Ausgabe bereitstellen.
e153   20  d0  df    jsr   $dfd0       nächsten Record suchen
e156   20  2f  d1    jsr   $d12f       Zeiger in Puffer setzen
e159   a5  85        lda   $85         Datenbyte aus Puffer
e15b   4c  3d  e1    jmp   $e13d       für Ausgabe bereitstellen; Ende
------------------------------------------------------------------------
e15e                                    Abbruch bei Fehler.
e15e   a6  82        ldx   $82         Kanalnummer
e160   a9  0d        lda   #$0d        'RETURN'
e162   9d  3e  02    sta   $023e,x     für Ausgabe auf Bus bereitstellen
e165   a9  81        lda   #$81        Kanalstatus zurücksetzen, da
e167   95  f2        sta   $f2,x       Arbeit abgebrochen worden ist
e169   a9  50        lda   #$50        Nummer der Fehlermeldung
e16b   20  c8  c1    jsr   $c1c8       '50, RECORD NOT PRESENT' ausgeben
------------------------------------------------------------------------
e16e                                    Letztes Zeichen im Record durch Zeiger markieren.
e16e   a6  82        ldx   $82         Kanalnummer
e170   b5  c1        lda   $c1,x       Zeiger auf nächsten Record
e172   85  87        sta   $87         merken
e174   c6  87        dec   $87         minus 1
e176   c9  02        cmp   #$02        Vergleich mit 2
e178   d0  04        bne   $e17e       verzweige, wenn ungleich
```

```
e17a   a9  ff       lda   #$ff        Zeiger auf $ff
e17c   85  87       sta   $87         setzen
e17e   b5  c7       lda   $c7,x       Recordlänge
e180   85  88       sta   $88         merken
e182   20  e8  d4   jsr   $d4e8       Pufferzeiger entsprechend setze-
e185   a6  82       ldx   $82         Kanalnummer
e187   c5  87       cmp   $87         Pufferzeiger größer als Recordze';e-?
e189   90  19       bcc   $e1a4       verzweige, wenn nein
e18b   f0  17       beq   $e1a4       verzweige, wenn beide gleich
e18d   20  1e  cf   jsr   $cf1e       Puffer wechseln
e190   20  b2  e1   jsr   $e1b2       Byte an Zeigerposition aus Puffer ho,en
e193   90  08       bcc   $e19d       verzweige bei Zeigerunterlauf
e195   a6  82       ldx   $82         Kanalnummer
e197   9d  44  02   sta   $0244,x     Endezeiger setzen
e19a   4c  1e  cf   jmp   $cf1e       Puffer wechseln; Ende
e19d   20  1e  cf   jsr   $cf1e       Puffer wechseln
e1a0   a9  ff       lda   #$ff        Recordzeiger auf Sff
e1a2   85  87       sta   $87         setzen
e1a4   20  b2  e1   jsr   $e1b2       Byte an Zeigerposition aus Puffer holen
e1a7   b0  03       bcs   $e1ac       verzweige, wenn nicht letztes Byte
e1a9   20  e8  d4   jsr   $d4e8       Pufferzeiger neu setzen
e1ac   a6  82       ldx   $82         Kanalnummer
e1ae   9d  44  02   sta   $0244,x     Endezeiger setzen
e1b1   60           rts               Ende
--------------------------------------------------------------------------
e1b2                                  Letztes Byte ungleich 0 eines Records finden.
e1b2   20  2b  de   jsr   $de2b       Pufferzeiger auf Null setzen
e1b5   a4  87       ldy   $87         Recordzeiger
e1b7   b1  94       lda   ($94),y     Byte aus Puffer holen
e1b9   d0  0d       bne   $e1c8       verzweige, wenn ungleich Null
e1bb   88           dey               nächstes Zeichen
e1bc   c0  02       cpy   #$02        schon letztes Zeichen geholt?
e1be   90  04       bcc   $e1c4       verzweige, wenn ja
e1c0   c6  88       dec   $88         Recordlänge minus 1
e1c2   d0  f3       bne   $e1b7       verzweige, wenn noch nicht Null
e1c4   c6  88       dec   $88         Recordlänge minus 1
e1c6   18           clc               Flag für ok; Ende gefunden
e1c7   60           rts               Ende
e1c8   98           tya               Recordzeiger
e1c9   38           sec               Flag für Ende noch nicht gefunden
e1ca   60           rts               Ende
--------------------------------------------------------------------------
e1cb                                  Ende des letzten Records feststellen und Zei- ger
                                      entsprechend setzen.
e1cb   20  d2  de   jsr   $ded2       Side.Sektor.Nummer holen
e1ce   85  d5       sta   $d5         und merken
e1d0   a9  04       lda   #$04        Zeiger in Puffer Lo
e1d2   85  94       sta   $94         auf 4 setzen
e1d4   a0  0a       ldy   #$0a        Defaultwert für Pufferindex 10
e1d6   d0  04       bne   $e1dc       unbedingter Sprung
e1d8   88           dey               Pufferindex minus 1
e1d9   88           dey               minus 1
e1da   30  26       bmi   $e202       verzweige, wenn Y kleiner 0; Fehler
e1dc   b1  94       lda   ($94),y     Tracknummer des vorherigen Blocks
e1de   f0  f8       beq   $e1d8       verzweige, wenn keiner vorhanden
e1e0   98           tya               Zeiger
e1e1   4a           lsr               getei l t durch 2
e1e2   c5  d5       cmp   $d5         gleich Nummer des aktuellen Blocks?
```

```
e1e4   f0  09       beq   $e1ef       verzweige, wenn ja
e1e6   85  d5       sta   $d5         Nummer neu merken
e1e8   a6  82       ldx   $82         Kanalnummer
e1ea   b5  cd       lda   $cd,x       Puffer für Side-Sektor
e1ec   20  1b  df   jsr   $df1b       Side-Sektor lesen
e1ef   a0  00       ldy   #$00        Pufferzeiger gleich 0
e1f1   84  94       sty   $94         setzen
e1f3   b1  94       lda   ($94),y     Tracknummer des Folgeblocks
e1f5   d0  0b       bne   $e202       verzweige, wenn Block folgt
e1f7   c8           iny               sonst Zeiger plus 1
e1f8   b1  94       lda   ($94),y     und Anzahl der Bytes des Blocks holen
e1fa   a8           tay               als Zeiger nehmen
e1fb   88           dey               minus 1 gibt Index
e1fc   84  d6       sty   $d6         und als Vergleichswert abspeichern
e1fe   98           tya               Anzahl wieder zurückholen
e1ff   4c  e9  de   jmp   $dee9       und Pufferzeiger auf letztes Byte setzen
e202   a9  67       lda   #$67        Nummer der Fehlermeldung
e204   20  45  e6   jsr   $e645       '67, ILLEGAL TRACK OR SEKTOR' ausgeben
--------------------------------------------------------------------------
e207                                  POSITION-Befehl (P-Befehl),
e207   20  b3  c2   jsr   $c2b3       Befehlsstring prufen
e20a   ad  01  02   lda   $0201       zweites Zeichen holen
e20d   85  83       sta   $83         und als Sekundäradresse setzen
e20f   20  eb  d0   jsr   $d0eb       Lesekanal suchen
e212   90  05       bcc   $e219       verzweige, wenn Kanal gefunden
e214   a9  70       lda   #$70        Nummer der Fehlermeldung
e216   20  c8  c1   jsr   $c1c8       '70, NO CHANNEL' ausgeben
e219   a9  a0       lda   #$a0        Bit 7 und 5
e21b   20  9d  dd   jsr   $dd9d       Kanalfiletyp setzen
e21e   20  25  d1   jsr   $d125       Filetyp prüfen
e221   f0  05       beq   $e228       verzweige, wenn relatives File
e223   a9  64       lda   #$64        Nummer der Fehlermeldung
e225   20  c8  c1   jsr   $c1c8       '64, FILE TYPE MISMATCH' ausgeben
e228   b5  ec       lda   $ec,x       Kanalfiletyp
e22a   29  01       and   #$01        Drivenummer isolieren
e22c   85  7f       sta   $7f         und übernehmen
e22e   ad  02  02   lda   $0202       drittes Zeichen aus Befehlsstring holen
e231   95  b5       sta   $b5,x       als Recordnummer Lo merken
e233   ad  03  02   lda   $0203       viertes Zeichen aus Befehlsstring holen
e236   95  bb       sta   $bb,x       als Recordnummer Hi merken
e238   a6  82       ldx   $82         Kanalnummer
e23a   a9  89       lda   #$89        READ/WRITE-Status
e23c   95  f2       sta   $f2,x       als Kanalstatus übernehmen
e23e   ad  04  02   lda   $0204       fünftes Zeichen aus Befehlsstring holen
e241   f0  10       beq   $e253       verzweige, wenn Null
e243   38           sec               teste, ob
e244   e9  01       sbc   #$01        Wert gleich 1?
e246   f0  0b       beq   $e253       verzweige, wenn ert gleich 1
e248   d5  c7       cmp   $c7,x       mit Recordlänge vergleichen
e24a   90  07       bcc   $e253       verzweige, wenn kleiner
e24c   a9  51       lda   #$51        Nummer der Fehlermeldung
e24e   8d  6c  02   sta   $026c       als Fehlerflag merken
e251   a9  00       lda   #$00        Zeiger auf Beginn des Records
e253   85  d4       sta   $d4         setzen
e255   20  0e  ce   jsr   $ce0e       Zeiger in relative Datei berechnen
e258   20  f8  de   jsr   $def8       entsprechenden Side-Sektor lesen
e25b   50  08       bvc   $e265       verzweige, wenn Side-Sektor ok
e25d   a9  80       lda   #$80        Bit 7
```

```
e25f   20  97  dd    jsr   $dd97       Kanalfiletyp wiederherstellen
e262   4c  5e  e1    jmp   $e15e       '50, RECORD NOT PRESENT' ausgeben
e265   20  75  e2    jsr   $e275       gewünschten Record holen
e268   a9  80        lda   #$80        Bit 7
e26a   20  a6  dd    jsr   $dda6       testen
e26d   f0  03        beq   $e272       verzweige, wenn nicht gesetzt
e26f   4c  5e  e1    jmp   $e15e       '50, RECORD NOT PRESENT' ausgeber
e272   4c  94  c1    jmp   $c194       Diskstatus bereitstellen; Ende
--------------------------------------------------------------------------
e275                                   Record in aktuellen Puffer holen und den nächs- ten Block
                                       gleich in den zweiten Puffer zr spä- teren Bearbeitung
                                       laden.
e275   20  9c  e2    jsr   $e29c       Block in aktiven Puffer laden
e278   a5  d7        lda   $d7         Zeiger in Record
e27a   20  c8  d4    jsr   $d4c8       Pufferzeiger setzen
e27d   a6  82        ldx   $82         Kanalnummer
e27f   b5  c7        lda   $c7,x       Recordlänge holen
e281   38            sec               mInus
e282   e5  d4        sbc   $d4         Beginn des Records
e284   b0  03        bcs   $e289       verzweige, wenn Ergebnis positiv
e286   4c  02  e2    jmp   $e202       '67, ILLEGAL TRACK OR SECTOR' ausgeben
e289   18            clc               addiere
e28a   65  d7        adc   $d7         Zeiger in Record
e28c   90  03        bcc   $e291       verzweige, wenn Ergebnis kleiner gleich 255
e28e   69  01        adc   #$01        sonst den Oberlauf plus 1 addieren
e290   38            sec               Flag setzen
e291   20  09  e0    jsr   $e009       Zeiger für Ausgabe setzen
e294   4c  38  e1    jmp   $e138       Byte für Ausgabe bereitstellen; Ende
e297   a9  51        lda   #$51        Nummer der Fehlermeldung
e299   20  c8  c1    jsr   $c1c8       '51, OVERFLOW IN RECORD' ausgeben
--------------------------------------------------------------------------
e29c                                   Datenblock in Puffer schreiben.
e29c   a5  94        lda   $94         Pufferzeiger Lo
e29e   85  89        sta   $89         merken
e2a0   a5  95        lda   $95         Pufferzeiger Hi
e2a2   85  8a        sta   $8a         merken
e2a4   20  d0  e2    jsr   $e2d0       gewünschter Block im Puffer?
e2a7   d0  01        bne   $e2aa       verzweige, wenn nein
e2a9   60            rts               Ende
e2aa   20  f1  dd    jsr   $ddf1       Pufferinhalt schreiben, wenn 'dirty'
e2ad   20  0c  de    jsr   $de0c       Folgetrack und -sektor holen
e2b0   a5  80        lda   $80         Tracknummer
e2b2   f0  0e        beq   $e2c2       verzweige, wenn kein Folgeblock vorhanden
e2b4   20  d3  e2    jsr   $e2d3       Track und Sektor vergleichen
e2b7   d0  06        bne   $e2bf       verzweige, wenn ungleich
e2b9   20  1e  cf    jsr   $cf1e       Puffer wechseln
e2bc   4c  da  d2    jmp   $d2da       inaktiven Puffer freimachen; Ende
e2bf   20  da  d2    jsr   $d2da       inaktiven Puffer freimachen
e2c2   a0  00        ldy   #$00        Pufferindex zeigt auf Tracknummr
e2c4   b1  89        lda   ($89),y     Tracknummer holen
e2c6   85  80        sta   $80         und übernehmen
e2c8   c8            iny               Zeiger auf Sektornummer
e2c9   b1  89        lda   ($89),y     Sektornummer holen
e2cb   85  81        sta   $81         und übernehmen
e2cd   4c  af  d0    jmp   $d0af       Block in Puffer lesen; Ende
```

```
------------------------------------------------------------------------
e2d0                                    Kontrolle, ob sich der gewünschte Datenblock im aktuellen
                                        Puffer befindet; dazu werden die Spur- und Sektornummern
                                        des Blocks mit den aktuellen der Zeropage verglichen.
e2d0   20  3e  de    jsr   $de3e       Track und Sektor aus Jobspeicher holen
e2d3   a0  00        ldy   #$00        Pufferindex auf Tracknummer
e2d5   b1  89        lda   ($89),y     Tracknummer holen
e2d7   c5  80        cmp   $80         mit gewünschtem Wert vergleiche
e2d9   f0  01        beq   $e2dc       verzweige, wenn gleich
e2db   60            rts               Ende
e2dc   c8            iny               Zeiger auf Sektornummer
e2dd   b1  89        lda   ($89),y     Sektornummer holen
e2df   c5  81        cmp   $81         mit gewünschtem Wert vergleichen
e2e1   60            rts               Ende
------------------------------------------------------------------------
e2e2                                    Pufferdaten jeweils den einzelnen Records zuordnen.
e2e2   20  2b  de    jsr   $de2b       Pufferzeiger setzen
e2e5   a0  02        ldy   #$02        Index auf erstes Datenbyte
e2e7   a9  00        lda   #$00        Füllbyte
e2e9   91  94        sta   ($94),y     gesamten Puffer löschen
e2eb   c8            iny               Zeiger auf nächstes Byte
e2ec   d0  fb        bne   $e2e9       und weitermachen
e2ee   20  04  e3    jsr   $e304       Position des nächsten Records berechnen
e2f1   95  c1        sta   $c1,x       und merken
e2f3   a8            tay               Position als Index
e2f4   a9  ff        lda   #$ff        $ff als Kennzeichen in Record
e2f6   91  94        sta   ($94),y     setzen
e2f8   20  04  e3    jsr   $e304       Zeiger auf nächsten Record berechnen
e2fb   90  f4        bcc   $e2f1       verzweige, wenn Record noch in diesem Block
e2fd   d0  04        bne   $e303       sonst verzweige, wenn Block voll
e2ff   a9  00        lda   #$00        oder
e301   95  c1        sta   $c1,x       lösche Recordzeiger
e303   60            rts               Ende
------------------------------------------------------------------------
e304                                    Kontrollieren, ob der nächste Record noch in den Puffer
                                        paßt.
e304   a6  82        ldx   $82         Kanalnummer
e306   b5  c1        lda   $c1,x       Recordzeiger holen
e308   38            sec               Fehlerflag setzen
e309   f0  0d        beq   $e318       verzweige, wenn Zeiger gleich Null
e30b   18            clc               Flag löschen
e30c   75  c7        adc   $c7,x       Recordlänge addieren
e30e   90  0b        bcc   $e31b       verzweige, wenn kein Überlauf
e310   d0  06        bne   $e318       verzweige, wenn ungleich Null
e312   a9  02        lda   #$02        Wert für Flag
e314   2c  cc  fe    bit   $fecc       Z-Flag setzen
e317   60            rts               Ende
e318   69  01        adc   #$01        Überlauf plus 1 addieren
e31a   38            sec               Flag für kein weiterer Record
e31b   60            rts               Ende
------------------------------------------------------------------------
e31c                                    Blöcke zum relativen File hinzufügen und die Side-Sektoren
                                        daraufhin aktualisieren.
e31c   20  d3  d1    jsr   $d1d3       Drivenummer holen und setzen
e31f   20  cb  e1    jsr   $e1cb       letzten Side-Sektor holen
e322   20  9c  e2    jsr   $e29c       gewünschte Datenblöcke lesen
```

```
e325   20  7b  cf    jsr   $cf7b       Zweipufferbetrieb generieren
e328   a5  d6        lda   $d6         Zeiger in Side-Sektor
e32a   85  87        sta   $87         merken
e32c   a5  d5        lda   $d5         Side-Sektor-Nummer
e32e   85  86        sta   $86         merken
e330   a9  00        lda   #$00        Flag für einen Block
e332   85  88        sta   $88         löschen
e334   a9  00        lda   #$00        Zeiger auf Beginn des Records
e336   85  d4        sta   $d4         setzen
e338   20  0e  ce    jsr   $ce0e       Side-Sektor-Zeiger berechnen
e33b   20  4d  ef    jsr   $ef4d       'BLOCKS FREE' berechnen
e33e   a4  82        ldy   $82         Kanalnummer
e340   b6  c7        ldx   $c7,y       Recordlänge
e342   ca            dex               minus 1
e343   8a            txa               in A zur
e344   18            clc               weiteren Berechnung
e345   65  d7        adc   $d7         plus Zeiger in Record
e347   90  0c        bcc   $e355       verzweige, wenn kein Überlauf
e349   e6  d6        inc   $d6         sonst Zeiger in Side-Sektor erhöhen
e34b   e6  d6        inc   $d6         zweimal, da Track und Sektor
e34d   d0  06        bne   $e355       verzweige, wenn kein Überlauf
e34f   e6  d5        inc   $d5         sonst Side-Sektor-Nummer erhöhen
e351   a9  10        lda   #$10        16
e353   85  d6        sta   $d6         Zeiger in Side-Sektor setzen
e355   a5  87        lda   $87         vorherigen Zeiger in Side-Sektor
e357   18            clc               plus
e358   69  02        adc   #$02        2
e35a   20  e9  de    jsr   $dee9       Pufferzeiger für Side-Sektor setzen
e35d   a5  d5        lda   $d5         Side-Sektor-Nummer
e35f   c9  06        cmp   #$06        kleiner als 6?
e361   90  05        bcc   $e368       verzweige, wenn ja
e363   a9  52        lda   #$52        Nummer der Fehlermeldung
e365   20  c8  c1    jsr   $c1c8       '52, FILE TOO LARGE' ausgeben
e368   a5  d6        lda   $d6         Zeiger in Side-Sektor
e36a   38            sec               minus
e36b   e5  87        sbc   $87         letztem Zeiger in Side-Sektor
e36d   b0  03        bcs   $e372       verzweige, wenn Ergebnis größer gleich 0
e36f   e9  0f        sbc   #$0f        sonst minus 15
e371   18            clc               und
e372   85  72        sta   $72         Ergebnis merken
e374   a5  d5        lda   $d5         Side-Sektor-Nummer
e376   e5  86        sbc   $86         minus letzte Side-Sektor-Nummer
e378   85  73        sta   $73         merken
e37a   a2  00        ldx   #$00        Summenspeicher
e37c   86  70        stx   $70         vollständig
e37e   86  71        stx   $71         löschen
e380   aa            tax               Differenz nach X
e381   20  51  df    jsr   $df51       Blockzahl der relativen Datei berechnen
e384   a5  71        lda   $71         Blockzahl Hi
e386   d0  07        bne   $e38f       verzweige, wenn ungleich Null
e388   a6  70        ldx   $70         Blockzahl Lo
e38a   ca            dex               minus 1
e38b   d0  02        bne   $e38f       verzweige, wenn Zahl ungleich 1 war
e38d   e6  88        inc   $88         Flag für einen Block setzen
e38f   cd  73  02    cmp   $0273       Anzahl der Blöcke auf der Diskette Hi
e392   90  09        bcc   $e39d       verzweige, wenn relative Datei kleiner
e394   d0  cd        bne   $e363       Fehler, wenn Datei größer
e396   ad  72  02    lda   $0272       freie Blockzahl auf Diskette Lo
```

```
e399   c5  70       cmp   $70        Vergleich mit Zahl der benötigten Blöcke Lo
e39b   90  c6       bcc   $e363      verzweige, wenn relative Datei größer
e39d   a9  01       lda   #$01       Zeiger auf zweites Byte
e39f   20  f6  d4   jsr   $d4f6      zweites Byte aus Puffer holen
e3a2   18           clc              1
e3a3   69  01       adc   #$01       addieren
e3a5   a6  82       ldx   $82        Kanalnummer
e3a7   95  c1       sta   $c1,x      als Recordzeiger setzen
e3a9   20  1e  f1   jsr   $f11e      freien Block in der BAM suchen
e3ac   20  fd  dd   jsr   $ddfd      Track und Sektor in Puffer schre1be
e3af   a5  88       lda   $88        Flag für einen Block
e3b1   d0  15       bne   $e3c8      verzweige, wenn gesetzt
e3b3   20  5e  de   jsr   $de5e      Block auf Diskette schreiben
e3b6   20  1e  cf   jsr   $cf1e      Puffer wechseln
e3b9   20  d0  d6   jsr   $d6d0      Parameter an Diskcontroller übergeben
e3bc   20  1e  f1   jsr   $f11e      freien Block in der BAM suchen
e3bf   20  fd  dd   jsr   $ddfd      Track und Sektor in Puffer
e3c2   20  e2  e2   jsr   $e2e2      Records in Puffer setzen
e3c5   4c  d4  e3   jmp   $e3d4      weiter...
e3c8   20  1e  cf   jsr   $cf1e      Puffer wechseln
e3cb   20  d0  d6   jsr   $d6d0      Parameter an Diskcontroller übergeben
e3ce   20  e2  e2   jsr   $e2e2      Records in Puffer setzen
e3d1   20  19  de   jsr   $de19      Null und Endezeiger in Puffer
e3d4   20  5e  de   jsr   $de5e      Block auf Diskette schreiben
e3d7   20  0c  de   jsr   $de0c      Track und Sektor holen
e3da   a5  80       lda   $80        Tracknummer
e3dc   48           pha              merken
e3dd   a5  81       lda   $81        Sektornummer
e3df   48           pha              merken
e3e0   20  3e  de   jsr   $de3e      Track und Sektor aus Jobspeicher
e3e3   a5  81       lda   $81        Sektornummer
e3e5   48           pha              merken
e3e6   a5  80       lda   $80        Tracknummer
e3e8   48           pha              merken
e3e9   20  45  df   jsr   $df45      Pufferzeiger für Side-Sektor setzen
e3ec   aa           tax              auf Null prüfen
e3ed   d0  0a       bne   $e3f9      verzweige, wenn ungleich Null
e3ef   20  4e  e4   jsr   $e44e      Side-Sektor schreiben; nächsten anlegen
e3f2   a9  10       lda   #$10       Pufferzeiger auf 16
e3f4   20  e9  de   jsr   $dee9      setzen
e3f7   e6  86       inc   $86        Side-Sektor-Nummer plus 1
e3f9   68           pla              Tracknummer zurückholen
e3fa   20  8d  dd   jsr   $dd8d      und in Side-Sektor schreiben
e3fd   68           pla              Sektornummer zurückholen
e3fe   20  8d  dd   jsr   $dd8d      und ebenfalls in Side-Sektor schreiben
e401   68           pla              aktuellen Sektor zurückholen
e402   85  81       sta   $81        und wieder übernehmen
e404   68           pla              aktuellen Track zurückholen
e405   85  80       sta   $80        und ebenfals wieder übernehmen
e407   f0  0f       beq   $e418      verzweige, wenn letzter Block
e409   a5  86       lda   $86        Side-Sektor-Nummer
e40b   c5  d5       cmp   $d5        identisch mit letzterer?
e40d   d0  a7       bne   $e3b6      verzweige, wenn nein
e40f   20  45  df   jsr   $df45      Pufferzeiger in Side-Sektor setzen
e412   c5  d6       cmp   $d6        vergleiche mit Side-Sektor-Zeiger
e414   90  a0       bcc   $e3b6      verzweige, wenn kleiner
e416   f0  b0       beq   $e3c8      verzweige, wenn gleich
e418   20  45  df   jsr   $df45      Pufferzeiger in Side-Sektor setzen
```

```
e41b   48           pha             als Endezeiger merken
e41c   a9  00       lda   #$00       Pufferzeiger auf Null
e41e   20  dc  de   jsr   $dedc      $dedc setzen
e421   a9  00       lda   #$00       Null
e423   a8           tay             Folgetrack
e424   91  94       sta   ($94),y    in Puffer schreiben
e426   c8           iny             jetzt auf Sektor
e427   68           pla             Endezeiger zurückholen
e428   38           sec             1 subtrahiert
e429   e9  01       sbc   #$01       ergibt
e42b   91  94       sta   ($94),y    Anzahl der Bytes dieses Blocks
e42d   20  6c  de   jsr   $de6c      Block auf Diskette schreiben
e430   20  99  d5   jsr   $d599      Jobausführung abwarten und prüfen
e433   20  f4  ee   jsr   $eef4      BAM neu schreiben
e436   20  0e  ce   jsr   $ce0e      Zeiger für relative Datei neu setzen
e439   20  1e  cf   jsr   $cf1e      Puffer wechseln
e43c   20  f8  de   jsr   $def8      richtiger Side.Sektor im Puffer?
e43f   70  03       bvs   $e444      verzweige, wenn nein
e441   4c  75  e2   jmp   $e275      Recordzeiger setzen; Ende
e444   a9  80       lda   #$80       Bit 7
e446   20  97  dd   jsr   $dd97      Kanalfiletyp wiederherstellen
e449   a9  50       lda   #$50       Nummer der Fehlermeldung
e44b   20  c8  c1   jsr   $c1c8      '50, RECORD NOT PRESENT' ausgeben
------------------------------------------------------------------------
e44e                                 Neuen Side-Sektor anlegen und die alten daraufhin anpassen.
e44e   20  1e  f1   jsr   $f11e      freien Block in der BAM suchen
e451   20  1e  cf   jsr   $cf1e      Puffer wechseln
e454   20  f1  dd   jsr   $ddf1      alten Side-Sektor schreiben
e457   20  93  df   jsr   $df93      Puffernummer holen
e45a   48           pha             und merken
e45b   20  c1  de   jsr   $dec1      Puffer löschen
e45e   a6  82       ldx   $82        Kanalnummer
e460   b5  cd       lda   $cd,x      zugehörige Puffernummer für Side.Sektor
e462   a8           tay             merken
e463   68           pla             Puffernummer zurückholen
e464   aa           tax             und als Index
e465   a9  10       lda   #$10       16 Bytes des alten Side-Sektors
e467   20  a5  de   jsr   $dea5      in den neuen Side-Sektor übernehmen
e46a   a9  00       lda   #$00       Pufferzeiger auf 0
e46c   20  dc  de   jsr   $dedc      setzen
e46f   a0  02       ldy   #$02       (Puffer mit altem Side-Sektor)
e471   b1  94       lda   ($94),y    Nummer des Side.Sektors
e473   48           pha             merken
e474   a9  00       lda   #$00       Pufferzeiger für neuen Side-Sektor
e476   20  c8  d4   jsr   $d4c8      auf Null setzen
e479   68           pla             Side-Sektor-Nummer zurückholen
e47a   18           clc             und
e47b   69  01       adc   #$01       plus 1
e47d   91  94       sta   ($94),y    als Nummer des neuen Side-Sektors nehmen
e47f   0a           asl             mal 2
e480   69  04       adc   #$04       plus 4
e482   85  89       sta   $89        als Zeiger für Track/Sektor merken
e484   a8           tay             als Index
e485   38           sec             und
e486   e9  02       sbc   #$02       minus 2
e488   85  8a       sta   $8a        als Zeiger auf alten Side-Sektor
e48a   a5  80       lda   $80        aktuelle Tracknummer
```

```
e48c   85 87       sta  $87        merken
e48e   91 94       sta  ($94),y    und in Puffer schreiben
e490   c8          iny             Zeiger plus 1
e491   a5 81       lda  $81        aktuelle Sektornummer
e493   85 88       sta  $88        merken
e495   91 94       sta  ($94),y    und in Puffer schreiben
e497   a0 00       ldy  #$00       Index auf erstes Byte im Puffer
e499   98          tya             Tracknummer 0
e49a   91 94       sta  ($94),y    als Marke für letzten Side-Sektor in Puffer
e49c   c8          iny             Zeiger auf Sektornummer
e49d   a9 11       lda  #$11       17
e49f   91 94       sta  ($94),y    als Anzahl der Bytes im Block merken
e4a1   a9 10       lda  #$10       Pufferzeiger auf 16
e4a3   20 c8 d4    jsr  $d4c8      setzen
e4a6   20 50 de    jsr  $de50      Block auf Diskette schreiben
e4a9   20 99 d5    jsr  $d599      Jobausführung abwarten und prüfen
e4ac   a6 82       ldx  $82        Kanalnummer
e4ae   b5 cd       lda  $cd,x      zugehörige Puffernummer zu Side-Sektor
e4b0   48          pha             merken
e4b1   20 9e df    jsr  $df9e      Puffernummer holen
e4b4   a6 82       ldx  $82        Kanalnummer
e4b6   95 cd       sta  $cd,x      Puffernummer in Tabelle schreiben
e4b8   68          pla             Puffernummer des Side-Sektors zurückholen
e4b9   ae 57 02    ldx  $0257      Nummer des zuletzt aktiven Puffers
e4bc   95 a7       sta  $a7,x      Puffer als belegt kennzeichnen
e4be   a9 00       lda  #$00       Pufferzeiger auf 0
e4c0   20 c8 d4    jsr  $d4c8      setzen
e4c3   a0 00       ldy  #$00       Pufferindex auf Tracknummer
e4c5   a5 80       lda  $80        aktuelle Tracknummer holen
e4c7   91 94       sta  ($94),y    und in Puffer eintragen
e4c9   c8          iny             Pufferindex zeigt jetzt auf Sektornummer
e4ca   a5 81       lda  $81        aktuelle Sektornummer holen
e4cc   91 94       sta  ($94),y    und in Puffer eintragen
e4ce   4c de e4    jmp  $e4de      weiter.,.
e4d1   20 93 df    jsr  $df93      Puffernummer holen
e4d4   a6 82       ldx  $82        Kanalnummer
e4d6   20 1b df    jsr  $df1b      nächsten Side-Sektor von Diskette lesen
e4d9   a9 00       lda  #$00       Pufferzeiger auf 0
e4db   20 c8 d4    jsr  $d4c8      setzen
e4de   c6 8a       dec  $8a        Zähler
e4e0   c6 8a       dec  $8a        minus 2
e4e2   a4 89       ldy  $89        Zeiger für Track/Sektor
e4e4   a5 87       lda  $87        Tracknummer holen
e4e6   91 94       sta  ($94),y    und in Puffer schreiben
e4e8   c8          iny             Zeiger auf Sektornummer
e4e9   a5 88       lda  $88        Sektornummer holen
e4eb   91 94       sta  ($94),y    und ebenfalls in Puffer schreiben
e4ed   20 5e de    jsr  $de5e      Side-Sektor auf Diskette schreiben
e4f0   20 99 d5    jsr  $d599      Jobausführung abwarten und prüfen
e4f3   a4 8a       ldy  $8a        Zähler
e4f5   c0 03       cpy  #$03       vergleiche mit 3
e4f7   b0 d8       bcs  $e4d1      weitermachen, wenn noch größer gleich 3
e4f9   4c 1e cf    jmp  $cf1e      Puffer wechseln; Ende
```

```
------------------------------------------------------------------------
e4fc                                Tabelle mit den ASCII-Codes aller Fehlermeldungen des DOS
                                    3.0.
e4fc 00 a0 4f cb                    00, oK
e500 20 21 22 23 24 27 d5 45        20/21/22/23/24/27, Read ErrorR
e508 41 44 89
e50b 52 83 20 54 4f 4f 20 4c        52, FilE too largE
e513 41 52 47 c5
e517 50 8b 06 20 50 52 45 53        50, RecorD NoT presenT
e51f 45 4e d4
e522 51 cf 56 45 52 46 4c 4f        51, Overflow in RecorD
e52a 57 20 49 4e 8b
e52f 25 28 8a 89                    25/28, WritE ErroR
e533 26 8a 20 50 52 4f 54 45        26, WritE protect oN
e53b 43 54 20 49 44 85
e540 29 88 20 49 44 85              29, DisK id MismatcH
e546 30 31 32 33 34 d3 59 4e        30/31/32/33/34, Syntax ErroR
e54e 54 41 58 89
e552 60 8a 03 84                    60, WritE FilE OpeN
e556 63 83 20 45 58 49 53 54        63, FilE existS
e55e d3
e55f 64 83 20 54 59 50 45 85        64, FilE type MismatcH
e567 65 ce 4f 20 42 4c 4f 43        65, No blocK
e56f cb
e570 66 67 c9 4c 4c 45 47 41        66/67, Illegal track or sektoR
e578 4c 20 54 52 41 43 4b 20
e580 4f 52 20 53 45 43 54 4f
e588 d2
e589 61 83 06 87                    61, FilE NoT OpeN
e58d 39 62 83 06 87                 39/62, FilE NoT FounD
e592 01 83 53 20 53 43 52 41        01, FilE 's scratcheD
e59a 54 43 48 54 c4
e59f 70 ce 4f 20 43 48 41 4e        70, No channeL
e5a7 4e 45 cc
e5aa 71 c4 49 52 89                 71, Dir ErroR
e5af 72 88 20 46 55 4c cc           72, DisK fulL
e5b6 73 c3 42 4d 20 44 4f 53        73, Cbm dos v3.0 1571
e5be 20 56 33 2e 30 20 31 35
e5c6 37 b1 bzw. b0 für 1570
e5c8 74 c4 52 49 56 45 06 20        74, Drive NoT readY
e5d0 52 45 41 44 d9
e5d5 09 c5 52 52 4f d2              ErroR
e5db 0a d7 52 49 54 c5              WritE
e5e1 03 c6 49 4c c5                 FilE
e5e6 04 cf 50 45 ce                 OpeN
e5eb 05 cd 49 53 4d 41 54 43        MismatcH
e5f3 c8
e5f4 06 ce 4f d4                    NoT
e5f8 07 c6 4f 55 4e c4              FounD
e5fe 08 c4 49 53 cb                 DisK
e603 0b d2 45 43 4f 52 c4           RecorD
------------------------------------------------------------------------
e60a                                Fehlerbehandlung nach Ausführung eines Jobs, wobei A die
                                    Fehlernummer und X die Puffernummer enthalten muß.
e60a   4c  b9  a9    jmp   $a9b9   Erweiterte Behandlung bei DOS 3.0
e60d   8a           txa             Puffernummer
e60e   0a           asl             mal 2
```

```
e60f   aa          tax             als Index
e610   b5  06      lda   $06,x     Tracknummer aus Jobspeicher holen
e612   85  80      sta   $80       und für Ausgabe übernehmen
e614   b5  07      lda   $07,x     Sektornummer aus Jobspeicher holen
e616   85  81      sta   $81       und für Ausgabe übernehmen
e618   68          pla             Fehlernummer zurückholen
e619   29  0f      and   #$0f      auf Fehlernummer $10 prüfen
e61b   f0  08      beq   $e625     verzweige, wenn Fehler $10
e61d   c9  0f      cmp   #$0f      auf Fehlernummer $Of testen
e61f   d0  06      bne   $e627     verzweige, wenn nein
e621   a9  74      lda   #$74      Nummer der Fehlermeldung
e623   d0  08      bne   $e62d     unbedingter Sprung zu '74, DRIVE NOT READY'
e625   a9  06      lda   #$06      6
e627   09  20      ora   #$20      plus 32
e629   aa          tax             nach X
e62a   ca          dex             und
e62b   ca          dex             minus 2
e62c   8a          txa             ergibt $24
e62d   48          pha             Fehlernummer merken; '24, READ ERROR'
e62e   ad  2a  02  lda   $022a     Nummer des auszufürenden Befehls
e631   c9  00      cmp   #$00      VALIDATE?
e633   d0  0f      bne   $e644     verzweige, wenn nein
e635   a9  ff      lda   #$ff      Befehlsnummer
e637   8d  2a  02  sta   $022a     löschen
e63a   68          pla             Fehlernummer zurückholen
e63b   20  c7  e6  jsr   $e6c7     Fehlermeldung in Puffer
e63e   20  42  d0  jsr   $d042     Diskette initialisieren
e641   4c  48  e6  jmp   $e648     weiter...
e644   68          pla             Fehlernummer zurückholen
e645   20  c7  e6  jsr   $e6c7     Fehlermeldung in Puffer
e648   20  bd  c1  jsr   $c1bd     INPUT-Puffer löschen
e64b   a9  00      lda   #$00      Flag für BAM auf Diskette schreiben
e64d   8d  f9  02  sta   $02f9     löschen; Schreiben verhindern
e650   20  2c  c1  jsr   $c12c     LED-Blinken aktivieren
e653   20  da  d4  jsr   $d4da     Lese-/Schreibkanäle schließen
e656   a9  00      lda   #$00      Zeiger in INPUT-Puffer
e658   85  a3      sta   $a3       löschen
e65a   a2  45      ldx   #$45      Stackpointer
e65c   9a          txs             initialisieren
e65d   a5  84      lda   $84       übliche Sekundäradresse
e65f   29  0f      and   #$0f      setzen
e661   85  83      sta   $83       und übernehmen
e663   c9  0f      cmp   #$0f      mit 15 (Kommandokanal) vergleichen
e665   f0  31      beq   $e698     verzweige, wenn 15
e667   78          sei             Diskcontroller abschalten
e668   a5  79      lda   $79       Flag für LISTEN gesetzt?
e66a   d0  1c      bne   $e688     verzweige, wenn ja
e66c   a5  7a      lda   $7a       Flag für TALK gesetzt?
e66e   d0  10      bne   $e680     verzweige, wenn ja
e670   a6  83      ldx   $83       Sekundäradresse
e672   bd  2b  02  lda   $022b,x   zugehöriger Kanalstatus
e675   c9  ff      cmp   #$ff      Kanal aktiv?
e677   f0  1f      beq   $e698     verzweige, wenn nein
e679   29  0f      and   #$0f      Kanalnummer isolieren
e67b   85  82      sta   $82       und übernehmen
e67d   4c  8e  e6  jmp   $e68e     weiter...
```

```
------------------------------------------------------------------------
e680                                    Fehlerbehandlung bei TALK vom Bus.
e680   20  eb  d0    jsr   $d0eb       Kanal zum Lesen suchen
e683   ea            nop               Rest vom
e684   ea            nop               DOS V2.6
e685   ea            nop               1541
e686   d0  06        bne   $e68e       unbedingter Sprung
------------------------------------------------------------------------
e688                                    Fehlerbehandlung bei LISTEN vom Bus.
e688   20  07  d1    jsr   $d107       Kanal zum Schreiben suchen
e68b   ea            nop               Rest vom
e68c   ea            nop               DOS V2.6
e68d   ea            nop               der 1541
e68e   20  25  d1    jsr   $d125       Filetyp holen
e691   c9  04        cmp   #$04        relative Datei?
e693   b0  03        bcs   $e698       verzweige, wenn ja
e695   20  27  d2    jsr   $d227       alle Kanäle außer dem Kommandokanal freigeben
e698   4c  6b  83    jmp   $836b       1571 zurücksetzen; zur Warteschleife
------------------------------------------------------------------------
e69b                                    Wandelt eine Hex-Zahl in eine Dezimalzahl um, wobei A beim
                                        Einsprung den Hex-Wert enthalten muß. Bei der Rückkehr
                                        enthält A das Ergebnis.
e69b   aa            tax               Wert retten
e69c   a9  00        lda   #$00        A initialisieren
e69e   f8            sed               auf Dezimalrechnung umschalten
e69f   e0  00        cpx   #$00        Hex-Byte gleich 0?
e6a1   f0  07        beq   $e6aa       verzweige, wenn ja
e6a3   18            clc               Im Akku wird nun
e6a4   69  01        adc   #$01        der Wert durch
e6a6   ca            dex               schrittweises Addieren
e6a7   4c  9f  e6    jmp   $e69f       aufgebaut
e6aa   d8            cld               auf Binärrechnung umschalten
e6ab   aa            tax               Dezimalwert retten
e6ac   4a            lsr               und
e6ad   4a            lsr               viermal rechtsverschieben, um
e6ae   4a            lsr               an den Wert des Hi-Nibbles
e6af   4a            lsr               heranzukommen
e6b0   20  b4  e6    jsr   $e6b4       Nibble in ASCII-Wert umwandeln
e6b3   8a            txa               Dezimalzahl zurückholen
e6b4   29  0f        and   #$0f        und Lo-Nibble ebenfalls
e6b6   09  30        ora   #$30        in eine ASCII.Ziffer umwandeln
e6b8   91  a5        sta   ($a5),y     und den Wert in den Puffer schreiben
e6ba   c8            iny               Pufferzeiger plus 1
e6bb   60            rts               Ende
------------------------------------------------------------------------
e6bc                                    Fehlermeldung in Fehlerpuffer schreiben. Bei Einsprung ab
                                        $e6bc wird die OK-Meldung im Puffer generiert.
e6bc   20  23  c1    jsr   $c123       Löschen des Fehlerstatus
e6bf   a9  00        lda   #$00        Code für OK-Meldung
e6c1   a0  00        ldy   #$00        Parameterwert 0
e6c3   84  80        sty   $80         Track gleich 0
e6c5   84  81        sty   $81         Sektor gleich 0

e6c7   a0  00        ldy   #$00        Puffer index auf Null
e6c9   a2  d5        ldx   #$d5        Lo.Byte für Fehlerpuffer
e6cb   86  a5        stx   $a5         Adresse Lo setzen
e6cd   a2  02        ldx   #$02        Hi-Byte für Fehlerpuffer
```

```
e6cf   86 a6       stx  $a6         Adresse Hi setzen
e6d1   20 ab e6    jsr  $e6ab       $e6ab Fehlernummer als ASCII in Puffer
e6d4   a9 2c       lda  #$2c        ',' Komma
e6d6   91 a5       sta  ($a5),y     in Puffer schreiben
e6d8   c8          iny              Zeiger erhöhen
e6d9   ad d5 02    lda  $02d5       Hi-Anteil der ASCII-Fehlernummer
e6dc   8d 43 02    sta  $0243       für Ausgabe bereitstellen
e6df   8a          txa              Fehlernummer
e6e0   20 06 e7    jsr  $e706       Text der Fehlermeldung in Puffer
e6e3   a9 2c       lda  #$2c        ',' Komma
e6e5   91 a5       sta  ($a5),y     in Puffer
e6e7   c8          iny              Zeiger erhöhen
e6e8   a5 80       lda  $80         Tracknummer
e6ea   20 9b e6    jsr  $e69b       in ASCII umwandeln und in Puffer
e6ed   a9 2c       lda  #$2c        ',' Komma
e6ef   91 a5       sta  ($a5),y     in Puffer
e6f1   c8          iny              Zeiger erhöhen
e6f2   a5 81       lda  $81         Sektornummer
e6f4   20 9b e6    jsr  $e69b       in ASCII umwandeln und in Puffer
e6f7   88          dey              Zeiger minus 1
e6f8   98          tya              und
e6f9   18          clc              Addition vorbereiten
e6fa   69 d5       adc  #$d5        Länge der Fehlermeldung setzen
e6fc   8d 49 02    sta  $0249       und abspeichern
e6ff   e6 a5       inc  $a5         Pufferadresse Lo auf zweites Byte setzen
e701   a9 88       lda  #$88        READY TO TALK Status
e703   85 f7       sta  $f7         setzen
e705   60          rts              Ende
--------------------------------------------------------------------------
e706                                Schreibt den Text für die Fehlermeldung aus der ASCII-
                                    Tabelle in de Fehlerpuffer.
e706   aa          tax              Fehlernummer retten
e707   a5 86       lda  $86         Wert
e709   48          pha              retten
e70a   a5 87       lda  $87         Wert
e70c   48          pha              retten
e70d   a9 fc       lda  #$fc        Adresse Lo der Systemmeldungen
e70f   85 86       sta  $86         setzen
e711   a9 e4       lda  #$e4        Adresse Hi der Systemmeldungen
e713   85 87       sta  $87         setzen
e715   8a          txa              Fehlernummer zurückholen
e716   a2 00       ldx  #$00        Fehlernummer in
e718   c1 86       cmp  ($86,x)     Tabelle suchen
e71a   f0 21       beq  $e73d       verzweige, wenn gefunden
e71c   48          pha              Fehlernummer retten
e71d   20 75 e7    jsr  $e775       Bit 7 ins Carry; im Byte löschen
e720   90 05       bcc  $e727       verzweige, wenn nicht gesetzt
e722   20 75 e7    jsr  $e775       Bit 7 ins Carry; im Byte löschen
e725   90 fb       bcc  $e722       verzweige, wenn nicht gesetzt
e727   a5 87       lda  $87         Zeiger in Tabelle Hi
e729   c9 e6       cmp  #$e6        vergleiche mit $e6
e72b   90 08       bcc  $e735       verzweige, wenn kleiner
e72d   d0 0a       bne  $e739       verzweige, wenn ungleich
e72f   a9 0a       lda  #$0a        10
e731   c5 86       cmp  $86         vergleiche mit Zeiger Lo
e733   90 04       bcc  $e739       verzweige, wenn kleiner
e735   68          pla              Fehlernummer zurückholen
e736   4c 18 e7    jmp  $e718       Suche fortführen
```

```
e739   68          pla             Stack wiederherstellen
e73a   4c  4d  e7   jmp   $e74d     Parameter wieder zurückholen; Ene
e73d   20  67  e7   jsr   $e767     Zeichen holen; Bit 7 ins Carry
e740   90  fb       bcc   $e73d     verzweige, wenn Bit 7 gelöscht
e742   20  54  e7   jsr   $e754     Zeichen in Fehlerpuffer schre;:-
e745   20  67  e7   jsr   $e767     Zeichen holen; Bit 7 ins Carry
e748   90  f8       bcc   $e742     verzweige, wenn Bit 7 gelöscht
e74a   20  54  e7   jsr   $e754     letztes Zeichen in Puffer schrei-
e74d   68          pla             Wert zurückholen
e74e   85  87       sta   $87       wieder in Zwischenspeicher
e750   68          pla             Wert zurückholen
e751   85  86       sta   $86       wieder in Zwischenspeicher
e753   60          rts             Ende
--------------------------------------------------------------------------
e754                               Prüft auf Kontrollcodes oder ASCII-Codes; im Falle eines
                                   ASCII.Zeichens wird dieses in den Puffer geschrieben;
                                   ansonsten erfolgt die Übergabe des Codes in x.
e754   c9  20       cmp   #$20      ' ' SPACE
e756   b0  0b       bcs   $e763     verzweige, wenn größer
e758   aa          tax             ASCII.Code merken
e759   a9  20       lda   #$20      ' ' SPACE
e75b   91  a5       sta   ($a5),y   in Puffer schreiben
e75d   c8          iny             Pufferindex erhöhen
e75e   8a          txa             ASCII.Code zurückholen
e75f   20  06  e7   jsr   $e706     entsprechende Meldung ausgeben
e762   60          rts             Ende
e763   91  a5       sta   ($a5),y   Zeichen in Fehlerpuffer schreiben
e765   c8          iny             Pufferindex erhöhen
e766   60          rts             Ende
--------------------------------------------------------------------------
e767                               Zeiger in Tabelle erhöhen und Byte aus der Tabelle holen.
                                   Bit 7 dieses Bytes in Carry.
e767   e6  86       inc   $86       Zeiger Lo erhöhen
e769   d0  02       bne   $e76d     verzweige, wenn kein Überlauf
e76b   e6  87       inc   $87       sonst Zeiger Hi erhöhen
e76d   a1  86       lda   ($86,x)   Zeichen aus ASCII.Tabelle holen
e76f   0a          asl             Bit 7 ins Carry.Flag schieben
e770   a1  86       lda   ($86,x)   Zeichen nochmal aus Tabelle
e772   29  7f       and   #$7f      Bit 7 im Zeichen löschen
e774   60          rts             Ende
--------------------------------------------------------------------------
e775                               Zeichen aus Fehlertabelle holen und Zeiger in Tabelle
                                   erhöhen.
e775   20  6d  e7   jsr   $e76d     Byte aus Tabelle; Bit 7 ins Carry
e778   e6  86       inc   $86       Zeiger Lo plus 1
e77a   d0  02       bne   $e77e     verzweige, wenn kein Überlauf
e77c   e6  87       inc   $87       sonst Zeiger Hi ebenfals plus 1
e77e   60          rts             Ende
e77f   60          rts             Ende
--------------------------------------------------------------------------
e780                               Überrest einer AUTOBOOT.Routine, die in den älteren
                                   Versionen des DOS 2.6 der 1541 noch existierte.
e780   60          rts             bei Aufruf sofortiger Rücksprung
e781   ea ...      nop             mit NOP's aufgefüllter
e7a1   ... ea      nop             Speicherbereich
e7a2   60          rts             Ende der ehemaligen AUTOBOOT.Routine
```

```
------------------------------------------------------------------------
e7a3                                    &-Befehl ('utility loader').
e7a3    4c  fe  a5      jmp   $a5fe     Sa5fe Erweiterung für 1571-Modus
e7a6    ea              nop             übriggebliebener
e7a7    ea              nop             Platzrest
e7a8    20  58  f2      jsr   $f258     (RTS)
e7ab    ad  78  02      lda   $0278     Anzahl der Filenamen
e7ae    48              pha             merken
e7af    a9  01          lda   #$01      1 als neue Anzahl
e7b1    8d  78  02      sta   $0278     setzen
e7b4    a9  ff          lda   #$ff      Flag für Startadresse
e7b6    85  86          sta   $86       setzen
e7b8    20  4f  c4      jsr   $c44f     File im Directory suchen
e7bb    ad  80  02      lda   $0280     Tracknummer des Files
e7be    d0  05          bne   $e7c5     verzweige, wenn File gefunden
e7c0    a9  39          lda   #$39      Nummer der Fehlermeldung
e7c2    20  c8  c1      jsr   $c1c8     '39, FILE NOT FOUND' ausgeben
e7c5    68              pla             Zahl der Filenamen zurückholen
e7c6    8d  78  02      sta   $0278     und wieder übernehmen
e7c9    ad  80  02      lda   $0280     Tracknummer des Files
e7cc    85  80          sta   $80       übernehmen
e7ce    ad  85  02      lda   $0285     Sektornummer des Files
e7d1    85  81          sta   $81       ebenfalls Übernehmen
e7d3    a9  03          lda   #$03      Code für Filetyp USR
e7d5    20  77  d4      jsr   $d477     File öffnen; ersten Block lesen
e7d8    a9  00          lda   #$00      Prüfsumme mit Defaultwert
e7da    85  87          sta   $87       vorbesetzen
e7dc    20  39  e8      jsr   $e839     Byte aus Puffer holen
e7df    85  88          sta   $88       und als Programmstartadresse Lo merken
e7e1    20  4b  e8      jsr   $e84b     Wert zu Prüfsumme addieren
e7e4    20  39  e8      jsr   $e839     Byte aus Puffer holen
e7e7    85  89          sta   $89       und als Programmstartadresse Hi merken
e7e9    20  4b  e8      jsr   $e84b     Wert zu Prüfsumme addieren
e7ec    a5  86          lda   $86       Startadressen Flag prüfen
e7ee    f0  0a          beq   $e7fa     verzweige, wenn Adresse schon geholt
e7f0    a5  88          lda   $88       Programmstartadresse Lo
e7f2    48              pha             merken
e7f3    a5  89          lda   $89       Programmstartadresse Hi
e7f5    48              pha             merken
e7f6    a9  00          lda   #$00      Flag für Adresse geholt
e7f8    85  86          sta   $86       setzen
e7fa    20  39  e8      jsr   $e839     Byte aus Puffer holen
e7fd    85  8a          sta   $8a       und als Anzahl der nachfolgenden Bytes merken
e7ff    20  4b  e8      jsr   $e84b     Wert zu Prüfsumme addieren
e802    20  39  e8      jsr   $e839     Byte aus Puffer holen
e805    a0  00          ldy   #$00      Byte als Programmbyte
e807    91  88          sta   ($88),y   abspeichern
e809    20  4b  e8      jsr   $e84b     Wert zu Prüfsumme addieren
e80c    a5  88          lda   $88       Programmstartadresse Lo
e80e    18              clc             plus 1
e80f    69  01          adc   #$01      zeigt damit auf
e811    85  88          sta   $88       das nächste Byte
e813    90  02          bcc   $e817     verzweige, wenn kein Übertrag
e815    e6  89          inc   $89       sonst auch Adresse Hi erhöhen
e817    c6  8a          dec   $8a       Anzahl der Bytes vermindern
e819    d0  e7          bne   $e802     und weitere Bytes holen, wenn nötig
e81b    20  35  ca      jsr   $ca35     noch ein Byte holen; ohne Test auf EOI
e81e    a5  85          lda   $85       Datenbyte mit
```

```
e820   c5  87       cmp   $87        Prüfsumme vergleichen
e822   f0  08       beq   $e82c       verzweige, wenn Prüfsumme stimmt
e824   20  3e  de   jsr   $de3e       Parameter für Fehlermeldung an Diskcontroller
e827   a9  50       lda   #$50        Nummer der Fehlermeldung
e829   20  45  e6   jsr   $e645       '50, RECORD NOT PRESENT' ausgee-
e82c   a5  f8       lda   $f8         auf EOI testen
e82e   d0  a8       bne   $e7d8       weitermachen, wenn noch kein EOI
e830   68           pla               Programmstartadresse Hi zurückholen
e831   85  89       sta   $89         und setzen
e833   68           pla               Programmstartadresse Lo zurückholen
e834   85  88       sta   $88         und ebenfalls setzen
e836   6c  88  00   jmp   ($0088)     Programm im Speicher ausführen
e839   20  35  ca   jsr   $ca35       Byte aus Puffer; ohne Test auf EOI
e83c   a5  f8       lda   $f8         EOI gesendet?
e83e   d0  08       bne   $e848       verzweige, wenn ja
e840   20  3e  de   jsr   $de3e       Parameter für Fehlermeldung an Diskcontroller
e843   a9  51       lda   #$51        Nummer der Fehlermeldung
e845   20  45  e6   jsr   $e645       '51, OVERFLOW IN RECORD' ausgeben
e848   a5  85       lda   $85         Datenbyte holen
e84a   60           rts               Ende
-------------------------------------------------------------------------
e84b                                  Prüfsumme bilden.
e84b   18           clc               Übertrag löschen
e84c   65  87       adc   $87         Byte zu Prüfsumme addieren
e84e   69  00       adc   #$00        Übertrag addieren (0 oder 1)
e850   85  87       sta   $87         und Summe wieder abspeichern
e852   60           rts               Ende
-------------------------------------------------------------------------
e853                                  IRQ-Routine für den seriellen Bus, die jedesmal ein
                                      eventuelles ATN vom Computer anzeigt.
e853   ad  01  18   lda   $1801       Port A (Buscontroller) lesen; IRQ-Flag löschen
e856   a9  01       lda   #$01        Flag für ATN
e858   85  7c       sta   $7c         setzen
e85a   60           rts               Ende; zum IRQ-Programm
-------------------------------------------------------------------------
e85b                                  Routine zur Bedienung des seriellen Bus nach Auftreten
                                      eines ATN.
e85b   78           sei               Diskcontroller ausschalten
e85c   a9  00       lda   #$00        Löschen:
e85e   85  7c       sta   $7c         Flag für ATN
e860   85  79       sta   $79         Flag für LISTEN
e862   85  7a       sta   $7a         Flag für TALK
e864   a2  45       ldx   #$45        Stackpointer
e866   9a           txs               zurücksetzen
e867   a9  80       lda   #$80        EOI-Flag
e869   85  f8       sta   $f8         löschen
e86b   85  7d       sta   $7d         Flag für ATN-Modus setzen
e86d   20  b7  e9   jsr   $e9b7       CLOCK OUT Hi setzen
e870   20  a5  e9   jsr   $e9a5       DATA OUT Lo setzen
e873   ad  00  18   lda   $1800       Bus lesen
e876   09  10       ora   #$10        Antwortleitung für ATN löschen
e878   8d  00  18   sta   $1800       und Signal auf Bus geben
e87b   ad  00  18   lda   $1800       Bus lesen
e87e   10  57       bpl   $e8d7       verzweige, wenn ATN zurückgesetzt ist
e880   29  04       and   #$04        ATN noch vorhanden; CLOCK IN testen
e882   d0  f7       bne   $e87b       warten, bis CLOCK IN Hi wird
e884   20  c9  e9   jsr   $e9c9       Kommandobyte vom Bus holen
e887   c9  3f       cmp   #$3f        UNLISTEN?
```

```
e889   d0  06       bne   $e891       verzweige, wenn nein
e88b   a9  00       lda   #$00        Flag für LISTEN
e88d   85  79       sta   $79         löschen
e88f   f0  71       beq   $e902       unbedingter Sprung
e891   c9  5f       cmp   #$5f        UNTALK?
e893   d0  06       bne   $e89b       verzweige, wenn nicht
e895   a9  00       lda   #$00        Flag für TALK
e897   85  7a       sta   $7a         löschen
e899   f0  67       beq   $e902       unbedingter Sprung
e89b   c5  78       cmp   $78         TALK-Adresse?
e89d   d0  0a       bne   $e8a9       verzweige, wenn nein
e89f   a9  01       lda   #$01        Flag für TALK
e8a1   85  7a       sta   $7a         TALK setzen
e8a3   a9  00       lda   #$00        Flag für LISTEN
e8a5   85  79       sta   $79         löschen
e8a7   f0  29       beq   $e8d2       unbedingter Sprung
e8a9   c5  77       cmp   $77         LISTEN-Adresse?
e8ab   d0  0a       bne   $e8b7       verzweige, wenn nein
e8ad   a9  01       lda   #$01        Flag für LISTEN
e8af   85  79       sta   $79         setzen
e8b1   a9  00       lda   #$00        Flag für TALK
e8b3   85  7a       sta   $7a         löschen
e8b5   f0  1b       beq   $e8d2       unbedingter Sprung
e8b7   aa           tax               Kommandobyte merken
e8b8   29  60       and   #$60        Bits 5 und 6
e8ba   c9  60       cmp   #$60        testen
e8bc   d0  3f       bne   $e8fd       verzweige, wenn ein Bit gesetzt ist
e8be   8a           txa               Kommandobyte zurückholen
e8bf   85  84       sta   $84         als Befehlssekundäradresse merken
e8c1   29  0f       and   #$0f        reine Sekundäradresse isolieren
e8c3   85  83       sta   $83         und abspeichern
e8c5   a5  84       lda   $84         Befehlssekundäradresse holen
e8c7   29  f0       and   #$f0        und Kommandobits isolieren
e8c9   c9  e0       cmp   #$e0        CLOSE-Kommando?
e8cb   d0  35       bne   $e902       verzweige, wenn nein
e8cd   58           cli               Diskcontroller wieder aktivieren
e8ce   20  c0  da   jsr   $dac0       CLOSE ausführen
e8d1   78           sei               Diskcontroller inaktivieren
e8d2   2c  00  18   bit   $1800       ATN immer noch gesetzt?
e8d5   30  ad       bmi   $e884       wieder versuchen, wenn ja
e8d7   a9  00       lda   #$00        ATN-Modus ist beendet
e8d9   85  7d       sta   $7d         Flag für ATN-Modus löschen
e8db   ad  00  18   lda   $1800       Bus lesen
e8de   29  ef       and   #$ef        Antwortsignal auf ATN setzen
e8e0   8d  00  18   sta   $1800       und senden
e8e3   a5  79       lda   $79         Flag für LISTEN gesetzt?
e8e5   f0  06       beq   $e8ed       verzweige, wenn nein
e8e7   20  2e  ea   jsr   $ea2e       Daten vom Bus in den Puffer schreiben
e8ea   4c  6b  83   jmp   $836b       Busbehandlung für 1571-Erweiterung
e8ed   a5  7a       lda   $7a         Flag für TALK gesetzt?
e8ef   f0  09       beq   $e8fa       verzweige, wenn nein
e8f1   20  9c  e9   jsr   $e99c       DATA OUT Hi setzen
e8f4   20  ae  e9   jsr   $e9ae       CLOCK OUT Lo setzen
e8f7   20  09  e9   jsr   $e909       Daten aus Puffer auf Bus ausgeben
e8fa   4c  4e  ea   jmp   $ea4e       zurück zur Warteschleife
e8fd   a9  10       lda   #$10        alle Leitungen bis auf die ATN ACK (Antwort-
e8ff   8d  00  18   sta   $1800       leitung für ATN) löschen
e902   2c  00  18   bit   $1800       Bus prüfen
```

```
e905   10  d0       bpl   $e8d7       $e8d7 Ende, wenn ATN nicht gesetzt
e907   30  f9       bmi   $e902       $e902 sonst warten, bis ATN zurückgesetzt wird
-------------------------------------------------------------------------------
e909                                  Routine zum Senden von Daten auf den Bus, als Folge eines
                                      TALK-Kommandos vom Computer.
e909   78           sei               Diskcontroller inaktivieren
e90a   20  eb  d0    jsr   $d0eb       freien Kanal zum Lesen suchen
e90d   b0  06       bcs   $e915       verzweige, wenn kein Kanal frei
e90f   a6  82       ldx   $82         Kanalnummer
e911   b5  f2       lda   $f2,x       Kanalstatus prüfen
e913   30  01       bmi   $e916       verzweige, wenn Status ok
e915   60           rts               Ende
e916   20  59  ea    jsr   $ea59       auf ATN-Signal prüfen
e919   20  c0  e9    jsr   $e9c0       warten, bis CLOCK IN Lo wird
e91c   29  01       and   #$01         Datenbit isolieren
e91e   08           php               und dessen Wertigkeit (Z-Flag=O oder 1) merken
e91f   20  b7  e9    jsr   $e9b7       CLOCK OUT-Leitung auf Hi setzen
e922   28           plp               Datenbit zurückholen
e923   f0  12       beq   $e937       verzweige, wenn Bit gleich 0 war
e925   20  59  ea    jsr   $ea59       auf ATN-Signal prüfen
e928   20  c0  e9    jsr   $e9c0       warten, bis CLOCK IN Lo wird
e92b   29  01       and   #$01         Datenbit isolieren
e92d   d0  f6       bne   $e925       verzweige, wenn Bit gleich 1 ist
e92f   a6  82       ldx   $82         Kanalnummer
e931   b5  f2       lda   $f2,x       Kanalstatus holen
e933   29  08       and   #$08         auf EOI testen
e935   d0  14       bne   $e94b       verzweige, wenn kein EOI

                                      Die folgenden Befehle senden das EOI zum Computer weiter:
e937   20  59  ea    jsr   $ea59       auf ATN-Signal prüfen
e93a   20  c0  e9    jsr   $e9c0       EOI senden; DATA-Leitung prüfen
e93d   29  01       and   #$01         Datenbit isolieren
e93f   d0  f6       bne   $e937       auf Reaktion des Computers warten
e941   20  59  ea    jsr   $ea59       auf ATN-Signal prüfen
e944   20  c0  e9    jsr   $e9c0       DATA-Leitung prüfen
e947   29  01       and   #$01         Datenbit isolieren
e949   f0  f6       beq   $e941       auf Reaktion des Computers warten
e94b   20  ae  e9    jsr   $e9ae       CLOCK OUT auf Lo setzen
e94e   20  59  ea    jsr   $ea59       auf ATN-Signal prüfen
e951   20  c0  e9    jsr   $e9c0       DATA-Leitung prüfen
e954   29  01       and   #$01         Datenbit isolieren
e956   d0  f3       bne   $e94b       auf Reaktion des Computers warten
e958   a9  08       lda   #$08         Zähler auf 8 Bits vom seriellen Bus
e95a   85  98       sta   $98         setzen (Bereitschaft, um ein Byte zu holen)
e95c   20  c0  e9    jsr   $e9c0       Port zum seriellen Bus lesen
e95f   29  01       and   #$01         Datenbit isolieren
e961   d0  36       bne   $e999       verzweige, wenn Bit gesetzt
e963   a6  82       ldx   $82         Kanalnummer
e965   bd  3e  02    lda   $023e,x     Datenregister aus Tabelle
e968   6a           ror               Bit vom Bus ins Datenregister schieben
e969   9d  3e  02    sta   $023e,x     und Registerinhalt wieder merken
e96c   b0  05       bcs   $e973       verzweige, wenn Datenbit gleich 1 war
e96e   20  a5  e9    jsr   $e9a5       DATA OUT-Leitung auf Lo setzen
e971   d0  03       bne   $e976       unbedingter Sprung
e973   20  9c  e9    jsr   $e99c       DATA OUT-Leitung auf Hi setzen
e976   20  b7  e9    jsr   $e9b7       CLOCK OUT-Leitung auf Hi setzen
e979   a5  23       lda   $23         auf Busverzögerung prüfen
```

```
e97b   d0  03       bne   $e980       verzweige, wenn schneller Busbetrieb
e97d   20  f3  fe    jsr   $fef3       sonst Verzögerung (für C64)
e980   20  fb  fe    jsr   $fefb       CLOCK OUT Lo und DATA OUT Hi setzen
e983   c6  98       dec   $98         Zähler für die Bits vom Bus vermindern
e985   d0  d5       bne   $e95c       verzweige, wenn noch keine Bits auszugeben sind
e987   20  59  ea    jsr   $ea59       auf ATN-Signal prüfen
e98a   20  c0  e9    jsr   $e9c0       auf Antwort vom Computer testen
e98d   29  01       and   #$01        Datenbit isolieren
e98f   f0  f6       beq   $e987       auf Reaktion vom Computer warten
e991   58           cli               Diskcontroller aktivieren
e992   20  aa  d3    jsr   $d3aa       nächstes Byte für Ausgabe bereitstelle,
e995   78           sei               Diskcontroller wieder inaktivieren
e996   4c  0f  e9    jmp   $e90f       und Byte über den Bus senden (TALK)
e999   4c  4e  ea    jmp   $ea4e       zurück zur Systemwarteschleife
------------------------------------------------------------------------------
e99c                                  DATA OUT-Leitung auf Hi setzen.
e99c   ad  00  18    lda   $1800       Port lesen
e99f   29  fd       and   #$fd        DATA OUT Bit löschen (entspricht Leitung Hi)
e9a1   8d  00  18    sta   $1800       neuen Zustand auf Bus ausgeben
e9a4   60           rts               Ende
------------------------------------------------------------------------------
e9a5                                  DATA OUT-Leitung auf Lo setzen.
e9a5   ad  00  18    lda   $1800       Port lesen
e9a8   09  02       ora   #$02        DATA OUT Bit setzen (entspricht Leitung Lo)
e9aa   8d  00  18    sta   $1800       neuen Zustand auf Bus ausgeben
e9ad   60           rts               Ende
------------------------------------------------------------------------------
e9ae                                  CLOCK OUT-Leitung auf Lo setzen.
e9ae   ad  00  18    lda   $1800       Port lesen
e9b1   09  08       ora   #$08        CLOCK OUT Bit setzen (entspricht Leitung Lo)
e9b3   8d  00  18    sta   $1800       neuen Zustand auf Bus ausgeben
e9b6   60           rts               Ende
------------------------------------------------------------------------------
e9b7                                  CLOCK OUT-Leitung auf Hi setzen.
e9b7   ad  00  18    lda   $1800       Port lesen
e9ba   29  f7       and   #$f7        CLOCK OUT Bit löschen (entspricht Leitung Hi)
e9bc   8d  00  18    sta   $1800       neuen Zustand auf Bus ausgeben
e9bf   60           rts               Ende
------------------------------------------------------------------------------
e9c0                                  Wartet auf Antwortsignal vom Bus.
e9c0   ad  00  18    lda   $1800       Port lesen
e9c3   cd  00  18    cmp   $1800       konstanter Wert?
e9c6   d0  f8       bne   $e9c0       warten, bis ein konstanter Wert anliegt
e9c8   60           rts               Ende
------------------------------------------------------------------------------
e9c9                                  Routine holt ein Datenbyte vom Bus als Folge eines LISTEN-
                                      Kommandos vom Computer.
e9c9   a9  08       lda   #$08        Zähler für die Übertragung von 8 Bits
e9cb   85  98       sta   $98         setzen
e9cd   20  59  ea    jsr   $ea59       auf ATN-Signal prüfen
e9d0   20  c0  e9    jsr   $e9c0       CLOCK IN-Leitung prüfen
e9d3   29  04       and   #$04        CLOCK IN-Bit prüfen
e9d5   d0  f6       bne   $e9cd       warten, bis Bit gleich 0 wird
e9d7   20  9c  e9    jsr   $e99c       DATA OUT-Leitung auf Hi setzen
e9da   a9  01       lda   #$01        255 Mikrosekunden Verzögerung
e9dc   4c  20  ff    jmp   $ff20       durch Tirner setzen
e9df   20  59  ea    jsr   $ea59       auf ATN-Signal prüfen
e9e2   ad  0d  18    lda   $180d       Timerwert prüfen
```

```
e9e5   29  40       and   #$40        Abfrage, ob EOI gesendet wurde
e9e7   d0  09       bne   $e9f2       verzweige, wenn ja
e9e9   20  c0  e9   jsr   $e9c0       CLOCK IN-Leitung prüfen
e9ec   29  04       and   #$04        CLOCK IN-Bit prüfen
e9ee   f0  ef       beq   $e9df       warten, bis Bit gleich 1 wird
e9f0   d0  19       bne   $ea0b       unbedingter Sprung
e9f2   20  a5  e9   jsr   $e9a5       DATA OUT auf Lo setzen; Antwortmeldung
e9f5   a2  0a       ldx   #$0a        Verzögerung für Computer
e9f7   ca           dex               von ca. 50 Mikrosekunden
e9f8   d0  fd       bne   $e9f7       ausführen
e9fa   20  9c  e9   jsr   $e99c       DATA OUT Hi setzen
e9fd   20  59  ea   jsr   $ea59       auf ATN-Signal prüfen
ea00   20  c0  e9   jsr   $e9c0       CLOCK IN-Leitung prüfen
ea03   29  04       and   #$04        CLOCK IN-Bit isolieren
ea05   f0  f6       beq   $e9fd       warten, bis Bit gleich 1 wird
ea07   a9  00       lda   #$00        EOI-Signal anzeigen
ea09   85  f8       sta   $f8         Flag setzen
ea0b   ad  00  18   lda   $1800       Port vom seriellen Bus lesen
ea0e   49  01       eor   #$01        Datenbit invertieren
ea10   4a           lsr               und ins Carry schieben
ea11   29  02       and   #$02        vorheriges CLOCK IN-Bit auf Daten testen
ea13   d0  f6       bne   $ea0b       noch einmal, wenn Bit ungültig
ea15   ea           nop               Speicherbereich, der
ea16   ea           nop               durch Modifizierung des ROM
ea17   ea           nop               in der 1541 entstand
ea18   66  85       ror   $85         Datenbit in Register schieben
ea1a   20  59  ea   jsr   $ea59       auf ATN-Signal prüfen
ea1d   20  c0  e9   jsr   $e9c0       CLOCK IN-Leitung prüfen
ea20   29  04       and   #$04        CLOCK IN-Bit isolieren
ea22   f0  f6       beq   $ea1a       warten, bis Bit gleich 1 wird
ea24   c6  98       dec   $98         Zähler für die 8 Bits vermindern
ea26   d0  e3       bne   $ea0b       weitermachen, wenn noch Bits zu holen sind
ea28   20  a5  e9   jsr   $e9a5       DATA OUT-Leitung Lo setzen; Antwortsignal
ea2b   a5  85       lda   $85         Datenbyte aus Zwischenspeicher holen
ea2d   60           rts               Ende
--------------------------------------------------------------------------
ea2e                                  Holen von Daten über den seriellen Bus und Schreiben dieser
                                      Daten in den aktuellen Puffer.
ea2e   78           sei               Diskcontroller inaktivieren
ea2f   20  07  d1   jsr   $d107       freien Schreibkanal suchen
ea32   b0  05       bcs   $ea39       verzweige, wenn kein Kanal frei
ea34   b5  f2       lda   $f2,x       Kanalstatus holen
ea36   6a           ror               testen, ob Kanal inaktiv
ea37   b0  0b       bcs   $ea44       verzweige, wenn Kanal nicht aktiv
ea39   a5  84       lda   $84         Befehlssekundäradresse
ea3b   29  f0       and   #$f0        Kommando isolieren
ea3d   c9  f0       cmp   #$f0        OPEN-Befehl?
ea3f   f0  03       beq   $ea44       verzweige, wenn ja
ea41   4c  4e  ea   jmp   $ea4e       zurück zur Systemwarteschleife
ea44   20  c9  e9   jsr   $e9c9       Datenbyte vom Bus holen
ea47   58           cli               Diskcontroller wieder aktivieren
ea48   20  b7  cf   jsr   $cfb7       und Datenbyte in aktiven Puffer schreiben
ea4b   4c  2e  ea   jmp   $ea2e       weitermachen; nächstes Byte holen
ea4e   a9  00       lda   #$00        Port für weiteren Signalempfang
ea50   8d  00  18   sta   $1800       löschen
ea53   4c  6b  83   jmp   $836b       zur erweiterten Behandlung für 1571-Betrieb
ea56   4c  5b  e8   jmp   $e85b       zur Busbedienung bei ATN vom Computer
```

```
------------------------------------------------------------------------
ea59                                    Testet auf ATN-Modus.
ea59    a5  7d      lda   $7d           Flag für ATN-Modus gesetzt?
ea5b    f0  06      beq   $ea63         Sea63 verzweige, wenn nein (kein ATN)
ea5d    ad  00  18  lda   $1800         Port zum seriellen Bus lesen
ea60    10  09      bpl   $ea6b         verzweige, wenn kein ATN anliegt
ea62    60          rts                 Ende
ea63    ad  00  18  lda   $1800         Port zum seriellen Bus lesen
ea66    10  fa      bpl   $ea62         verzweige, wenn kein ATN mehr
ea68    4c  b3  a7  jmp   $a7b3         zur 1571 oder 1541-Busbedienung (je nach Modus)
ea6b    4c  ac  a9  jmp   $a9ac         zur 1571 oder 1541.Busbedienung (je nach Modus)
------------------------------------------------------------------------
ea6e                                    Routine zur Behandlung von Hardwaredefekten oder Fehler
                                        beim Selbsttest. Wird ab $ea71 eingesprungen, so erfolgt
                                        die Blinkanzeige für einen Hardwaredefekt. Bei Einsprung ab
                                        $ea6e wird ein Prüfsummenfehler im ROM oder ein defektes
                                        RAM angezeigt. In allen Fällen blinkt die LED des Laufwerks
                                        und die Floppy wird gegen weiteren Betrieb durch
                                        Verriegelung abgesichert, bis der Schaden behoben worden
                                        ist.
ea6e    a2  00      ldx   #$00         Wert 0 für Hardwaredefekt
ea70    2c          .byte $2c          nächsten Befehl überspringen

ea71    a6  6f      ldx   $6f          Fehlernummer für RAM oder ROM-Fehler
ea73    9a          txs                Wert merken
ea74    ba          tsx                X-Inhalt zurückholen
ea75    a9  08      lda   #$08         LED-Bit setzen
ea77    0d  00  1c  ora   $1c00        und die LED am Laufwerk
ea7a    4c  ea  fe  jmp   $feea        einschalten; Rückkehr mit JMP
ea7d    98          tya                Verzögerung
ea7e    18          clc                für Blinkbetrieb
ea7f    69  01      adc   #$01         durch Additionen
ea81    d0  fc      bne   $ea7f        setzen; verzweige, wenn ungleich Null
ea83    88          dey                Verzögerungszähler Hi vermindern
ea84    d0  f8      bne   $ea7e        und weitermachen
ea86    ad  00  1c  lda   $1c00        LED-Status holen
ea89    29  f7      and   #$f7         LED-Bit löschen und
ea8b    8d  00  1c  sta   $1c00        LED dadurch ausschalten
ea8e    98          tya                Verzögerung
ea8f    18          clc                für Blinkbetrieb
ea90    69  01      adc   #$01         durch Additionen
ea92    d0  fc      bne   $ea90        setzen; verzweige, wenn ungleich Null
ea94    88          dey                Verzögerungszähler Hi vermindern
ea95    d0  f8      bne   $ea8f        und weitermachen
ea97    ca          dex                Fehlernummer ebenfalls vermindern
ea98    10  db      bpl   $ea75        durch fehlerspezifischen Blinkrhythmus
ea9a    e0  fc      cpx   #$fc         der LED kann der Fehler optisch
ea9c    d0  f0      bne   $ea8e        angezeigt und analysiert
ea9e    f0  d4      beq   $ea74        werden, da Floppybetrieb nicht mehr möglich
------------------------------------------------------------------------
eaa0                                    RESET-Routine der Floppystation. Beim ersten Einschalten
                                        befindet sich die Floppy grundsätzlich im 1541-Modus und
                                        muß erst vom Computer in den 1571-Modus gebracht werden.
eaa0    78          sei                Diskcontroller inaktivieren
eaa1    d8          cld                Prozessor auf hexadezimale Rechnung stellen
```

```
eaa2   a2  66       ldx   #$66        Wert für DDRA des Buscontrollers
eaa4   4c  10  ff   jmp   $ff10       VIAs setzen; Zurück mit jmp $eaa7
eaa7   e8           inx              X = $67
eaa8   a0  00       ldy   #$00        Zähler für RAM-Test
eaaa   a2  00       ldx   #$00        vorbereiten
eaac   8a           txa              Füllinhalt
eaad   95  00       sta   $00,x       Zeropage löschen
eaaf   e8           inx              nächste Speicherstelle
eab0   d0  fa       bne   $eaac       bis alles gelöscht
eab2   8a           txa              Testwert für Speicherstellen
eab3   d5  00       cmp   $00,x       mit Speicherstelle vergleichen
eab5   d0  b7       bne   $ea6e       Fehler, wenn ungleich
eab7   f6  00       inc   $00,x       Wert in Zeropageadresse erhöhen
eab9   c8           iny              Zähler für Testwert erhöhen
eaba   d0  fb       bne   $eab7       alle Werte durchprobieren
eabc   d5  00       cmp   $00,x       immer noch in Ordnung?
eabe   d0  ae       bne   $ea6e       verzweige zur Fehleranzeige, wenn nein
eac0   94  00       sty   $00,x       Zähler merken
eac2   b5  00       lda   $00,x       Zeropageadresse testen
eac4   d0  a8       bne   $ea6e       verzweige, wenn Fehler
eac6   e8           inx              nächste Adresse testen
eac7   d0  e9       bne   $eab2       bis die gesamte Zeropage getestet wurde
eac9   e6  6f       inc   $6f         Zeiger für RAM/ROM-Test erhöhen
eacb   a2  80       ldx   #$80        128 Pages für Test
eacd   86  76       stx   $76         ROM-Startadresse Hi setzen
eacf   a9  00       lda   #$00        Startadresse Lo des ROM-Bereichs
ead1   85  75       sta   $75         ebenfalls setzen
ead3   a0  02       ldy   #$02        Index auf 2 (erste zwei Bytes im ROM weglassen)
ead5   18           clc              da diese die Prüfsumme beinhalten
ead6   e6  76       inc   $76         Zeiger Hi plus 1
ead8   71  75       adc   ($75),y     Speicherstelle addieren
eada   c8           iny              nächstes Byte
eadb   d0  fb       bne   $ead8       weitermachen, wenn noch nicht alle addiert
eadd   ca           dex              nächste Page
eade   d0  f6       bne   $ead6       weitermachen, wenn noch nicht alle addiert
eae0   69  ff       adc   #$ff        Prüfsumme mit Obertrag ausgleichen
eae2   85  76       sta   $76         und merken
eae4   d0  39       bne   $eb1f       verzweige, wenn ungleich Null
eae6   ea           nop              Speicherplatz, der
eae7   ea           nop              durch Modifikation des 1541-ROM
eae8   ea           nop              übriggeblieben
eae9   ea           nop              ist
eaea   a9  01       lda   #$01        Adresse für RAM-Test (Page 1)
eaec   85  76       sta   $76         Adresse Hi setzen
eaee   e6  6f       inc   $6f         Testzeiger plus 1
eaf0   a2  07       ldx   #$07        7 Pages sind zu testen (bis $07ff)
eaf2   98           tya              Testwert nach A
eaf3   18           clc              und
eaf4   65  76       adc   $76         Prüfwert für Speicher daraus bilden
eaf6   91  75       sta   ($75),y     Speicherstelle testen
eaf8   c8           iny              nächste Speicheradresse
eaf9   d0  f7       bne   $eaf2       eine Page mit Testwerten füllen
eafb   e6  76       inc   $76         Zeiger Hi auf nächste Page
eafd   ca           dex              Pagezähler minus 1
eafe   d0  f2       bne   $eaf2       weitermachen, wenn noch Pages zu testen sind
eb00   a2  07       ldx   #$07        wieder 7 Pages
eb02   c6  76       dec   $76         diesmal von oben nach unten
eb04   88           dey              Index auf Speicherstelle einer Page
```

```
eb05   98          tya             wieder den Prüfwert
eb06   18          clc             bilden
eb07   65  76       adc   $76       und
eb09   d1  75       cmp   ($75),y   Speicherstelle mit Prüfwert vergleichen
eb0b   d0  12       bne   $eb1f     zur Fehleranzeige, wenn ungleich
eb0d   49  ff       eor   #$ff      sonst Prufwert invertieren
eb0f   91  75       sta   ($75),y   und in Speicherstelle
eb11   51  75       eor   ($75),y   Speicherstelle invertieren
eb13   91  75       sta   ($75),y   und noch einmal abspeichern (wieder alter Wert)
eb15   d0  08       bne   $eb1f     verzweige, wenn Fehler
eb17   98          tya             nächster Prüfwert
eb18   d0  ea       bne   $eb04     und weitermachen
eb1a   ca          dex             nächste Page testen
eb1b   d0  e5       bne   $eb02     weitermachen bis zur Zeropage
eb1d   f0  03       beq   $eb22     unbedingter Sprung
eb1f   4c  71  ea   jmp   $ea71     Sprung zum Fehlerblinken für Hardwaredefekte
eb22   4c  c0  a7   jmp   $a7c0     Stackpointer setzen; zurück mit jmp $eb25
eb25   ad  00  1c   lda   $1c00     Port für Diskcontroller lesen
eb28   29  f7       and   #$f7      LED-Bit löschen
eb2a   8d  00  1c   sta   $1c00     und LED damit ausschalten
eb2d   a9  03       lda   #$03      Interrupte;ngänge der VIA triggern
eb2f   8d  0c  18   sta   $180c     ATN IN neg. und WRITE PROTECT auf pos. Flanke
eb32   a9  82       lda   #$82      IRQ-Anzeige im IFR
eb34   8d  0d  18   sta   $180d     setzen
eb37   8d  0e  18   sta   $180e     IRQs im IER ermöglichen
eb3a   ad  00  18   lda   $1800     Port des Buscontrollers lesen
eb3d   29  60       and   #$60      Bits 5 und 6 für Gerätenummer
eb3f   0a          asl             isolieren und
eb40   2a          rol             nach Bitposition
eb41   2a          rol             0 und 1 schieben, um
eb42   2a          rol             die ordnungsgemäße Geräteadresse
eb43   09  48       ora   #$48      für TALK zu erhalten
eb45   85  78       sta   $78       Nummer für TALK merken
eb47   49  60       eor   #$60      Nummer für LISTEN herstellen
eb49   85  77       sta   $77       und ebenfalls abspeichern
eb4b   a2  00       ldx   #$00      Indexregister
eb4d   a0  00       ldy   #$00      initialisieren, um
eb4f   a9  00       lda   #$00      die Lo-Bytes der Pufferadressen
eb51   95  99       sta   $99,x     zu setzen
eb53   e8          inx             nächste Speicherstelle
eb54   b9  e0  fe   lda   $fee0,y   Pufferadresse Hi
eb57   95  99       sta   $99,x     setzen
eb59   e8          inx             nächste Speicherstelle
eb5a   c8          iny             Zeiger für Pufferadresse
eb5b   c0  05       cpy   #$05      schon alle 5 Adressen gesetzt?
eb5d   d0  f0       bne   $eb4f     weitermachen, wenn nein
eb5f   a9  00       lda   #$00      sonst Lo-Byte des INPUT-Puffers
eb61   95  99       sta   $99,x     setzen
eb63   e8          inx             und
eb64   a9  02       lda   #$02      Hi-Byte des INPUT-Puffers (gibt Adresse $0200)
eb66   95  99       sta   $99,x     ebenfalls setzen
eb68   e8          inx             nächste Speicherstelle:
eb69   a9  d5       lda   #$d5      Lo-Byte des ERROR-Puffers
eb6b   95  99       sta   $99,x     setzen
eb6d   e8          inx             und
eb6e   a9  02       lda   #$02      Hi-Byte (ergibt Adresse S02d5)
eb70   95  99       sta   $99,x     ebenfalls setzen
eb72   a9  ff       lda   #$ff      $ff für Kanalstatus (heißt: Kanal nicht belegt)
```

```
eb74   a2  12       ldx   #$12        13 Speicherstellen
eb76   9d  2b  02    sta   $022b,x     Kanalstatus setzen
eb79   ca            dex               nächste Kanalnummer
eb7a   10  fa        bpl   $eb76       und weitermachen
eb7c   a2  05        ldx   #$05        5 Puffer
eb7e   95  a7        sta   $a7,x       als frei ($ff heißt 'Puffer frei')
eb80   95  ae        sta   $ae,x       deklarieren und diesen Status
eb82   95  cd        sta   $cd,x       in alle Puffertabellen eintragen
eb84   ca            dex               nächste Puffernummer
eb85   10  f7        bpl   $eb7e       und weitermachen
eb87   a9  05        lda   #$05        Puffer Nummer 5 (INPUT-Puffer)
eb89   85  ab        sta   $ab         dem Kanal 4 zuordnen
eb8b   a9  06        lda   #$06        Puffer Nummer 6 (Fehlerpuffer)
eb8d   85  ac        sta   $ac         dem Kanal 5 zuordnen
eb8f   a9  ff        lda   #$ff        Puffer 7 existiert nicht und ist
eb91   85  ad        sta   $ad         demzufolge
eb93   85  b4        sta   $b4         unbenutzt und inaktiv
eb95   a9  05        lda   #$05        Kanal für Fehlerpuffer
eb97   8d  3b  02    sta   $023b       wird zum Lesen benutzt
eb9a   a9  84        lda   #$84        Kanal für INPUT-Puffer ist
eb9c   8d  3a  02    sta   $023a       generell ein Schreibkanal
eb9f   a9  0f        lda   #$0f        die Kanäle 0 bis 3
eba1   8d  56  02    sta   $0256       dienen dem allgemeinen Betrieb
eba4   a9  01        lda   #$01        Flag für 'READY TO LISTEN'
eba6   85  f6        sta   $f6         setzen
eba8   a9  88        lda   #$88        Flag für 'READY TO TALK'
ebaa   85  f7        sta   $f7         setzen
ebac   a9  e0        lda   #$e0        Puffer 0 bis 4 (Bitmuster)
ebae   8d  4f  02    sta   $024f       freigeben
ebb1   a9  ff        lda   #$ff        alle restlichen Puffer
ebb3   8d  50  02    sta   $0250       sind belegt
ebb6   a9  01        lda   #$01        Flags für
ebb8   85  1c        sta   $1c         Drivestatus und WRITE PROTECT
ebba   85  1d        sta   $1d         setzen
ebbc   20  63  cb    jsr   $cb63       Sprungtabelle für USER-Befehle initialisieren
ebbf   20  fa  ce    jsr   $cefa       Kanaltabelle initialisieren
ebc2   20  82  ff    jsr   $ff82       Diskcontroller für 1541 und 1571 setzen
ebc5   a9  22        lda   #$22        Zeiger für NMI
ebc7   85  65        sta   $65         auf
ebc9   a9  eb        lda   #$eb        die Adresse $EB22
ebcb   85  66        sta   $66         setzen
ebcd   a9  06        lda   #$06        Sektorabstand beim Schreiben auf Diskette
ebcf   85  69        sta   $69         (engl. 'sector interleave') auf 6 setzen
ebd1   a9  05        lda   #$05        Anzahl der Leseversuche (engl. 'retries')
ebd3   85  6a        sta   $6a         setzen
ebd5   a9  73        lda   #$73        Nummer der (Fehler-)Meldung
ebd7   20  c1  e6    jsr   $e6c1       '73, CBM DOS V3.0 1571' ausgeben
ebda   a9  00        lda   #$00        seriellen Bus
ebdc   8d  00  18    sta   $1800       initialisieren
ebdf   a9  1a        lda   #$1a        Datenrichtungsregister für seriellen Bus
ebe1   8d  02  18    sta   $1802       setzen
ebe4   20  86  a7    jsr   $a786       Peripherie für 1571 (CIA und WD1770) setzen
```

```
--------------------------------------------------------------------------
ebe7                                    Eingang der Systemwarteschleife. Dieser Programmabschnitt
                                        wird solange durchlaufen, bis ein gültiger Befehl über den
                                        Bus kommt. Ansonsten befindet sich die Floppy in einer Art
                                        von Wartezustand, der nur durch ein ATN unterbrochen wird.
ebe7    58          cli                 Diskcontroller wieder aktivieren
ebe8    ad  00  18  lda   $1800         CLOCK, DATA und ATN ACK-Leitungen auf Hi
ebeb    29  e5      and   #$e5          setzen, um den Bus damit in einen
ebed    8d  00  18  sta   $1800         definierten Zustand zu bringen
ebf0    ad  55  02  lda   $0255         Liegt ein Befehl vom Computer an?
ebf3    f0  0a      beq   $ebff         verzweige, wenn nein
ebf5    a9  00      lda   #$00          sonst wird das Kommandoflag gelöscht, zum
ebf7    8d  55  02  sta   $0255         Zeichen, daß der Befehl erkannt wurde
ebfa    85  67      sta   $67           NMI-Zustand löschen (ohne Belang!!!)
ebfc    4c  1c  a6  jmp   $a61c         Befehl analysieren und ggf. ausführen
ebff    58          cli                 Diskcontroller wieder aktivieren
ec00    a5  7c      lda   $7c           ATN vom Bus eingetroffen?
ec02    f0  03      beq   $ec07         verzweige, wenn nein
ec04    4c  b3  a7  jmp   $a7b3         zur Bedienung bei aufgetretenem ATN
```

```
------------------------------------------------------------------------
ec07   58          cli             Diskcontroller wieder aktivieren
ec08   a9  0e      lda   #$0e       maximale Sekundäradresse (14) für Files
ec0a   85  72      sta   $72        setzen (15 ist ja der Kommandokanal)
ec0c   a9  00      lda   #$00       Zähler für anliegende Jobs
ec0e   85  6f      sta   $6f        für Drive 0 und
ec10   85  70      sta   $70        für Drive 1 löschen
ec12   a6  72      ldx   $72        Sekundäradresse für Files
ec14   bd  2b  02  lda   $022b,x    entsprechenden Kanal auf aktiv
ec17   c9  ff      cmp   #$ff       prüfen
ec19   f0  10      beq   $ec2b      verzweige, wenn Kanal inaktiv
ec1b   29  3f      and   #$3f       sonst Kanalnummer isolieren
ec1d   85  82      sta   $82        und setzen
ec1f   20  93  df  jsr   $df93      zugehörige Puffernummer holen
ec22   aa          tax             und als Index
ec23   bd  5b  02  lda   $025b,x    Drivenummer für entsprechenden Puffer
ec26   29  01      and   #$01       isolieren
ec28   aa          tax             und als Index
ec29   f6  6f      inc   $6f,x      Zähler für die Jobs erhöhen
ec2b   c6  72      dec   $72        nächste Sekundäradresse
ec2d   10  e3      bpl   $ec12      untersuchen, falls noch nicht fertig
ec2f   a0  04      ldy   #$04       Index für Puffer
ec31   b9  00  00  lda   $0000,y    Jobspeicher prüfen
ec34   10  05      bpl   $ec3b      verzweige, wenn kein Job anliegt
ec36   29  01      and   #$01       sonst Drivenummer isolieren
ec38   aa          tax             und als Index nehmen
ec39   f6  6f      inc   $6f,x      entsprechenden Jobzähler erhöhen
ec3b   88          dey             nächsten Puffer untersuchen
ec3c   10  f3      bpl   $ec31      bis auch Puffer 0 geprüft wurde
ec3e   78          sei             Diskcontroller inaktivieren
ec3f   ad  00  1c  lda   $1c00      Steuerport des Diskcontrollers
ec42   29  f7      and   #$f7       LED-Bit löschen
ec44   48          pha             sonstigen Inhalt merken
ec45   a5  7f      lda   $7f        aktuelle Drivenummer
ec47   85  86      sta   $86        merken
ec49   a9  00      lda   #$00       Drive 0 als neue Nummer
ec4b   85  7f      sta   $7f        setzen
ec4d   a5  6f      lda   $6f        Jobzähler prüfen; liegen Jobs an?
ec4f   f0  0b      beq   $ec5c      verzweige, wenn nein
ec51   a5  1c      lda   $1c        wurde die Diskette gewechselt?
ec53   f0  03      beq   $ec58      verzweige, wenn nein
ec55   20  13  d3  jsr   $d313      sonst alle Kanäle für Drive 0 schließen
ec58   68          pla             Wert für Steuerport zurückholen
ec59   09  08      ora   #$08       und LED-Bit setzen
ec5b   48          pha             Wert wieder merken
ec5c   e6  7f      inc   $7f        Drive 1 setzen
ec5e   a5  70      lda   $70        liegen Jobs für Drive 1 an?
ec60   f0  0b      beq   $ec6d      verzweige, wenn nein
ec62   a5  1d      lda   $1d        wurde die Diskette gewechselt?
ec64   f0  03      beq   $ec69      verzweige, wenn nein
ec66   20  13  d3  jsr   $d313      alle Kanäle für Drive 1 schließen
ec69   68          pla             Wert für Steuerport zurückholen
ec6a   09  00      ora   #$00       (LED-Bit für Drive 1 setzen) n.v.!!!
ec6c   48          pha             Wert wieder merken
ec6d   a5  86      lda   $86        Drivenummer für aktuelles Drive zurückholen
ec6f   85  7f      sta   $7f        und wieder setzen
ec71   68          pla             Wert für Steuerport zurückholen
ec72   ae  6c  02  ldx   $026c      Fehlerflag prüfen
```

```
ec75   f0  21       beq   $ec98       verzweige, wenn kein Fehler
ec77   ad  00  1c   lda   $1c00       sonst Steuerport lesen
ec7a   e0  80       cpx   #$80        Fehlerflag auf neu erkannten Fehler testen
ec7c   d0  03       bne   $ec81       verzweige, wenn ein neuer Fehler erkannt wurde
ec7e   4c  8b  ec   jmp   $ec8b       sonst LED-Blinken des 'alten' Fehlers steuern
ec81   ae  05  18   ldx   $1805       Timer auslesen; ist die Blinkzeit abgelaufen?
ec84   30  12       bmi   $ec98       verzweige, wenn nein
ec86   a2  a0       ldx   #$a0        sonst den Timer
ec88   8e  05  18   stx   $1805       neu setzen
ec8b   ce  6c  02   dec   $026c       und den Fehlerzähler vermindern
ec8e   d0  08       bne   $ec98       verzweige, wenn Zähler nicht abgelaufen 1s:
ec90   4d  6d  02   eor   $026d       sonst LED-Maske umdrehen
ec93   a2  10       ldx   #$10        und den Blinkzähler neu
ec95   8e  6c  02   stx   $026c       setzen
ec98   8d  00  1c   sta   $1c00       LED-Status neu setzen
ec9b   4c  ff  eb   jmp   $ebff       weiter in der Systemwarteschleife
-------------------------------------------------------------------------
ec9e                                  Laden und Aufbereiten des Directory.
ec9e   a9  00       lda   #$00        interne Sekundäradresse auf 0 (LOAD)
eca0   85  83       sta   $83         setzen
eca2   a9  01       lda   #$01        einen
eca4   20  e2  d1   jsr   $d1e2       Kanal suchen und Puffer belegen
eca7   a9  00       lda   #$00        Pufferzeiger auf Null
eca9   20  c8  d4   jsr   $d4c8       setzen
ecac   a6  82       ldx   $82         Kanalnummer
ecae   a9  00       lda   #$00        Endezeiger für Kanalnummer
ecb0   9d  44  02   sta   $0244,x     löschen
ecb3   20  93  df   jsr   $df93       Puffernummer für Kanal holen
ecb6   aa           tax               und als Index
ecb7   a5  7f       lda   $7f         aktuelle Drivenummer
ecb9   9d  5b  02   sta   $025b,x     in Tabelle schreiben
ecbc   a9  01       lda   #$01        $01 in den
ecbe   20  f1  cf   jsr   $cff1       Puffer schreiben (Startadresse Lo)
ecc1   a9  04       lda   #$04        $04 in den
ecc3   20  f1  cf   jsr   $cff1       Puffer schreiben (Adresse $0401 im Speicher)
ecc6   a9  01       lda   #$01        $01
ecc8   20  f1  cf   jsr   $cff1       zweimal in den Puffer
eccb   20  f1  cf   jsr   $cff1       schreiben (Dummy für BASIC-Interpreter)
ecce   ad  72  02   lda   $0272       Drivenummer für Directory
ecd1   20  f1  cf   jsr   $cff1       in den Puffer schreiben (1. Teil der Zeilennr.)
ecd4   a9  00       lda   #$00        zweiter Teil der Zeilennummer (für BASIC)
ecd6   20  f1  cf   jsr   $cff1       ebenfalls in Puffer (gibt '0' als Zeilennummer)
ecd9   20  59  ed   jsr   $ed59       Diskettennamen in Puffer schreiben
ecdc   20  93  df   jsr   $df93       Puffernummer holen
ecdf   0a           asl               mal 2
ece0   aa           tax               als Index
ece1   d6  99       dec   $99,x       Pufferzeiger
ece3   d6  99       dec   $99,x       minus 2
ece5   a9  00       lda   #$00        $00 als BASIC-Zeilenende in den Puffer
ece7   20  f1  cf   jsr   $cff1       schreiben (erste Directoryzeile fertig)
ecea   a9  01       lda   #$01        $01 wieder zweimal in den Puffer, damit der
ecec   20  f1  cf   jsr   $cff1       BASIC-Interpreter daraus die echten Linker
ecef   20  f1  cf   jsr   $cff1       zwischen den BASIC-Zeilennummern macht
ecf2   20  ce  c6   jsr   $c6ce       Directoryeintrag holen
ecf5   90  2c       bcc   $ed23       verzweige, wenn kein Eintrag mehr
ecf7   ad  72  02   lda   $0272       sonst Blockzahl Lo
ecfa   20  f1  cf   jsr   $cff1       in den Puffer
ecfd   ad  73  02   lda   $0273       Blockzahl Hi (des Fileeintrags)
```

```
ed00   20  f1  cf    jsr   $cff1       in den Puffer
ed03   20  59  ed    jsr   $ed59       Directoryeintrag dahinter schreiben
ed06   a9  00        lda   #$00        $00 als Zeilenendekennzeicher
ed08   20  f1  cf    jsr   $cff1       in den Puffer
ed0b   d0  dd        bne   $ecea       verzweige, wenn Puffer noch nicht voll
ed0d   20  93  df    jsr   $df93       Puffernummer holen
ed10   0a            asl               mal 2
ed11   aa            tax               als Index
ed12   a9  00        lda   #$00        Pufferzeiger auf Null
ed14   95  99        sta   $99,x       setzen
ed16   a9  88        lda   #$88        Flag für READY TO TALK setzen, damit der
ed18   a4  82        ldy   $82         Computer den Directoryteil abholt; Kanalnummer
ed1a   8d  54  02    sta   $0254       Flag als Status für den Directorykanal
ed1d   99  f2  00    sta   $00f2,y     und als Status allgemein setzen
ed20   a5  85        lda   $85         Datenbyte in Akku
ed22   60            rts               Ende
-------------------------------------------------------------------------
ed23                                   Abschluß des Directory 'BLOCKS FREE'-Angabe herstellen.
ed23   ad  72  02    lda   $0272       Blockzahl der Diskette Lo
ed26   20  f1  cf    jsr   $cff1       in Puffer schreiben
ed29   ad  73  02    lda   $0273       Blockzahl der Diskette Hi
ed2c   20  f1  cf    jsr   $cff1       in Puffer schreiben
ed2f   20  59  ed    jsr   $ed59       'BLOCKS FREE' dahinter in Puffer schreiben
ed32   20  93  df    jsr   $df93       Puffernummer holen
ed35   0a            asl               mal 2
ed36   aa            tax               als Index
ed37   d6  99        dec   $99,x       Pufferzeiger
ed39   d6  99        dec   $99,x       minus 2
ed3b   a9  00        lda   #$00        dreimal $00 als
ed3d   20  f1  cf    jsr   $cff1       Kennzeichen für BASIC-Programmende
ed40   20  f1  cf    jsr   $cff1       in den Puffer
ed43   20  f1  cf    jsr   $cff1       schreiben
ed46   20  93  df    jsr   $df93       Puffernummer holen
ed49   0a            asl               mal 2
ed4a   a8            tay               als Index
ed4b   b9  99  00    lda   $0099,y     Pufferzeiger Lo
ed4e   a6  82        ldx   $82         Kanalnummer
ed50   9d  44  02    sta   $0244,x     Pufferzeiger als Endezeiger in den Puffer
ed53   de  44  02    dec   $0244,x     Endezeiger minus 1 ist Anzahl der Bytes
ed56   4c  0d  ed    jmp   $ed0d       Ende; Abschluß des Directory
-------------------------------------------------------------------------
ed59                                   Directoryzeile in den Ausgabepuffer schreiben, damit die
                                       Ausgabe an den Computer erfolgen kann.
ed59   a0  00        ldy   #$00        Zeiger auf erstes Zeichen
ed5b   b9  b1  02    lda   $02b1,y     Zeichen aus dem Directorypuffer
ed5e   20  f1  cf    jsr   $cff1       in einen Programmpuffer schreiben
ed61   c8            iny               nächstes Zeichen nehmen
ed62   c0  1b        cpy   #$1b        und ausgeben; Ende schon erreicht?
ed64   d0  f5        bne   $ed5b       weitermachen, wenn nein
ed66   60            rts               sonst Ende
-------------------------------------------------------------------------
ed67                                   Byte aus Directory holen; ggf. nächsten Block von der
                                       Diskette nachladen.
ed67   20  37  d1    jsr   $d137       Byte aus Datei holen; ggf. Block nachladen
ed6a   f0  01        beq   $ed6d       verzweige, wenn Fileende erreicht
ed6c   60            rts               Ende
```

```
ed6d    85  85       sta   $85         Datenbyte merken
ed6f    a4  82       ldy   $82         Kanalnummer als Index
ed71    b9  44  02    lda   $0244,y     Endezeiger des Blocks holen
ed74    f0  08       beq   $ed7e       verzweige, wenn Null
ed76    a9  80       lda   #$80        Flag für EOI
ed78    99  f2  00    sta   $00f2,y     setzen
ed7b    a5  85       lda   $85         Datenbyte holen
ed7d    60           rts                Ende
ed7e    48           pha                Endezeiger (0) merken
ed7f    20  ea  ec    jsr   $ecea       Directoryzeile in Puffer schreiben
ed82    68           pla                Endezeiger zurückholen
ed83    60           rts                Ende
--------------------------------------------------------------------------
ed84                                    VALIDATE-Befehl (V-Befehl).
ed84    20  d1  c1    jsr   $c1d1       Drivenummer aus Befehlsstring holen
ed87    20  42  d0    jsr   $d042       Diskette initialisieren; BAM laden
ed8a    a9  40       lda   #$40        Flag für BAM 'dirty'
ed8c    8d  f9  02    sta   $02f9       setzen
*1 ed8f 20  c7  a7    jsr   Sa7c7       neue BAM im Puffer erzeugen
*0 ed8f 20  b7  ee    jsr   $eeb7       neue BAM im Puffer erzeugen
ed92    a9  00       lda   #$00        Flag für Suche nach gültigem Directoryeintrag
ed94    8d  92  02    sta   $0292       setzen
ed97    20  ac  c5    jsr   $c5ac       Eintrag in Directory suchen
ed9a    d0  3d       bne   $edd9       verzweige, wenn gefunden
ed9c    a9  00       lda   #$00        sonst Sektornummer 0
ed9e    85  81       sta   $81         für BAM setzen
eda0    ad  85  fe    lda   $fe85       und Tracknummer 18
eda3    85  80       sta   $80         ebenfalls für BAM setzen
eda5    20  e5  ed    jsr   $ede5       Directoryblöcke in der BAM belegen
eda8    a9  00       lda   #$00        BAM 'dirty' Flag
edaa    8d  f9  02    sta   $02f9       löschen
edad    20  ff  ee    jsr   $eeff       BAM auf Diskette schreiben
edb0    4c  94  c1    jmp   $c194       Diskstatus bereitstellen; Ende
--------------------------------------------------------------------------
edb3                                    Blöcke eines Files im Directory nacheinander durchgehen und
                                        auf Vorhandensein testen. Sind sie vorhanden, werden sie in
                                        der BAM belegt.
edb3    c8           iny                nächste Position
edb4    b1  94       lda   ($94),y     Tracknummer holen
edb6    48           pha                und merken
edb7    c8           iny                Zeiger auf Sektor
edb8    b1  94       lda   ($94),y     Sektornummer holen
edba    48           pha                und merken
edbb    a0  13       ldy   #$13        19; Zeiger auf Side-Sektor-Block
edbd    b1  94       lda   ($94),y     Track für Side-Sektor holen
edbf    f0  0a       beq   $edcb       verzweige, wenn kein Side-Sektor vorhanden
edc1    85  80       sta   $80         Tracknummer merken
edc3    c8           iny                Zeiger auf Sektornummer
edc4    b1  94       lda   ($94),y     Sektornummer des Side-Sektors holen
edc6    85  81       sta   $81         und merken
edc8    20  e5  ed    jsr   $ede5       Side-Sektor-Blöcke als belegt kennzeichnen
edcb    68           pla                Sektornummer des Files zurückholen
edcc    85  81       sta   $81         und wieder setzen
edce    68           pla                Tracknummer zurückholen
edcf    85  80       sta   $80         und ebenfalls setzen
edd1    20  e5  ed    jsr   $ede5       Blöcke des Files in der BAM belegen
edd4    20  04  c6    jsr   $c604       nächsten gültigen Fileeintrag holen
edd7    f0  c3       beq   $ed9c       verzweige, wenn kein Eintrag mehr vorhanden
```

```
edd9    a0  00        ldy   #$00        Zeiger auf Filetyp
eddb    b1  94        lda   ($94),y     Filetyp aus Puffer holen
eddd    30  d4        bmi   $edb3       verzweige, wenn File geschlossen
eddf    20  b6  c8    jsr   $c8b6       sonst File löschen (SCRATCH)
ede2    4c  d4  ed    jmp   $edd4       und weitermachen
--------------------------------------------------------------------------
ede5                                    File anhand der Linkbytes in den Sektoren nachverfolgen und
                                        Blöcke in der BAM belegen
ede5    20  5f  d5    jsr   $d55f       Track und Sektor prüfen
ede8    20  90  ef    jsr   $ef90       Block in der BAM als belegt kennzeichnen
edeb    20  75  d4    jsr   $d475       Kanal öffnen und Block lesen
edee    a9  00        lda   #$00        Pufferzeiger auf Null
edf0    20  c8  d4    jsr   $d4c8       setzen
edf3    20  37  d1    jsr   $d137       Byte aus Puffer holen
edf6    85  80        sta   $80         und als Tracknummer merken
edf8    20  37  d1    jsr   $d137       noch ein Byte aus Puffer holen
edfb    85  81        sta   $81         und als Sektornummer merken
edfd    a5  80        lda   $80         Tracknummer prüfen
edff    d0  03        bne   $ee04       verzweige, wenn noch ein Block folgt
ee01    4c  27  d2    jmp   $d227       letzter Block im Puffer; Kanal schließen; Ende
ee04    20  90  ef    jsr   $ef90       Block in der BAM belegen
ee07    20  4d  d4    jsr   $d44d       nächsten Block lesen
ee0a    4c  ee  ed    jmp   $edee       und weitermachen
--------------------------------------------------------------------------
ee0d                                    NEW-Befehl (N-Befehl).
ee0d    20  12  c3    jsr   $c312       Driveparameter setzen
ee10    a5  e2        lda   $e2         Drivenummer holen
ee12    10  05        bpl   $ee19       verzweige, wenn Drivenummer ok
ee14    a9  33        lda   #$33        Nummer der Fehlermeldung
ee16    4c  c8  c1    jmp   $c1c8       '33, SYNTAX ERROR' ausgeben
ee19    29  01        and   #$01        Drivenummer isolieren
ee1b    85  7f        sta   $7f         und setzen
ee1d    20  9c  ff    jsr   $ff9c       Drivestatus setzen und LED einschalten
ee20    a5  7f        lda   $7f         Drivenummer
ee22    0a            asl               mal 2
ee23    aa            tax               als Index
ee24    ac  7b  02    ldy   $027b       Position der ID im Befehlsstring
ee27    cc  74  02    cpy   $0274       wurde eine neue ID angegeben?
ee2a    f0  1a        beq   $ee46       verzweige, wenn nein
ee2c    b9  00  02    lda   $0200,y     erstes Zeichen der neuen ID
ee2f    95  12        sta   $12,x       setzen
ee31    b9  01  02    lda   $0201,y     zweites Zeichen der neuen ID
ee34    95  13        sta   $13,x       setzen
ee36    20  07  d3    jsr   $d307       alle Kanäle schließen
ee39    a9  01        lda   #$01        Track 1 als Starttrack für die Formatierung
ee3b    85  80        sta   $80         setzen
ee3d    20  2f  ff    jsr   $ff2f       auf Format prüfen; Diskette formatieren
*1 ee40 4c  64  a7    jmp   $a764       BAM-Puffer löschen; Diskettenseite setzen
*0 ee40 20  05  f0    jsr   $f005       BAM-Puffer löschen
ee43    4c  56  ee    jmp   $ee56       weiter...
ee46    20  42  d0    jsr   $d042       Diskette initialisieren
ee49    a6  7f        ldx   $7f         Drivenummer
ee4b    bd  01  01    lda   $0101,x     Formatkennzeichen holen
ee4e    cd  d5  fe    cmp   $fed5       mit 'A' vergleichen
ee51    f0  03        beq   $ee56       verzweige, wenn alles ok
ee53    4c  72  d5    jmp   $d572       '73, CBM DOS V3.0 1571' ausgeben
ee56    20  c7  a7    jsr   $a7c7       neue BAM erzeugen (ein- oder zweiseitig)
ee59    a5  f9        lda   $f9         Puffernummer
```

```
ee5b   a8          tay             als Index
ee5c   0a          asl             mal 2
ee5d   aa          tax             als Index
ee5e   ad  88  fe  lda  $fe88       Konstante 590; Position des Disknamens
ee61   95  99      sta  $99,x       setzen
ee63   ae  7a  02  ldx  $027a       Puffernummer holen
ee66   a9  1b      lda  #$1b       27; Länge des Disknamens insgesamt
ee68   20  6e  c6  jsr  $c66e       Diskname in BAM-Puffer schreiben
ee6b   a0  12      ldy  #$12       18 als Position der ID
ee6d   a6  7f      ldx  $7f        Drivenummer
ee6f   ad  d5  fe  lda  $fed5       ASCII-Code für 'A'; 1541/1571 Format
ee72   9d  01  01  sta  $0101,x     setzen
ee75   8a          txa             Drivenummer
ee76   0a          asl             mal 2
ee77   aa          tax             als Index
ee78   b5  12      lda  $12,x      erstes Zeichen der ID
ee7a   91  94      sta  ($94),y    in Puffer schreiben
ee7c   c8          iny             Index auf 19
ee7d   b5  13      lda  $13,x      zweites Zeichen der ID
ee7f   91  94      sta  ($94),y    ebenfalls in Puffer schreiben
ee81   c8          iny             Zeiger auf
ee82   c8          iny             21
ee83   a9  32      lda  #$32       ASCII-Code für '2'
ee85   91  94      sta  ($94),y    in Puffer
ee87   c8          iny             und
ee88   ad  d5  fe  lda  $fed5       ASCII-Code für 'A'
ee8b   91  94      sta  ($94),y    ebenfalls in Puffer
ee8d   a0  02      ldy  #$02       und
ee8f   91  6d      sta  ($6d),y    an Position 2 noch einmal in Puffer
ee91   ad  85  fe  lda  $fe85       18 (Tracknummer der BAM)
ee94   85  80      sta  $80        setzen
ee96   20  93  ef  jsr  $ef93       Block 18,0 in der BAM belegen
ee99   a9  01      lda  #$01       Sektornummer 1 (erster Directoryblock)
ee9b   85  81      sta  $81        setzen
ee9d   20  93  ef  jsr  $ef93       Block 18,1 in der BAM belegen
eea0   20  ff  ee  jsr  $eeff       neue BAM auf Diskette schreiben
eea3   20  05  f0  jsr  $f005       BAM-Puffer löschen
eea6   a0  01      ldy  #$01       ersten Directoryblock mit
eea8   a9  ff      lda  #$ff       $ff als Anzahl der Bytes
eeaa   91  6d      sta  ($6d),y    herstellen
eeac   20  64  d4  jsr  $d464       Block auch auf Diskette schreiben
eeaf   c6  81      dec  $81        Sektornummer auf 0
*1 eeb1 20  42  d0  jsr  $d042      Diskette initialisieren
*0 eeb1 20  60  d4  jsr  $d460      Block von Diskette lesen
eeb4   4c  94  c1  jmp  $c194       Diskstatus bereitstellen
-------------------------------------------------------------------------
eeb7                               Neue BAM erzeugen.
eeb7   20  d1  f0  jsr  $f0d1       BAM-Puffer löschen
eeba   a0  00      ldy  #$00       Index in Puffer setzen
eebc   a9  12      lda  #$12       18
eebe   91  6d      sta  ($6d),y    als Linkbyte für Directory in Puffer
eec0   c8          iny             und
eec1   98          tya             1
eec2   91  6d      sta  ($6d),y    als Sektor für Directory
eec4   c8          iny             Zeiger in Puffer
eec5   c8          iny             plus 3
eec6   c8          iny             auf Position 4 setzen
eec7   a9  00      lda  #$00       Startwert für Eintragungen
```

```
eec9    85  6f       sta   $6f         24 Bits für die Belegung
eecb    85  70       sta   $70         der Blöcke eines Tracks
eecd    85  71       sta   $71         reservieren
eecf    98           tya               Zeiger in BAM
eed0    4a           lsr               geteilt durch 4
eed1    4a           lsr               ergibt Tracknummer der Eintra,;
eed2    20  4b  f2   jsr   $f24b       maximale Anzahl der Sektoren des Tracks holen
eed5    91  6d       sta   ($6d),y     und in BAM-Puffer
eed7    c8           iny               Zeiger plus 1
eed8    aa           tax               Anzahl der Sektoren als Zähler setzen
eed9    38           sec               Bitmuster der
eeda    26  6f       rol   $6f         belegten Blöcke
eedc    26  70       rol   $70         eines Tracks
eede    26  71       rol   $71         erzeugen
eee0    ca           dex               nächster Sektor des Tracks
eee1    d0  f6       bne   $eed9       und weitermachen
eee3    b5  6f       lda   $6f,x       jetzt:
eee5    91  6d       sta   ($6d),y     die Blockbelegungen aus dem Zwischenspeicher
eee7    c8           iny               in die BAM
eee8    e8           inx               übertragen;
eee9    e0  03       cpx   #$03        insgesamt drei Bytes
eeeb    90  f6       bcc   $eee3       weitermachen
eeed    c0  90       cpy   #$90        schon Ende der BAM erreicht?
eeef    90  d6       bcc   $eec7       nächster Track, wenn nein
eef1    4c  75  d0   jmp   $d075       Anzahl der 'BLOCKS FREE' berechnen und eintragen
-------------------------------------------------------------------------------
eef4                                   BAM auf Diskette schreiben, wenn sie im Puffer geändert
                                       wurde (BAM 'dirty').
eef4    20  93  df   jsr   $df93       Puffernummer holen
eef7    aa           tax               und als Index
eef8    bd  5b  02   lda   $025b,x     Jobcode für Puffer holen
eefb    29  01       and   #$01        Drivenummer isolieren
eefd    85  7f       sta   $7f         und übernehmen
eeff    a4  7f       ldy   $7f         Drivenummer als Index
ef01    b9  51  02   lda   $0251,y     zugehöriges BAM 'dirty' Flag gesetzt?
ef04    d0  01       bne   $ef07       verzweige, wenn ja
ef06    60           rts               sonst Ende; BAM in Ordnung
ef07    a9  00       lda   #$00        'dirty' Flag
ef09    99  51  02   sta   $0251,y     löschen
ef0c    20  3a  ef   jsr   $ef3a       Pufferzeiger für BAM setzen
ef0f    a5  7f       lda   $7f         Drivenummer
ef11    0a           asl               mal 2
ef12    48           pha               merken
ef13    20  a5  f0   jsr   $f0a5       Eintragungen (Drive 0) holen
ef16    68           pla               Drivenummer mal 2 zurückholen
ef17    18           clc               und
ef18    69  01       adc   #$01        plus 1
ef1a    20  a5  f0   jsr   $f0a5       Eintragungen (Drive 1) holen
ef1d    a5  80       lda   $80         Tracknummer
ef1f    48           pha               merken
ef20    a9  01       lda   #$01        Track 1
ef22    85  80       sta   $80         setzen
ef24    0a           asl               1 mal 4
ef25    0a           asl               ergibt 4 als
ef26    85  6d       sta   $6d         Anzahl der Bytes pro Track in der BAM
*1 ef28 20  37  a9   jsr   $a937       Anzahl der 'BLOCKS FREE' in der BAM prüfen
*0 ef28 20  20  f2   jsr   $f220       Anzahl der 'BLOCKS FREE' in der BAM prüfen
ef2b    e6  80       inc   $80         Tracknummer plus 1
```

```
ef2d    a5  80       lda   $80         Tracknummer
*1 ef2f cd  ac  02   cmp   $02ac       schon Maximalwert (36,72) erreicht?
*0 ef2f cd  d7  fe   cmp   $fed7       schon Maximalwert (36) erreicht?
ef32    90  f0       bcc   $ef24       verzweige, wenn nein
ef34    68           pla               Tracknummer zurückholen
ef35    85  80       sta   $80         und wieder setzen
*1 ef37 4c  8d  a5   jmp   $a58d       BAM auf die Diskette schreiben; Ende
*0 ef37 4c  8a  d5   jmp   $d58a       BAM auf Diskette schreiben; Ende
-------------------------------------------------------------------------------
ef3a                                   BAM, falls notwendig, lesen und Zeiger auf BAM setzen.
ef3a    20  0f  f1   jsr   $f10f       Kanalnummer für BAM (6) holen
ef3d    aa           tax               als Index
ef3e    20  df  f0   jsr   $f0df       zugehörigen Puffer belegen
ef41    a6  f9       ldx   $f9         Puffernummer als Index
ef43    bd  e0  fe   lda   $fee0,x     Pufferadresse Hi holen
ef46    85  6e       sta   $6e         und setzen
ef48    a9  00       lda   #$00        Pufferadresse Lo (0)
ef4a    85  6d       sta   $6d         setzen
ef4c    60           rts               Ende
-------------------------------------------------------------------------------
ef4d                                   Anzahl der freien Blöcke auf Diskette aus $02fa und $02fc
                                       holen.
ef4d    a6  7f       ldx   $7f         aktuelle Drivenummer
ef4f    bd  fa  02   lda   $02fa,x     AnzahL der Blöcke Lo
ef52    8d  72  02   sta   $0272       übernehmen
ef55    bd  fc  02   lda   $02fc,x     Anzahl der Blöcke Hi
ef58    8d  73  02   sta   $0273       übernehmen
ef5b    60           rts               Ende
-------------------------------------------------------------------------------
ef5c                                   Block in der BAM freigeben.
ef5c    20  f1  ef   jsr   $eff1       BAM schreiben, wenn 'dirty'
*1 ef5f 4c  3e  a8   jmp   $a83e       Zeiger für 1541 holen; 1571 Block freigeben
*0 ef5f 20  cf  ef   jsr   $efcf       Zeiger in BAM holen
ef62    38           sec               Flag für Block frei setzen
ef63    d0  22       bne   $ef87       verzweige, wenn Block schon frei
ef65    b1  6d       lda   ($6d),y     Bitmuster für Block holen
ef67    1d  e9  ef   ora   $efe9,x     Block als frei kennzeichnen
ef6a    91  6d       sta   ($6d),y     und Bitmuster wieder abspeichern
ef6c    20  88  ef   jsr   $ef88       BAM 'dirty' Flag setzen
ef6f    a4  6f       ldy   $6f         Zeiger auf Zahl der BLOCKS FREE pro Track
ef71    18           clc               Addition vorbereiten
ef72    b1  6d       lda   ($6d),y     Anzahl der freien Blöcke pro Track
ef74    69  01       adc   #$01        plus 1
ef76    91  6d       sta   ($6d),y     und wieder abspeichern
ef78    a5  80       lda   $80         Tracknummer
ef7a    cd  85  fe   cmp   $fe85       mit 18 vergleichen (Directorytrack)
ef7d    f0  3b       beq   $efba       übergehen, wenn Track 18
ef7f    fe  fa  02   inc   $02fa,x     Anzahl der BLOCKS FREE erhöhen
ef82    d0  03       bne   $ef87       verzweige, wenn kein Überlauf
ef84    fe  fc  02   inc   $02fc,x     sonst auch Zahl Hi erhöhen
ef87    60           rts               Ende
-------------------------------------------------------------------------------
ef88                                   Flag für BAM geändert ('dirty' Flag) setzen.
ef88    a6  7f       ldx   $7f         aktuelle Drivenummer
ef8a    a9  01       lda   #$01        'dirty' Flag
ef8c    9d  51  02   sta   $0251,x     setzen
ef8f    60           rts               Ende
```

```
------------------------------------------------------------------------
ef90                                    Block in der BAM als belegt kennzeichnen.
ef90    20  f1  ef    jsr   $eff1       BAM schreiben, wenn 'dirty'
*1 ef93 4c  74  a8    jmp   $a874       Zeiger holen (1541); Block belegen (1571)
*0 ef93 20  cf  ef    jsr   $efcf       Zeiger in BAM holen
ef96    f0  36        beq   $efce       verzweige, wenn schon belegt
ef98    b1  6d        lda   ($6d),y     Byte mit Bitmuster des Blocks holen
ef9a    5d  e9  ef    eor   $efe9,x     Bit des Blocks löschen (Block belegt)
ef9d    91  6d        sta   ($6d),y     Bitmuster wieder abspeichern
ef9f    20  88  ef    jsr   $ef88       BAM 'dirty' Flag setzen
efa2    a4  6f        ldy   $6f         Zeiger auf Anzahl der BLOCKS FREE pro Track
efa4    b1  6d        lda   ($6d),y     Anzahl der freien Blöcke des Tracks holen
efa6    38            sec               und
efa7    e9  01        sbc   #$01        einen Block abziehen
efa9    91  6d        sta   ($6d),y     Wert wieder abspeichern
efab    a5  80        lda   $80         Tracknummer
efad    cd  85  fe    cmp   $fe85       gleich 18 (Directorytrack)?
efb0    f0  0b        beq   $efbd       übergehen, wenn ja
efb2    bd  fa  02    lda   $02fa,x     sonst Anzahl der BLOCKS FREE Lo
efb5    d0  03        bne   $efba       verzweige, wenn ungleich 0
efb7    de  fc  02    dec   $02fc,x     Anzahl Hi minus 1
efba    de  fa  02    dec   $02fa,x     Anzahl Lo minus 1
efbd    bd  fc  02    lda   $02fc,x     Anzahl der BLOCKS FREE Hi
efc0    d0  0c        bne   $efce       verzweige, ungleich Null
efc2    bd  fa  02    lda   $02fa,x     Anzahl der BLOCKS FREE Lo
efc5    c9  03        cmp   #$03        kleiner 3?
efc7    b0  05        bcs   $efce       verzweige, wenn nein
efc9    a9  72        lda   #$72        Nummer der Fehlermeldung
efcb    20  c7  e6    jsr   $e6c7       '72, DISK FULL' ausgeben
efce    60            rts               Ende
------------------------------------------------------------------------
efcf                                    Berechnet den Index in die BAM, der für den entsprechenden
                                        Block zuständig ist. Bei der Rückkehr zeigt das Zero-Flag
                                        den Zustand des gewünschten Bits an: 1 = Block belegt 0 =
                                        Block frei.
efcf    20  11  f0    jsr   $f011       Bitmuster für Track in der BAM suchen
efd2    98            tya               Zeiger auf Bitmuster in Y
efd3    85  6f        sta   $6f         Zeiger merken
efd5    a5  81        lda   $81         Sektornummer
efd7    4a            lsr               geteilt durch 8
efd8    4a            lsr               ergibt das für den Block zuständige
efd9    4a            lsr               Byte in der BAM (0 bis 2)
efda    38            sec               plus Carry
efdb    65  6f        adc   $6f         plus Zeiger auf Anfang der Bitmap
efdd    a8            tay               ergibt Zeiger auf zuständiges Byte
efde    a5  81        lda   $81         Sektornummer
efe0    29  07        and   #$07        Nummer des zuständigen Bits holen
efe2    aa            tax               und als Index
efe3    b1  6d        lda   ($6d),y     Byte aus BAM holen
efe5    3d  e9  ef    and   $efe9,x     zuständiges Bit löschen (Block belegen)
efe8    60            rts               Ende
------------------------------------------------------------------------
efe9                                    Tabelle der Bitmasken für jedes Bit
efe9 01 02 04 08 10 20 40 80            eines Bytes (Zweierpotenzen).
```

```
-------------------------------------------------------------------------
eff1                                    BAM bei Bedarf auf Diskette schreiben
eff1    a9  ff       lda   #$ff         Flag für BAM schreiben
eff3    2c  f9  02   bit   $02f9        prüfen
eff6    f0  0c       beq   $f004        verZlo/e i gen
eff8    10  0a       bpl   $f004        wenn Flag nicht
effa    70  08       bvs   $f004        gesetzt ist
effc    a9  00       lda   #$00         Flag für BAM schreiben
effe    8d  f9  02   sta   $02f9        löschen
*1 f001 4c  8d  a5   jmp   $a58d        BAM auf Diskette schreiben
*0 f001 4c  8a  d5   jmp   $d58a        BAM auf Diskette schreiben
f004    60           rts                Ende
-------------------------------------------------------------------------
f005                                    Puffer für BAM löschen.
*1 f005 4c  38  a7   jmp   $a738        BAM-Zusatzpuffer für 1571 löschen
*0 f005 20  3a  ef   jsr   $ef3a        BAM lesen; Zeiger setzen
f008    a0  00       ldy   #$00         Zeiger setzen
f00a    98           tya                Akku = 0
f00b    91  6d       sta   ($6d),y      Speicherstelle löschen
f00d    c8           iny                nächste Speicherzelle
f00e    d0  fb       bne   $f00b        und weitermachen, bis Puffer gelöscht
f010    60           rts                Ende
-------------------------------------------------------------------------
f011                                    BAM-Maske im Puffer erzeugen.
f011    a5  6f       lda   $6f          Parameter
f013    48           pha                retten
f014    a5  70       lda   $70          Parameter
f016    48           pha                retten
f017    4c  25  aa   jmp   $aa25        Drivestatus holen
f01a    ea           nop                freier Platz
f01b    f0  05       beq   $f022        verzweige, wenn Drivestatus ok
f01d    a9  74       lda   #$74         sonst Nummer der Fehlermeldung
f01f    20  48  e6   jsr   $e648        '74, DRIVE NOT READY' ausgeben
f022    20  0f  f1   jsr   $f10f        Puffer- und Kanalnummer holen
f025    85  6f       sta   $6f          Kanalnummer setzen
f027    8a           txa                Puffernummer
f028    0a           asl                mal 2
f029    85  70       sta   $70          merken
f02b    aa           tax                als Index
f02c    a5  80       lda   $80          aktuelle Tracknummer
f02e    dd  9d  02   cmp   $029d,x      gleich Track für BAM?
f031    f0  0b       beq   $f03e        verzweige, wenn ja
f033    e8           inx                nächsten Kanal (Alternative)
f034    86  70       stx   $70          merken
f036    dd  9d  02   cmp   $029d,x      Track jetzt ok?
f039    f0  03       beq   $f03e        verzweige, wenn ja
f03b    20  5b  f0   jsr   $f05b        BAM herstellen
f03e    a5  70       lda   $70          Kanalnummer
f040    a6  7f       ldx   $7f          Drivenummer
f042    9d  9b  02   sta   $029b,x      merken, daß BAM wiederhergestellt
f045    0a           asl                Kanalnummer
f046    0a           asl                mal 4
f047    18           clc                und
f048    69  a1       adc   #$a1         plus 161
f04a    85  6d       sta   $6d          ergibt Zeiger auf Bitmap Lo
f04c    a9  02       lda   #$02         Übertrag des Ergebnisses
f04e    69  00       adc   #$00         addieren
f050    85  6e       sta   $6e          und Zeiger auf Bitmap Hi merken
```

```
f052   a0  00       ldy   #$00        Index löschen
f054   68           pla               Parameter wieder holen
f055   85  70       sta   $70         und abspeichern
f057   68           pla               Parameter wieder holen
f058   85  6f       sta   $6f         und abspeichern
f05a   60           rts               Ende
--------------------------------------------------------------------------
f05b                                  BAM-Masken im Puffer vertauscrer.
f05b   a6  6f       ldx   $6f         $6f Kanalnummer holen
f05d   20  df  f0   jsr   $f0df       BAM lesen
f060   a5  7f       lda   $7f         Drivenummer
f062   aa           tax               als Index
f063   0a           asl               mal 2
f064   1d  9b  02   ora   $029b,x     mit Flag verknüpfen
f067   49  01       eor   #$01        Bit 0 invertieren
f069   29  03       and   #$03        und isolieren
f06b   85  70       sta   $70         Wert merken
f06d   20  a5  f0   jsr   $f0a5       Bitmuster in BAM schreiben
f070   a5  f9       lda   $f9         Puffernummer
f072   0a           asl               mal 2
f073   aa           tax               als Index
f074   a5  80       lda   $80         aktuelle Tracknummer
f076   0a           asl               mal 4
f077   0a           asl               und
f078   95  99       sta   $99,x       als Pufferzeiger setzen
f07a   a5  70       lda   $70         Bitmuster zurückholen
f07c   0a           asl               und
f07d   0a           asl               mal 4 nehmen
f07e   a8           tay               als Index
f07f   a1  99       lda   ($99,x)     Wert aus Puffer holen
f081   99  a1  02   sta   $02a1,y     und merken
f084   a9  00       lda   #$00        bisherigen Wert
f086   81  99       sta   ($99,x)     löschen
f088   f6  99       inc   $99,x       Pufferzeiger erhöhen
f08a   c8           iny               Index erhöhen
f08b   98           tya               und
f08c   29  03       and   #$03        die Bits 0 und 1 isolieren
f08e   d0  ef       bne   $f07f       verzweige, wenn ungleich Null
f090   a6  70       ldx   $70         Bitmuster zurückholen
f092   a5  80       lda   $80         Tracknummer
f094   9d  9d  02   sta   $029d,x     merken
f097   ad  f9  02   lda   $02f9       BAM 'dirty' Flag testen
f09a   d0  03       bne   $f09f       verzweige, wenn BAM ok
*1 f09c 4c  8d  a5  jmp   $a58d       sonst BAM auf Diskette schreiben
*0 f09c 4c  8a  d5  jmp   $d58a       BAM auf Diskette schreiben
f09f   09  80       ora   #$80        BAM 'dirty' Flag
f0a1   8d  f9  02   sta   $02f9       setzen; BAM nicht ok
f0a4   60           rts               Ende
--------------------------------------------------------------------------
f0a5                                  Maske der BAM in die richtige Position im Speicher bringen.
f0a5   a8           tay               Wert merken
f0a6   b9  9d  02   lda   $029d,y     Maske für BAM aus Zwischenspeicher holen
f0a9   f0  25       beq   $f0d0       verzweige, wenn BAM nicht im Speicher
f0ab   48           pha               Wert merken
f0ac   a9  00       lda   #$00        Flag für BAM im Speicher
f0ae   99  9d  02   sta   $029d,y     setzen
f0b1   a5  f9       lda   $f9         Puffernummer
```

```
f0b3    0a          asl             mal 2
f0b4    aa          tax             als Index
f0b5    68          pla             Wert zurückholen
f0b6    0a          asl             und
f0b7    0a          asl             mal 4 nehmen
f0b8    95  99      sta   $99,x     als Pufferzeiger Lo setzen
f0ba    98          tya             Einsprungert zurückholen
f0bb    0a          asl             und
f0bc    0a          asl             mal 4 nehmen
f0bd    a8          tay             als Index nehmen
f0be    b9  a1  02  lda   $02a1,y   Wert aus Zwischenspeicher
f0c1    81  99      sta   ($99,x)   als Pufferzeiger setzen
f0c3    a9  00      lda   #$00      Zwischenspeicher
f0c5    99  a1  02  sta   $02a1,y   löschen
f0c8    f6  99      inc   $99,x     Pufferzeiger erhöhen
f0ca    c8          iny             Index erhöhen
f0cb    98          tya             vom Index
f0cc    29  03      and   #$03      die Bits 0 und 1 isolieren
f0ce    d0  ee      bne   $f0be     verzweige, wenn ungleich Null
f0d0    60          rts             Ende
-------------------------------------------------------------------------
f0d1                                Tracknummer für BAM auf Null setzeno
f0d1    a5  7f      lda   $7f       Drivenummer
f0d3    0a          asl             mal 2
f0d4    aa          tax             als Index
f0d5    a9  00      lda   #$00      Tracknummer
f0d7    9d  9d  02  sta   $029d,x   löschen
f0da    e8          inx             und
f0db    9d  9d  02  sta   $029d,x   noch einmal löschen
f0de    60          rts             Ende
-------------------------------------------------------------------------
f0df                                BAM von Diskette lesen, sofern nötig.
f0df    b5  a7      lda   $a7,x     Puffernummer
f0e1    c9  ff      cmp   #$ff      Puffer frei?
f0e3    d0  25      bne   $f10a     verzweige, wenn nein
f0e5    8a          txa             Kanalnummer
f0e6    48          pha             merken
f0e7    20  8e  d2  jsr   $d28e     freien Puffer suchen
f0ea    aa          tax             Puffernummer
f0eb    10  05      bpl   $f0f2     verzweige, wenn Puffer gefunden
f0ed    a9  70      lda   #$70      Nummer der Fehlermeldung
f0ef    20  c8  c1  jsr   $c1c8     '70, NO CHANNEL' ausgeben
f0f2    86  f9      stx   $f9       Puffernummer setzen
f0f4    68          pla             Kanalnummer zurückholen
f0f5    a8          tay             und als Index
f0f6    8a          txa             Puffernummer
f0f7    09  80      ora   #$80      Puffer als belegt kennzeichnen
f0f9    99  a7  00  sta   $00a7,y   und als Status in Tabelle
f0fc    0a          asl             Puffernummer mal 2
f0fd    aa          tax             als Index
f0fe    ad  85  fe  lda   $fe85     18, Track für BAM
f101    95  06      sta   $06,x     in Jobspeicher
f103    a9  00      lda   #$00      0, Sektor für BAM
f105    95  07      sta   $07,x     in Jobspeicher
*1 f107 4c  67  a6  jmp   $a667     BAM von Diskette lesen
*0 f107 4c  86  d5  jmp   $d586     BAM von Diskette lesen
```

```
f10a   29  0f       and   #$0f        Puffernummer isolieren
f10c   85  f9       sta   $f9         und abspeichern
f10e   60           rts               Ende
--------------------------------------------------------------------------
f10f                                  Kanalnummer für Bearbeitung der BAM in den Akku holen.
f10f   a9  06       lda   #$06        Kanalnummer 6
f111   a6  7f       ldx   $7f         Drivenummer
f113   d0  03       bne   $f118       verzweige, wenn Null
f115   18           clc               Kanalnummer
f116   69  07       adc   #$07        plus Puffernummer
f118   60           rts               ergibt 13 (Kanal für BAM); Ende
--------------------------------------------------------------------------
f119                                  Kanalnummer für BAM holen und in X übergeben.
f119   20  0f  f1    jsr   $f10f       Kanalnummer holen
f11c   aa           tax               und nach X
f11d   60           rts               Ende
--------------------------------------------------------------------------
f11e                                  Mit der Angabe der aktuellen Track- und Sektornummer sucht
                                      diese Routine nach dem nächsten verfügbaren Sektor.
f11e   20  3e  de    jsr   $de3e       Track und Sektor holen
f121   a9  03       lda   #$03        Zählwert
f123   85  6f       sta   $6f         setzen
f125   a9  01       lda   #$01        Bit 1 als Flag für BAM nicht auf Diskette
f127   0d  f9  02    ora   $02f9       schreiben
f12a   8d  f9  02    sta   $02f9       setzen
*1 f12d 4c  db  a8    jmp   $a8db       Zählwert und ggf. Bitmuster aus BAM holen
*0 f12d a5  6f       lda   $6f         aktuellen Zeiger in BAM
*0 f12f 48           pha               merken
f130   20  11  f0    jsr   $f011       richtiges Bitmuster in BAM holen
f133   68           pla               Zählwert zurückholen
f134   85  6f       sta   $6f         und wieder merken
f136   b1  6d       lda   ($6d),y     Anzahl der freien Blöcke des Tracks
f138   d0  39       bne   $f173       verzweige, wenn noch Blöcke frei
f13a   a5  80       lda   $80         Tracknummer
f13c   cd  85  fe    cmp   $fe85       mit 18 (Directorytrack) vergleichen
f13f   f0  19       beq   $f15a       verzweige, wenn gleich
f141   90  1c       bcc   $f15f       verzweige, wenn kleiner 18
f143   e6  80       inc   $80         Tracknummer plus 1
f145   a5  80       lda   $80         Tracknummer
*1 f147 cd  ac  02    cmp   $02ac       mit höchster Nummer vergleichen
*0 f147 cd  d7  fe    cmp   $fed7       mit höchster Nummer vergleichen
f14a   d0  e1       bne   $f12d       verzweige, wenn ungleich
f14c   ae  85  fe    ldx   $fe85       18; Directorytrack
f14f   ca           dex               minus 1
f150   86  80       stx   $80         setzen
f152   a9  00       lda   #$00        Sektor 0
f154   85  81       sta   $81         setzen
f156   c6  6f       dec   $6f         Zählwert vermindern
f158   d0  d3       bne   $f12d       weitersuchen, wenn ungleich Null
f15a   a9  72       lda   #$72        sonst Nummer der Fehlermeldung
f15c   20  c8  c1    jsr   $c1c8       '72, DISK FULL' ausgeben
f15f   c6  80       dec   $80         Tracknummer minus 1
f161   d0  ca       bne   $f12d       weitermachen, wenn noch nicht Null
f163   ae  85  fe    ldx   $fe85       18; Directorytrack
f166   e8           inx               plus 1
f167   86  80       stx   $80         als Tracknummer setzen
```

```
f169    a9  00       lda   #$00        Sektor 0
f16b    85  81       sta   $81         setzen
f16d    c6  6f       dec   $6f         Zählwert vermindern
f16f    d0  bc       bne   $f12d       weitermachen, wenn noch nicht Null
f171    f0  e7       beq   $f15a       unbedingt; '72, DISK FULL' ausgeben
-------------------------------------------------------------------------
f173                                   Optimalen nächsten Sektor der aktuellen Spur ausfindig
                                       machen.
f173    a5  81       lda   $81         Sektornummer
f175    18           clc               und
f176    65  69       adc   $69         Schrittweite (normal 6) addieren
f178    85  81       sta   $81         neue Sektornummer merken
f17a    a5  80       lda   $80         Tracknummer
f17c    20  4b  f2   jsr   $f24b       dazu maximale Sektornummer holen
f17f    8d  4e  02   sta   $024e       und merken
f182    8d  4d  02   sta   $024d       zweimal!
f185    c5  81       cmp   $81         mit neuer Sektornummer vergleichen
f187    b0  0c       bcs   $f195       verzweige, wenn größer gleich
f189    38           sec               und
f18a    a5  81       lda   $81         aktuelle Sektornummer
f18c    ed  4e  02   sbc   $024e       minus maximale Sektornummer
f18f    85  81       sta   $81         setzen
f191    f0  02       beq   $f195       verzweige, wenn gleich
f193    c6  81       dec   $81         Sektornummer minus 1
f195    20  fa  f1   jsr   $f1fa       ggf. anderen freien Sektor suchen
f198    f0  03       beq   $f19d       verzweige, wenn kein Sektor der Spur frei
f19a    4c  90  ef   jmp   $ef90       sonst Block in BAM belegen; Ende
f19d    a9  00       lda   #$00        Sektornummer 0
f19f    85  81       sta   $81         setzen
f1a1    20  fa  f1   jsr   $f1fa       wiederum freien Sektor suchen
f1a4    d0  f4       bne   $f19a       verzweige, wenn gefunden
f1a6    4c  f5  f1   jmp   $f1f5       '71, DIR ERROR' ausgeben; Ende
-------------------------------------------------------------------------
f1a9                                   Nächsten optimalen Sektor suchen und belegen.
f1a9    a9  01       lda   #$01        Flag für BAM nicht schreiben
f1ab    0d  f9  02   ora   $02f9       setzen
f1ae    8d  f9  02   sta   $02f9       und abspeichern
f1b1    a5  86       lda   $86         Zwischenspeicherwert
f1b3    48           pha               retten
f1b4    a9  01       lda   #$01        Zählwert für Tracknummern
f1b6    85  86       sta   $86         setzen
f1b8    ad  85  fe   lda   $fe85       18; Directorytrack
f1bb    38           sec               minus
f1bc    e5  86       sbc   $86         Zähler für Tracks
f1be    85  80       sta   $80         Ergebnis merken
f1c0    90  09       bcc   $f1cb       verzweige, wenn Zähler kleiner 18
f1c2    f0  07       beq   $f1cb       verzweige, wenn Zähler gleich 18
*1 f1c4 4c  05  a9   jmp   $a905       richtiges Bitmuster in BAM holen
*0 f1c4 20  11  f0   jsr   $f011       Zeiger in BAM setzen
f1c7    b1  6d       lda   ($6d),y     Zahl der freien Blöcke des Tracks
f1c9    d0  1b       bne   $f1e6       verzweige, wenn noch Blöcke frei
f1cb    ad  85  fe   lda   $fe85       18; Directorytrack
f1ce    18           clc               plus
f1cf    65  86       adc   $86         Zähler für Tracks
f1d1    85  80       sta   $80         als Tracknummer merken
f1d3    e6  86       inc   $86         plus 1
*1 f1d5 cd  ac  02   cmp   $02ac       mit höchster Tracknummer vergleichen
*0 f1d5 cd  d7  fe   cmp   $fed7       mit höchster Tracknummer vergleichen
```

```
f1d8    90  05       bcc   $f1df       verzweige, wenn kleiner
f1da    a9  67       lda   #$67        Nummer der Fehlermeldung
f1dc    20  45  e6   jsr   $e645       '67, ILLEGAL TRACK OR SECTOR' ausgeben
*1 f1df 4c  1e  a9   jmp   $a91e       richtiges Bitmaster in BAM holen
*0 f1df 20  11  f0   jsr   $f011       Zeiger in BAM setzen
f1e2    b1  6d       lda   ($6d),y     Zahl der freien Blöcke des Tracks
f1e4    f0  d2       beq   $f1b8       verzweige, wenn kein Block fre;
f1e6    68           pla               Zwischenspeicherwert zurückholen
f1e7    85  86       sta   $86         und wieder abspeichern
f1e9    a9  00       lda   #$00        Sektor 0 als Startwert
f1eb    85  81       sta   $81         setzen
f1ed    20  fa  f1   jsr   $f1fa       und freien Sektor suchen
f1f0    f0  03       beq   $f1f5       verzweige, wenn keinen gefunden
f1f2    4c  90  ef   jmp   $ef90       sonst Block in BAM belegen; Ende
f1f5    a9  71       lda   #$71        Nummer der Fehlermeldung
f1f7    20  45  e6   jsr   $e645       '71, DIR ERROR' ausgeben
-------------------------------------------------------------------------
f1fa                                   nächsten freien Sektor eines Tracks suchen.
f1fa    20  11  f0   jsr   $f011       richtiges Bitmuster in BAM holen
f1fd    98           tya               Index auf Beginn des Bitmusters
f1fe    48           pha               merken
f1ff    20  20  f2   jsr   $f220       Bitmuster in BAM prüfen
f202    a5  80       lda   $80         Tracknummer
f204    20  4b  f2   jsr   $f24b       maximale Anzahl von Sektoren dieses Tracks holen
f207    8d  4e  02   sta   $024e       und merken
f20a    68           pla               Index in Bitmuster zurückholen
f20b    85  6f       sta   $6f         und merken
f20d    a5  81       lda   $81         Sektornummer
f20f    cd  4e  02   cmp   $024e       mit maximaler Anzahl vergleichen
f212    b0  09       bcs   $f21d       verzweige, wenn größer gleich
f214    20  d5  ef   jsr   $efd5       Bitstatus des Sektors holen
f217    d0  06       bne   $f21f       verzweige, wenn Sektor frei
f219    e6  81       inc   $81         Sektornummer plus 1
f21b    d0  f0       bne   $f20d       und wieder prüfen
f21d    a9  00       lda   #$00        Flag für alle Sektoren belegt setzen
f21f    60           rts               Ende
-------------------------------------------------------------------------
f220                                   Gültigkeit der Blockangaben in der BAM prüfen.
f220    a5  6f       lda   $6f         Speicherwert
f222    48           pha               retten
f223    a9  00       lda   #$00        Zähler gleich Null
f225    85  6f       sta   $6f         setzen
f227    ac  86  fe   ldy   $fe86       4; Anzahl der Bytes pro Track
f22a    88           dey               minus 1 als Index
f22b    a2  07       ldx   #$07        Bitzeiger
f22d    b1  6d       lda   ($6d),y     8 Bits aus BAM holen
f22f    3d  e9  ef   and   $efe9,x     und ein Bit isolieren
f232    f0  02       beq   $f236       verzweige, wenn Block belegt
f234    e6  6f       inc   $6f         Anzahl der freien Blöcke plus 1
f236    ca           dex               Bitzeiger auf nächstes Bit
f237    10  f4       bpl   $f22d       alle Sektoren überprüfen
f239    88           dey               Zeiger auf die nächsten 8 Bits
f23a    d0  ef       bne   $f22b       und wieder prüfen
f23c    b1  6d       lda   ($6d),y     eingetragene Zahl der freien Blöcke
f23e    c5  6f       cmp   $6f         mit Rechenwert vergleichen
f240    d0  04       bne   $f246       Fehler, wenn ungleich
f242    68           pla               Speicherwert zurückholen
f243    85  6f       sta   $6f         und wieder setzen
```

```
f245   60           rts             Ende
f246   a9  71       lda   #$71      Nummer der Fehlermeldung
f248   20  45  e6   jsr   $e645     '71, DIR ERROR' ausgeben; Ende
------------------------------------------------------------------------
f24b                               Stellt die Maximalanzahl der Sektoren einer Spur fest;
                                   Tracknummer in A.
*1 f24b20  4f  a7   jsr   $a74f     Bearbeitung für Tracks größer 35
*0 f24bae  d6  fe   ldx   $fed6     Anzahl der Trackzonen auf Diskette
f24e   dd  d6  fe   cmp   $fed6,x   Sektornummer mit Maximum vergleichen
f251   ca           dex             Zeiger minus 1
f252   b0  fa       bcs   $f24e     verzweige, wenn Sektor größer
f254   bd  d1  fe   lda   $fed1,x   Maximalzahl der Sektoren erhöhen
f257   60           rts             Ende
f258   60           rts             Ende
------------------------------------------------------------------------
f259                               Initialisierung der Register für den Diskcontroller.
f259   a9  6f       lda   #$6f      Bit 4 und 7 auf Eingang schalten
f25b   8d  02  1c   sta   $1c02     entspricht SYNC und WRITE PROTECT
f25e   29  f0       and   #$f0      korrespondierende Bits
f260   8d  00  1c   sta   $1c00     Status für Drivemotor löschen
f263   4c  f8  a9   jmp   $a9f8     PCR lesen
f266   29  fe       and   #$fe      CA2 auf negative Flanke triggern (BYTE READY)
f268   09  0e       ora   #$0e      CB1 auf Eingang und CB2 als Kontrolle für
f26a   09  e0       ora   #$e0      Schreiben/Lesen schalten
f26c   8d  0c  1c   sta   $1c0c     PCR setzen
f26f   a9  41       lda   #$41      Timer 1 auf 'free running mode'
f271   8d  0b  1c   sta   $1c0b     schalten
f274   a9  00       lda   #$00      Timerwert für IRQ (Diskcontroller) etwa
f276   8d  06  1c   sta   $1c06     alle 8 ms setzen
f279   a9  20       lda   #$20      Timerwert Hi
f27b   8d  07  1c   sta   $1c07     setzen
f27e   8d  05  1c   sta   $1c05     Timer starten
f281   a9  7f       lda   #$7f      IRQ-Flag
f283   8d  0e  1c   sta   $1c0e     löschen
f286   a9  c0       lda   #$c0      Interruptmaske
f288   8d  0d  1c   sta   $1c0d     setzen;
f28b   8d  0e  1c   sta   $1c0e     IRQs zulassen
f28e   a9  ff       lda   #$ff      alle Laufwerke inaktiv
f290   85  3e       sta   $3e       setzen
f292   85  51       sta   $51       keine laufende Formatierung anzeigen
f294   a9  08       lda   #$08      8 als Konstante für Blockheader
f296   85  39       sta   $39       setzen
f298   a9  07       lda   #$07      7 als Konstante für Datenblock
f29a   85  47       sta   $47       setzen
f29c   a9  05       lda   #$05      Zeiger auf
f29e   85  62       sta   $62       $fa05; Routine für
f2a0   a9  fa       lda   #$fa      Steppermotorbehandlung
f2a2   85  63       sta   $63       setzen
f2a4   a9  c8       lda   #$c8      200; minimale Anzahl der Schritte für
f2a6   85  64       sta   $64       den schnellen Steppermodus
f2a8   a9  04       lda   #$04      4; Wert zum Anfahren und Abbremsen
f2aa   85  5e       sta   $5e       des Steppermotors setzen
f2ac   a9  04       lda   #$04      und noch einmal
f2ae   85  5f       sta   $5f       setzen (Drive 1)
```

```
------------------------------------------------------------------------
f2b0                                    IRQ-Routine des Diskcontrollers. Prüft auf Jobs und führt
                                        diese ggf. aus.
f2b0    ba          tsx                 Stackpointer
f2b1    86  49      stx   $49          merken
f2b3    ad  04  1c  lda   $1c04        IRQ-Flag durch Lesen löschen
f2b6    ad  0c  1c  lda   $1c0c        Bits 1,2 und 3 setzen, um
f2b9    09  0e      ora   #$0e         die BYTE READY-Leitung zu
f2bb    8d  0c  1c  sta   $1c0c        initialisieren
f2be    a0  05      ldy   #$05         Index in Jobspeicher setzen
f2c0    b9  00  00  lda   $0000,y      liegt Job für Puffer an?
f2c3    10  2e      bpl   $f2f3        verzweige, wenn nein
f2c5    c9  d0      cmp   #$d0         Job: Programm im Puffer ausführe?
f2c7    d0  04      bne   $f2cd        verzweige, wenn nein
f2c9    98          tya                Puffernummer zur Ausführung
f2ca    4c  70  f3  jmp   $f370        übergeben und Programm starten
f2cd    29  01      and   #$01         Drivenummer isolieren
f2cf    f0  07      beq   $f2d8        verzweige, wenn Drivenummer 0
f2d1    84  3f      sty   $3f          Puffernummer merken
f2d3    a9  0f      lda   #$0f         Nummer der Fehlermeldung
f2d5    4c  69  f9  jmp   $f969        '74, DRIVE NOT READY' ausgeben
f2d8    aa          tax                Drivenummer (0) nach X
f2d9    85  3d      sta   $3d          Nummer für Diskcontroller setzen
f2db    c5  3e      cmp   $3e          läuft aktuelles Drive?
f2dd    f0  0a      beq   $f2e9        verzweige, wenn ja
f2df    20  7e  f9  jsr   $f97e        Laufwerksmotor einschalten
f2e2    a5  3d      lda   $3d          Drivenummer für Diskcontroller
f2e4    85  3e      sta   $3e          als aktuelles Drive übernehmen
f2e6    4c  9c  f9  jmp   $f99c        weiter in Jobschleife
f2e9    a5  20      lda   $20          Drive schon auf konstanter Geschwindigkeit?
f2eb    30  03      bmi   $f2f0        verzweige zur Jobschleife, wenn nein
f2ed    0a          asl                Steppermotor in Aktion?
f2ee    10  09      bpl   $f2f9        verzweige, wenn nein
f2f0    4c  9c  f9  jmp   $f99c        zur Jobschleife
f2f3    88          dey                Index in Jobspeicher minus 1
f2f4    10  ca      bpl   $f2c0        weiter, wenn noch Puffer übrig
f2f6    4c  9c  f9  jmp   $f99c        zur Jobsch lei fe
f2f9    a9  20      lda   #$20         Flag für Drive in Aktion
f2fb    85  20      sta   $20          setzen
f2fd    a0  05      ldy   #$05         Index in Jobspeicher
f2ff    84  3f      sty   $3f          setzen
f301    20  93  f3  jsr   $f393        Pufferadresse für Job setzen
f304    30  1a      bmi   $f320        verzweige, wenn Job anliegt
f306    c6  3f      dec   $3f          Index auf nächsten Jobspeicher
f308    10  f7      bpl   $f301        nächsten Jobspeicher prüfen
f30a    a4  41      ldy   $41          Puffernummer für nächsten Job
f30c    20  95  f3  jsr   $f395        Pufferadresse für Job setzen
f30f    a5  42      lda   $42          Traekdifferenz zu letztem Job
f311    85  4a      sta   $4a          setzen
f313    06  4a      asl   $4a          Wert mal 2 (Anzahl der Stepperschritte)
f315    a9  60      lda   #$60         Flag für Steppermodus
f317    85  20      sta   $20          setzen
f319    b1  32      lda   ($32),y      Tracknummer für Job holen
f31b    85  22      sta   $22          und übernehmen
f31d    4c  9c  f9  jmp   $f99c        Kopfpositionierung vorbereiten; Jobschleife
f320    29  01      and   #$01         Drivenummer isolieren
f322    c5  3d      cmp   $3d          gleich Nummer des letzten Jobs?
f324    d0  e0      bne   $f306        verzweige, wenn nein
```

```
f326   a5  22       lda   $22         Tracknummer des letzten Jobs
f328   f0  12       beq   $f33c       verzweige, wenn nicht gesetzt
f32a   38           sec               und
f32b   f1  32       sbc   ($32),y     Abstand zu neuem Track berechnen
f32d   f0  0d       beq   $f33c       verzweige, wenn gleicher Track
f32f   49  ff       eor   #$ff        Anzahl der Steps erzeugen
f331   85  42       sta   $42         und setzen
f333   e6  42       inc   $42         plus 1
f335   a5  3f       lda   $3f         Puffernummer des Jobs
f337   85  41       sta   $41         übernehmen
f339   4c  06  f3    jmp   $f306       weitere Jobs prüfen
f33c   a2  04       ldx   #$04        4; Anzahl der Trackzonen auf der Diskette (GCR)
f33e   b1  32       lda   ($32),y     Tracknummer für Job
f340   85  40       sta   $40         übernehmen
f342   dd  d6  fe   cmp   $fed6,x     mit Zonengrenzen vergleichen, um
f345   ca           dex               Anzahl der Sektoren in der Zone
f346   b0  fa       bcs   $f342       zu errechnen
f348   bd  d1  fe   lda   $fed1,x     Anzahl der Sektoren holen
f34b   85  43       sta   $43         und übernehmen
f34d   8a           txa               Zonennummer
f34e   0a           asl               jetzt
f34f   0a           asl               mal 32
f350   0a           asl               rechnen, um
f351   0a           asl               die Bits an die
f352   0a           asl               richtige Stelle im Byte zu verschieben
f353   85  44       sta   $44         als Wert für Kontrollport setzen
f355   ad  00  1c   lda   $1c00       Kontrollport des Diskcontrollers
f358   29  9f       and   #$9f        entsprechende Bits (5 und 6) löschen, um Timer-
f35a   05  44       ora   $44         konstante für Diskcontroller-Hardware zu setzen
f35c   8d  00  1c   sta   $1c00       wichtig für Timing beim Diskettenbetrieb
f35f   a6  3d       ldx   $3d         Drivenummer
f361   a5  45       lda   $45         Jobcode für dieses Drive
f363   c9  40       cmp   #$40        BUMP des Tonkopfes ausführen (Nullanschlag)?
f365   f0  15       beq   $f37c       verzweige, wenn ja
f367   c9  60       cmp   #$60        Jobprogramm im Puffer ausführen?
f369   f0  03       beq   $f36e       verzweige, wenn ja
f36b   4c  b1  f3    jmp   $f3b1       sonst Blockheader auf Track suchen
f36e   a5  3f       lda   $3f         Puffernummer
f370   18           clc               plus 3
f371   69  03       adc   #$03        rechnen, um die
f373   85  31       sta   $31         Pufferadresse Hi zu erhalten
f375   a9  00       lda   #$00        Pufferadresse Lo
f377   85  30       sta   $30         setzen
f379   6c  30  00    jmp   ($0030)     Sprung in Puffer; Programm ausführen
-------------------------------------------------------------------------------
f37c                                   Routine zum Ausführen eines BUMP, d.h. der Tonkopf wird an
                                       den Nullanschlag zurückgefahren, um danach eine
                                       einwandfreie Positionierung zu sichern.
f37c   a9  60       lda   #$60        Flag für Steppermodus
f37e   85  20       sta   $20         setzen
f380   ad  00  1c   lda   $1c00       Steuerport des Diskcontrollers
f383   29  fc       and   #$fc        Bit für Steppermotor löschen, d.h.
f385   8d  00  1c   sta   $1c00       Steppermotor an
f388   a9  a4       lda   #$a4        164; entspricht -45; Anzahl der
f38a   85  4a       sta   $4a         zu überfahrenden Tracks bei BUMP
f38c   a9  01       lda   #$01        Tracknummer 1 als Anschluß an BUMP
f38e   85  22       sta   $22         setzen
f390   4c  69  f9    jmp   $f969       Abschluß der Jobschleife
```

```
------------------------------------------------------------------------
f393                                    Pufferadresse und Pufferzeiger für Job setzen; in $30/31
                                        und $32.
f393    a4  3f      ldy   $3f           Index in Jobspeicher
f395    b9  00  00  lda   $0000,y       Jobcode holen
f398    48          pha                 und merken
f399    10  10      bpl   $f3ab         verzweige, wenn kein aktueller .:=
f39b    29  78      and   #$78          reinen Jobcode isolieren
f39d    85  45      sta   $45           und abspeichern
f39f    98          tya                 Index in Jobspeicher
f3a0    0a          asl                 mal 2
f3a1    69  06      adc   #$06          plus 6
f3a3    85  32      sta   $32           als Zeiger in Jobspeicher (auf 7"a:i-.rre-;
f3a5    98          tya                 Index in Jobspeicher
f3a6    18          clc                 und
f3a7    69  03      adc   #$03          plus 3
f3a9    85  31      sta   $31           als Pufferadresse Hi setzen
f3ab    a0  00      ldy   #$00          Pufferadresse Lo
f3ad    84  30      sty   $30           setzen
f3af    68          pla                 Jobcode zurückholen
f3b0    60          rts                 Ende
------------------------------------------------------------------------
f3b1                                    Suchroutine zum Finden einer Spur auf Diskette (SEEK).
                                        Hierbei wird nach einem beliebigen gültigen Blockheader
                                        gesucht, wobei 90 Leseversuche gemacht werden.
f3b1    a2  5a      ldx   #$5a          90; Anzahl der Leseversuche
f3b3    86  4b      stx   $4b           als Zähler setzen
f3b5    a2  00      ldx   #$00          Index in Zwischenspeicher setzen
f3b7    a9  52      lda   #$52          GCR-Code für S08 (Blockheaderkennzeichen)
f3b9    85  24      sta   $24           für Suche setzen
f3bb    20  56  f5  jsr   $f556         SYNC-Signal abwarten
f3be    50  fe      bvc   $f3be         warten bis Byte eingelesen
f3c0    b8          clv                 Flag löschen
f3c1    ad  01  1c  lda   $1c01         Byte vom Diskcontroller holen
f3c4    c5  24      cmp   $24           und mit Blockheaderkennzeichen vergleichen
f3c6    d0  3f      bne   $f407         verzweige, wenn kein Headerkennzeichen
f3c8    50  fe      bvc   $f3c8         Byte einlesen
f3ca    b8          clv                 Flag für 'Byte eingelesen' (BYTE READY) löschen
f3cb    ad  01  1c  lda   $1c01         Byte vom Diskcontroller holen
f3ce    95  25      sta   $25,x         und in Zwischenspeicher für Blockheader
f3d0    e8          inx                 nächstes Byte
f3d1    e0  07      cpx   #$07          schon 7 Bytes eingelesen?
f3d3    d0  f3      bne   $f3c8         weiterlesen, wenn nein
f3d5    20  97  f4  jsr   $f497         Blockheader von GCR nach binär umwandeln
f3d8    a0  04      ldy   #$04          Index für Prüfsumme
f3da    a9  00      lda   #$00          Prüfsumme über den
f3dc    59  16  00  eor   $0016,y       Blockheader bilden
f3df    88          dey                 noch ein Byte?
f3e0    10  fa      bpl   $f3dc         ja; weitermachen
f3e2    c9  00      cmp   #$00          Prüfsumme korrekt?
f3e4    d0  38      bne   $f41e         '27, READ ERROR', wenn nein
f3e6    a6  3e      ldx   $3e           Drivenummer für Job
f3e8    a5  18      lda   $18           Tracknummer von gelesenem Blockheader
f3ea    95  22      sta   $22,x         übernehmen
f3ec    a5  45      lda   $45           Jobcode
f3ee    c9  30      cmp   #$30          Job: Sektor suchen (SEEK)?
f3f0    f0  1e      beq   $f410         verweige, wenn ja
```

```
f3f2   a5  3e       lda   $3e         Drivenummer für den Job
f3f4   0a           asl               mal 2
f3f5   a8           tay               als Index
f3f6   b9  12  00   lda   $0012,y     ID 1 holen und
f3f9   c5  16       cmp   $16         mit gelesener ID vergleichen
f3fb   d0  1e       bne   $f41b       '29, DISK ID MISMATCH', wenn falsch
f3fd   b9  13  00   lda   $0013,y     ID 2 holen
f400   c5  17       cmp   $17         und mit gelesener ID vergleichen
f402   d0  17       bne   $f41b       '29, D:SK ID MISMATCH', wenn ungleich
f404   4c  23  f4   jmp   $f423       nächstbesten Sektor bearbeiten
f407   c6  4b       dec   $4b         Zähler für Leseversuche minus 1
f409   d0  b0       bne   $f3bb       weitermachen, wenn noch keine 90 Versuche
f40b   a9  02       lda   #$02        Nummer der Fehlermeldung
f40d   20  69  f9   jsr   $f969       '20, READ ERROR' ausgeben
f410   a5  16       lda   $16         ID 1 vom Blockheader
f412   85  12       sta   $12         als neue ID 1 setzen
f414   a5  17       lda   $17         ID 2 vom Blockheader
f416   85  13       sta   $13         ebenfalls übernehmen
f418   a9  01       lda   #$01        Nummer der Rückmeldung (alles ok!)
f41a   2c           .byte $2c         nächsten Befehl überspringen
f41b   a9  0b       lda   #$0b        Nummer für '29, DISK ID MISMATCH'
f41d   2c           .byte $2c         nächsten Befehl überspringen
f41e   a9  09       lda   #$09        Nummer für '27, READ ERROR'
f420   4c  69  f9   jmp   $f969       (Fehler-)Meldungen ausgeben; Ende
-------------------------------------------------------------------------
f423                                  Sucht nach dem besten Job, der bearbeitet werden kann.
                                      Günstig ist jeweils der Job, der mit der geringsten
                                      Kopfbewegung auf der Diskette verbunden ist, also am
                                      nächsten liegt.
f423   a9  7f       lda   #$7f        Entfernungszähler zu nächstem Sektor
f425   85  4c       sta   $4c         setzen
f427   a5  19       lda   $19         Sektornummer vom Blockheader
f429   18           clc               mit
f42a   69  02       adc   #$02        2 addieren; gibt optimalen Sektor
f42c   c5  43       cmp   $43         kleiner als der Maximalwert?
f42e   90  02       bcc   $f432       verzweige, wenn ja; alles ok
f430   e5  43       sbc   $43         sonst Maximalwert abziehen
f432   85  4d       sta   $4d         und eventuellen Wert für nächsten Sektor merken
f434   a2  05       ldx   #$05        Puffernummer
f436   86  3f       stx   $3f         setzen
f438   a2  ff       ldx   #$ff        zugehörige Pufferadresse
f43a   20  93  f3   jsr   $f393       setzen
f43d   10  44       bpl   $f483       verzweige, wenn kein Job für diesen Puffer
f43f   85  44       sta   $44         sonst Jobcode merken
f441   29  01       and   #$01        Drivenummer isolieren
f443   c5  3e       cmp   $3e         und mit aktuellem Drive vergleichen
f445   d0  3c       bne   $f483       verzweige, wenn Job für anderes Drive
f447   a0  00       ldy   #$00        Index in Jobspeicher
f449   b1  32       lda   ($32),y     Tracknummer für Job holen
f44b   c5  40       cmp   $40         gleich Tracknummer dieses Jobs?
f44d   d0  34       bne   $f483       verzweige, wenn nein
f44f   a5  45       lda   $45         Jobcode holen
f451   c9  60       cmp   #$60        Jobprogramm im Puffer ausführen?
f453   f0  0c       beq   $f461       verzweige, wenn ja
f455   a0  01       ldy   #$01        Index in Jobspeicher
f457   38           sec               zeigt jetzt auf Sektornummer
f458   b1  32       lda   ($32),y     Sektornummer aus Jobspeicher holen
f45a   e5  4d       sbc   $4d         größer gleich neue Sektornummer?
```

```
f45c   10  03       bpl   $f461       verzweige, wenn ja
f45e   18           clc               sonst
f45f   65  43       adc   $43         Maximalzahl addieren
f461   c5  4c       cmp   $4c         und mit Sektorzähler prüfen
f463   b0  1e       bcs   $f483       verzweige, wenn Entfernung ungünstig
f465   48           pha               sonst Vergleichswert merken
f466   a5  45       lda   $45         Jobcode
f468   f0  14       beq   $f47e       verzweige, wenn kein Job anliegt
f46a   68           pla               Vergleichswert holen
f46b   c9  09       cmp   #$09        kleiner als 9?
f46d   90  14       bcc   $f483       nächsten Job prüfen, wenn ja
f46f   c9  0c       cmp   #$0c        größer gleich 12?
f471   b0  10       bcs   $f483       nächsten Job prüfen, wenn ja
f473   85  4c       sta   $4c         Entfernungswert merken
f475   a5  3f       lda   $3f         Puffernummer für den Job
f477   aa           tax               jetzige Nummer merken
f478   69  03       adc   #$03        außerdem plus 3
f47a   85  31       sta   $31         als Pufferadresse Hi setzen
f47c   d0  05       bne   $f483       unbedingter Sprung; nächsten Job prüfen
f47e   68           pla               Entfernungswert zurückholen
f47f   c9  06       cmp   #$06        kleiner als 6
f481   90  f0       bcc   $f473       Job bearbeiten, da Aufwand gering
f483   c6  3f       dec   $3f         Puffernummer minus 1
f485   10  b3       bpl   $f43a       nächsten Job prüfen
f487   8a           txa               wurde ein Job gefunden?
f488   10  03       bpl   $f48d       verzweige, wenn ja
f48a   4c  9c  f9    jmp   $f99c       zur Jobschleife
f48d   86  3f       stx   $3f         Puffernummer wieder merken
f48f   20  93  f3    jsr   $f393       zugehörige Pufferadresse setzen
f492   a5  45       lda   $45         Jobcode
f494   4c  ca  f4    jmp   $f4ca       prüfen; ggf. Datenblock lesen
--------------------------------------------------------------------------
f497                                   Konvertiert die Bytes des gelesenen Headers vom GCR-(5
                                       Bit)'Code in den normalen Binär-(4 Bit)-Code. Der GCR-
                                       codierte Header steht dabei ab $24; das Ergebnis der
                                       Umwandlung steht dann von $16 bis $1a und zwar:
                                       $16 - ID 1 der Diskette
                                       $17 - ID 2 der Diskette
                                       $18 - Tracknummer des Sektors
                                       $19 - Sektornummer des Sektors
                                       $1a - Prüfsumme über den Blockheader.
f497   a5  30       lda   $30         Pufferadresse Lo
f499   48           pha               retten
f49a   a5  31       lda   $31         Pufferadresse Hi
f49c   48           pha               retten
f49d   a9  24       lda   #$24        Pufferzeiger Lo auf $24
f49f   85  30       sta   $30         setzen
f4a1   a9  00       lda   #$00        Pufferzeiger Hi auf $00
f4a3   85  31       sta   $31         setzen; ergibt Adresse $0024
f4a5   a9  00       lda   #$00        Zeiger auf erstes GCR-Byte für Decodierung
f4a7   85  34       sta   $34         setzen
f4a9   20  e6  f7    jsr   $f7e6       5 GCR-Bytes in 4 Binärbytes konvertieren
f4ac   a5  55       lda   $55         Tracknummer
f4ae   85  18       sta   $18         übernehmen
f4b0   a5  54       lda   $54         Sektornummer
f4b2   85  19       sta   $19         übernehmen
f4b4   a5  53       lda   $53         Prüfsumme
```

```
f4b6   85  1a       sta   $1a         übernehmen
f4b8   20  e6  f7   jsr   $f7e6       5 GCR-Bytes in 4 Binärbytes konvertieren
f4bb   a5  52       lda   $52         zweites Zeichen der ID (ID 2)
f4bd   85  17       sta   $17         übernehmen
f4bf   a5  53       lda   $53         erstes Zeichen der ID (ID 1)
f4c1   85  16       sta   $16         übernehmen
f4c3   68           pla               Pufferadresse Hi zurückholen
f4c4   85  31       sta   $31         und setzen
f4c6   68           pla               Pufferadresse Lo zurückholen
f4c7   85  30       sta   $30         und setzen
f4c9   60           rts               Ende
-------------------------------------------------------------------------
f4ca                                  Prüft auf Jobcode für Lesen; wenn ja, wird der verlangte
                                      Blockheader gesucht und der Block gelesen und decodiert.
f4ca   c9  00       cmp   #$00        Jobcode für Block lesen (READ)
f4cc   f0  03       beq   $f4d1       verzweige, wenn ja
f4ce   4c  6e  f5   jmp   $f56e       sonst Jobcode weiter prüfen
f4d1   20  0a  f5   jsr   $f50a       gewünschten Blockheader suchen
f4d4   50  fe       bvc   $f4d4       Byte einlesen
f4d6   b8           clv               Flag wieder löschen
f4d7   ad  01  1c   lda   $1c01       Byte vom Diskcontroller holen
f4da   91  30       sta   ($30),y     und in den Puffer schreiben
f4dc   c8           iny               nächstes Byte
f4dd   d0  f5       bne   $f4d4       und 256 Bytes einlesen
f4df   a0  ba       ldy   #$ba        Index in Ausweichpuffer
f4e1   50  fe       bvc   $f4e1       Byte einlesen
f4e3   b8           clv               Flag wieder löschen
f4e4   ad  01  1c   lda   $1c01       Byte vom Diskcontroller holen
f4e7   99  00  01   sta   $0100,y     und in Ausweichpuffer schreiben
f4ea   c8           iny               nächstes Byte
f4eb   d0  f4       bne   $f4e1       weitermachen, bis 70 Bytes gelesen
f4ed   20  e0  f8   jsr   $f8e0       Pufferinhalte nach Binär konvertieren
f4f0   a5  38       lda   $38         erstes Byte des Datenblocks
f4f2   c5  47       cmp   $47         gleich $07; Datenblockkennzeichen?
f4f4   f0  05       beq   $f4fb       verzweige, wenn ja
f4f6   a9  04       lda   #$04        Nummer der Fehlermeldung
f4f8   4c  69  f9   jmp   $f969       '22, READ ERROR' ausgeben
f4fb   20  e9  f5   jsr   $f5e9       Prüfsumme über Datenblock berechnen
f4fe   c5  3a       cmp   $3a         mit gelesenem Wert vergleichen
f500   f0  03       beq   $f505       verzweige, wenn korrekt
f502   a9  05       lda   #$05        Nummer für '23, READ ERROR'
f504   2c           .byte $2c         nächsten Befehl überspringen
f505   a9  01       lda   #$01        Nummer für '00, OK'
f507   4c  69  f9   jmp   $f969       Rückmeldung ausgeben
-------------------------------------------------------------------------
f50a                                  Sucht nach einem bestimmten Blockheader und wartet das
                                      SYNC-Signal des nachfolgenden Daten- blocks ab.
f50a   20  10  f5   jsr   $f510       Blockheader suchen
f50d   4c  56  f5   jmp   $f556       SYNC-Signal abwarten; Ende
-------------------------------------------------------------------------
f510                                  Sucht nach einem Blockheader, dessen Parameter vorher
                                      gesetzt wurden. Die Parameter sind ent. sprechend den
                                      speicherstellen S12 und $13 zu setzen; im Jobspeicher muß
                                      Track- und Sektornummer des Blocks stehen.
f510   a5  3d       lda   $3d         Drivenummer für den Job
```

```
f512   0a           asl             mal 2
f513   aa           tax             als Index
f514   b5  12       lda   $12,x      ID 1 holen
f516   85  16       sta   $16        und übernehmen
f518   b5  13       lda   $13,x      ID 2 holen
f51a   85  17       sta   $17        und übernehmen
f51c   a0  00       ldy   #$00       Index in Jobspeicher
f51e   b1  32       lda   ($32),y    Tracknummer aus Jobspeicher holen
f520   85  18       sta   $18        und übernehmen
f522   c8           iny             Index auf Sektor
f523   b1  32       lda   ($32),y    Sektornummer aus Jobspeicher holen
f525   85  19       sta   $19        und übernehmen
f527   a9  00       lda   #$00       Startwert für Prüfsumme
f529   45  16       eor   $16        Prüfsumme über
f52b   45  17       eor   $17        den zusammengestellten
f52d   45  18       eor   $18        Blockheader
f52f   45  19       eor   $19        berechnen und
f531   85  1a       sta   $1a        abspeichern
f533   20  34  f9   jsr   $f934      Blockheader in GCR umwandeln
f536   a2  5a       ldx   #$5a       90 Leseversuche maximal setzen
f538   20  56  f5   jsr   $f556      SYNC-Signal abwarten
f53b   a0  00       ldy   #$00       Index auf Start
f53d   50  fe       bvc   $f53d      Byte einlesen
f53f   b8           clv             Flag löschen
f540   ad  01  1c   lda   $1c01       Byte vom Diskcontroller holen
f543   d9  24  00   cmp   $0024,y    und mit abgespeicherten GCR-Bytes vergleichen
f546   d0  06       bne   $f54e      verzweige, wenn ungleich
f548   c8           iny             Index erhöhen
f549   c0  08       cpy   #$08       schon alle Headerbytes geprüft?
f54b   d0  f0       bne   $f53d      weitermachen, wenn nein
f54d   60           rts             Ende; alles ok
f54e   ca           dex             Zähler für Leseversuche minus 1
f54f   d0  e7       bne   $f538      und wieder versuchen
f551   a9  02       lda   #$02       Nummer der Fehlermeldung
f553   4c  69  f9   jmp   $f969      '20, READ ERROR' ausgeben; Ende
--------------------------------------------------------------------------
f556                                Wartet ein SYNC-Signal auf Diskette ab.
f556   a9  d0       lda   #$d0       53 ms als maximale Suchzeit
f558   8d  05  18   sta   $1805      Timer starten
f55b   a9  03       lda   #$03       Nummer der Fehlermeldung setzen
f55d   2c  05  18   bit   $1805      Timer schon abgelaufen?
f560   10  f1       bpl   $f553      '21, READ ERROR', wenn ja
f562   2c  00  1c   bit   $1c00      SYNC-Signal erhalten?
f565   30  f6       bmi   $f55d      weitermachen, wenn nein
f567   ad  01  1c   lda   $1c01       Diskcontroller lesen; Port freimachen
f56a   b8           clv             Flag löschen
f56b   a0  00       ldy   #$00       Index löschen
f56d   60           rts             Ende
--------------------------------------------------------------------------
f56e                                Prüft auf Jobcode für Schreiben und schreibt ggf. einen
                                    Block aus dem aktuellen Puffer auf die Diskette.
f56e   c9  10       cmp   #$10       Jobcode für Block schreiben?
f570   f0  03       beq   $f575      verzweige, wenn ja
f572   4c  91  f6   jmp   $f691      Jobcode weiter prüfen
f575   20  e9  f5   jsr   $f5e9      Prüfsumme über Datenblock im Puffer berechnen
f578   85  3a       sta   $3a        und abspeichern
f57a   ad  00  1c   lda   $1c00      Kontrollport lesen
f57d   29  10       and   #$10       WRITE PROTECT isolieren
```

```
f57f   d0  05       bne   $f586       verzweige, wenn Lichtschranke frei
f581   a9  08       lda   #$08        sonst Nummer der Fehlermeldung
f583   4c  69  f9    jmp   $f969       '26, WRITE PROTECT' ausgeben
f586   20  8f  f7    jsr   $f78f       Pufferinhalt in GCR umwandeln
f589   20  10  f5    jsr   $f510       Blockheader suchen
f58c   a2  09       ldx   #$09        9 Bytes
f58e   50  fe       bvc   $f58e       Byte überlesen, um hinter den Header
f590   b8           clv               des Sektors zu kommen
f591   ca           dex               weitermachen
f592   d0  fa       bne   $f58e       und nächstes Byte überlesen
f594   a9  ff       lda   #$ff        Schreib-/Lesekopf auf Schreiben
f596   8d  03  1c   sta   $1c03       umschalten
f599   ad  0c  1c   lda   $1c0c       PCR laden und
f59c   29  1f       and   #$1f        auf Schreibbetrieb
f59e   09  c0       ora   #$c0        umschalten
f5a0   8d  0c  1c   sta   $1c0c       PCR wieder initialisieren
f5a3   a9  ff       lda   #$ff        $FF; Bytewert für SYNC-Markierung
f5a5   a2  05       ldx   #$05        5 Bytes
f5a7   8d  01  1c   sta   $1c01       zum Diskcontroller
f5aa   b8           clv               und
f5ab   50  fe       bvc   $f5ab       Schreibvorgang abwarten
f5ad   b8           clv               Flag wieder löschen
f5ae   ca           dex               Zähler auf nächstes Byte
f5af   d0  fa       bne   $f5ab       weitermachen
f5b1   a0  bb       ldy   #$bb        Index in Ausweichpuffer
f5b3   b9  00  01   lda   $0100,y     Byte aus Ausweichpuffer
f5b6   50  fe       bvc   $f5b6       nächsten Schreibvorgang abwarten
f5b8   b8           clv               Flag löschen
f5b9   8d  01  1c   sta   $1c01       Byte zum Diskcontroller
f5bc   c8           iny               nächstes Byte
f5bd   d0  f4       bne   $f5b3       und gesamten Ausweichpuffer schreiben
f5bf   b1  30       lda   ($30),y     Y=0; 'normalen' Puffer
f5c1   50  fe       bvc   $f5c1       Schreibvorgang abwarten
f5c3   b8           clv               Flag löschen
f5c4   8d  01  1c   sta   $1c01       und Byte zum Diskcontroller
f5c7   c8           iny               nächstes Byte aus Puffer
f5c8   d0  f5       bne   $f5bf       und weitermachen, bis alle Daten geschrieben
f5ca   50  fe       bvc   $f5ca       Schreib-Ende abwarten
f5cc   ad  0c  1c   lda   $1c0c       PCR laden und
f5cf   09  e0       ora   #$e0        und wieder auf Lesen
f5d1   8d  0c  1c   sta   $1c0c       umschalten
f5d4   a9  00       lda   #$00        Schreib-/Lesekopf wieder auf Lesen
f5d6   8d  03  1c   sta   $1c03       umschalten
f5d9   20  f2  f5   jsr   $f5f2       Pufferinhalt wieder in Binärbytes umwandeln
f5dc   a4  3f       ldy   $3f        Puffernummer für Job
f5de   b9  00  00   lda   $0000,y     Jobcode aus Puffer von 'Schreiben'
f5e1   49  30       eor   #$30        in VERIFY umwandeln
f5e3   99  00  00   sta   $0000,y     und wieder abspeichern
f5e6   4c  b1  f3   jmp   $f3b1       VERIFY ausführen
-------------------------------------------------------------------------
f5e9                                  Prüfsumme über Datenblock bilden und im Akku an
                                      übergeordnete Routine übergeben.
f5e9   a9  00       lda   #$00        Startwert für Prüfsumme
f5eb   a8           tay               Index laden
f5ec   51  30       eor   ($30),y     Pufferinhalt verknüpfen
f5ee   c8           iny               bis
f5ef   d0  fb       bne   $f5ec       gesamter Puffer abgefragt
f5f1   60           rts               Ende; Summe in A
```

```
--------------------------------------------------------------------------
f5f2                                    Routine wandelt den Inhalt des Ausweichpuffers und des
                                        aktuellen Datenpuffers aus dem GCR-(5 Bit)-Code wieder in
                                        den Binär-(4 Bit)-Code um und schreibt diese Werte dann in
                                        den aktuellen Puffer zurück.
f5f2   a9  00       lda   #$00          Pufferaddressen und Indizes löschen:
f5f4   85  2e       sta   $2e           Pufferadresse für Konvertierung Lc
f5f6   85  30       sta   $30           Pufferadresse Lo
f5f8   85  4f       sta   $4f           Zeiger in späteren Puffer für Umwandlung
f5fa   a5  31       lda   $31           Pufferadresse Hi
f5fc   85  4e       sta   $4e           Adresse des aktuellen Puffers merken
f5fe   a9  01       lda   #$01          Adresse des Ausweichpuffers Hi
f600   85  31       sta   $31           setzen
f602   85  2f       sta   $2f           setzen
f604   a9  bb       lda   #$bb          Index in Ausweichpuffer
f606   85  34       sta   $34           setzen
f608   85  36       sta   $36           setzen
f60a   20  e6  f7   jsr   $f7e6         5 GCR-Bytes in 4 Binärbytes umwandeln
f60d   a5  52       lda   $52           erstes umgewandeltes Byte
f60f   85  38       sta   $38           als Datenblockkennzeichen merken
f611   a4  36       ldy   $36           Index in Ausweichpuffer
f613   a5  53       lda   $53           zweites umgewandeltes Byte
f615   91  2e       sta   ($2e),y       in aktuellen Puffer schreiben
f617   c8           iny                 nächstes Byte
f618   a5  54       lda   $54           drittes umgewandeltes Byte
f61a   91  2e       sta   ($2e),y       in aktuellen Puffer schreiben
f61c   c8           iny                 nächstes Byte
f61d   a5  55       lda   $55           viertes umgewandeltes Byte
f61f   91  2e       sta   ($2e),y       in aktuellen Puffer schreiben
f621   c8           iny                 nächstes Byte
f622   84  36       sty   $36           Index merken
f624   20  e6  f7   jsr   $f7e6         5 GCR-Bytes in 4 Binärbytes umwandeln
f627   a4  36       ldy   $36           Index in Puffer holen
f629   a5  52       lda   $52           erstes umgewandeltes Byte
f62b   91  2e       sta   ($2e),y       in aktuellen Puffer schreiben
f62d   c8           iny                 nächstes Byte
f62e   a5  53       lda   $53           zweites umgewandeltes Byte
f630   91  2e       sta   ($2e),y       in aktuellen Puffer schreiben
f632   c8           iny                 nächstes Byte
f633   f0  0e       beq   $f643         verzweige, wenn Puffer schon voll
f635   a5  54       lda   $54           drittes umgewandeltes Byte
f637   91  2e       sta   ($2e),y       in aktuellen Puffer schreiben
f639   c8           iny                 nächstes Byte
f63a   a5  55       lda   $55           viertes umgewandeltes Byte
f63c   91  2e       sta   ($2e),y       in aktuellen Puffer schreiben
f63e   c8           iny                 nächstes Byte
f63f   84  36       sty   $36           Index merken
f641   d0  e1       bne   $f624         weitermachen, wenn Puffer noch nicht voll
f643   a5  54       lda   $54           drittes umgewandeltes Byte
f645   91  30       sta   ($30),y       in aktuellen Puffer schreiben
f647   c8           iny                 nächstes Byte
f648   a5  55       lda   $55           viertes umgewandeltes Byte
f64a   91  30       sta   ($30),y       in aktuellen Puffer schreiben
f64c   c8           iny                 nächstes Byte
f64d   84  36       sty   $36           Index merken
f64f   20  e6  f7   jsr   $f7e6         5 GCR-Bytes in 4 Binärbytes umwandeln
f652   a4  36       ldy   $36           Index holen
```

```
f654   a5  52       lda   $52         erstes umgewandeltes Byte
f656   91  30       sta   ($30),y     in aktuellen Puffer schreiben
f658   c8           iny               nächstes Byte
f659   a5  53       lda   $53         zweites umgewandeltes Byte
f65b   91  30       sta   ($30),y     in aktuellen Puffer schreiben
f65d   c8           iny               nächstes Byte
f65e   a5  54       lda   $54         drittes umgewandeltes Byte
f660   91  30       sta   ($30),y     in aktuellen Puffer schreiben
f662   c8           iny               nächstes Byte
f663   a5  55       lda   $55         viertes umgewandeltes Byte
f665   91  30       sta   ($30),y     in aktuellen Puffer schreiben
f667   c8           iny               nächstes Byte
f668   84  36       sty   $36         Index merken
f66a   c0  bb       cpy   #$bb        erster Teil schon erledigt?
f66c   90  e1       bcc   $f64f       weitermachen, bis erledigt
f66e   a9  45       lda   #$45        Pufferadresse Lo auf zweiten Teil
f670   85  2e       sta   $2e         setzen
f672   a5  31       lda   $31         Pufferadresse Hi
f674   85  2f       sta   $2f         setzen
f676   a0  ba       ldy   #$ba        Index auf erstes Byte im aktuellen Puffer
f678   b1  30       lda   ($30),y     Byte aus erstem Teil des aktuellen Puffers
f67a   91  2e       sta   ($2e),y     in zweiten Teil des aktuellen Puffers
f67c   88           dey               nächstes Byte
f67d   d0  f9       bne   $f678       verzweige, wenn noch nicht fertig
f67f   b1  30       lda   ($30),y     letztes Byte
f681   91  2e       sta   ($2e),y     ebenfalls übertragen
f683   a2  bb       ldx   #$bb        Index auf Ausweichpuffer
f685   bd  00  01    lda   $0100,x     Teil des Ausweichpuffers
f688   91  30       sta   ($30),y     vorne an die Bytes im aktuellen Puffer setzen
f68a   c8           iny               nächstes Byte im aktuellen Puffer
f68b   e8           inx               nächstes Byte aus Ausweichpuffer
f68c   d0  f7       bne   $f685       wei termachen
f68e   86  50       stx   $50         Flag für Pufferinhalt im Binärcode setzen; x=o
f690   60           rts               Ende
-------------------------------------------------------------------------
f691                                   Prüft auf den Jobcode für VERIFY und vergleicht ggf. die
                                       Daten im aktuellen Puffer nach GCR. Codierung direkt mit
                                       den entsprechenden Daten auf der Diskette.
f691   c9  20       cmp   #$20        Jobcode für VERIFY?
f693   f0  03       beq   $f698       verzweige, wenn ja
f695   4c  ca  f6    jmp   $f6ca       Jobcode weiter prüfen
f698   20  e9  f5    jsr   $f5e9       Prüfsumme über den Datenblock berechnen
f69b   85  3a       sta   $3a         und merken
f69d   20  8f  f7    jsr   $f78f       Datenblock in GCR umwandeln
f6a0   20  0a  f5    jsr   $f50a       Datenblockbeginn suchen
f6a3   a0  bb       ldy   #$bb        Index in Ausweichpuffer
f6a5   b9  00  01    lda   $0100,y     70 Bytes aus Ausweichpuffer
f6a8   50  fe       bvc   $f6a8       Byte lesen
f6aa   b8           clv               Flag löschen
f6ab   4d  01  1c    eor   $1c01       Byte aus Puffer mit Byte von DC vergleichen
f6ae   d0  15       bne   $f6c5       verzweige, wenn ungleich
f6b0   c8           iny               Index auf nächstes Byte
f6b1   d0  f2       bne   $f6a5       weitermachen
f6b3   b1  30       lda   ($30),y     nun die Daten aus dem aktuellen Puffer
f6b5   50  fe       bvc   $f6b5       Byte lesen
f6b7   b8           clv               Flag löschen?
f6b8   4d  01  1c    eor   $1c01       mit Daten von Diskette vergleichen
```

```
f6bb   d0  08       bne   $f6c5       verzweige, wenn ungleich
f6bd   c8           iny               nächstes Byte
f6be   c0  fd       cpy   #$fd        schon Ende des Puffers erreicht?
f6c0   d0  f1       bne   $f6b3       weitermachen, wenn nein
f6c2   4c  18  f4   jmp   $f418       Ende; alles ok
f6c5   a9  07       lda   #$07        Nummer der Fehlermeldung
f6c7   4c  69  f9   jmp   $f969       '25, WRITE ERROR' ausgeben; Ende
--------------------------------------------------------------------------
f6ca                                  Sucht nach einem Blockheader.
f6ca   20  10  f5   jsr   $f510       Blockheader auf Diskette suchen
f6cd   4c  18  f4   jmp   $f418       Ende; alles ok
--------------------------------------------------------------------------
f6d0                                  Wandelt 4 Binärbytes in den Speicherstellen $52 bis $55 in
                                      5 GCR-codierte Bytes um und schreibt diese in den aktuellen
                                      Puffer. Pufferzeiger dabei in $34.
f6d0   a9  00       lda   #$00        Rechenbereich
f6d2   85  57       sta   $57         löschen
f6d4   85  5a       sta   $5a         löschen
f6d6   a4  34       ldy   $34         Pufferzeiger holen
f6d8   a5  52       lda   $52         erstes Byte holen
f6da   29  f0       and   #$f0        Hi-Nibble isolieren
f6dc   4a           lsr               und die
f6dd   4a           lsr               4 Bits in
f6de   4a           lsr               die unteren 4
f6df   4a           lsr               Bitpositionen schieben
f6e0   aa           tax               Wert als Index
f6e1   bd  7f  f7   lda   $f77f,x     5-Bit-Nibble als GCR-Äquivalent holen
f6e4   0a           asl               5 Bits im Byte
f6e5   0a           asl               an die oberen
f6e6   0a           asl               Positionen (Bits 3 bis 7) schieben
f6e7   85  56       sta   $56         und diesen Wert merken
f6e9   a5  52       lda   $52         jetzt das Ausgangsbyte holen
f6eb   29  0f       and   #$0f        und das Lo-Nibble
f6ed   aa           tax               entsprechend
f6ee   bd  7f  f7   lda   $f77f,x     umwandeln
f6f1   6a           ror               die zwei
f6f2   66  57       ror   $57         untersten Bits in
f6f4   6a           ror               den Zwischenspeicher
f6f5   66  57       ror   $57         an oberste Position (6,7) bringen
f6f7   29  07       and   #$07        Restwert im Akku auf 3 Bits Länge begrenzen
f6f9   05  56       ora   $56         und an Hi-Nibble anhängen; 1. GCR-Byte fertig
f6fb   91  30       sta   ($30),y     Byte in Puffer schreiben
f6fd   c8           iny               Zeiger auf nächstes Byte
f6fe   a5  53       lda   $53         zweites Byte für Umwandlung holen
f700   29  f0       and   #$f0        und Hi-Nibble isolieren
f702   4a           lsr               ebenfalls das Hi-Nibble
f703   4a           lsr               an die
f704   4a           lsr               untersten 4 Positionen im Byte
f705   4a           lsr               schieben
f706   aa           tax               Wert als Index
f707   bd  7f  f7   lda   $f77f,x     5 Bit GCR-Äquivalent holen
f70a   0a           asl               5 Bits auf Positionen 1 bis 5 bringen
f70b   05  57       ora   $57         und mit 2 Bits des vorherigen Bytes summieren
f70d   85  57       sta   $57         Wert ablegen; es fehlt jetzt noch Bit 0
f70f   a5  53       lda   $53         zweites Byte für Umwandlung nochmals holen
f711   29  0f       and   #$0f        und Lo-Nibble jetzt isolieren
f713   aa           tax               Wert wiederum
```

```
f714   bd  7f  f7    lda   $f77f,x     in 5 Bit GCR-Äqivalent umwandeln
f717   2a            rol               und im Byte
f718   2a            rol               an die vier (!!) höchsten Stellen
f719   2a            rol               schieben (4 bis 7); das höchstwertige Bit
f71a   2a            rol               landet dabei im Carry-Flag
f71b   85  58        sta   $58         vier GCR-Bits merken
f71d   2a            rol               höchstes Bit aus Carry wieder in Akku
f71e   29  01        and   #$01        und auf Position 0 schieben
f720   05  57        ora   $57         dieses Bit zu den 6 vorigen GCR-Bits summieren
f722   91  30        sta   ($30),y     und dieses zweite GCR-Byte in den Puffer
f724   c8            iny               Pufferzeiger erhöhen
f725   a5  54        lda   $54         drittes Byte für Umwandlung holen
f727   29  f0        and   #$f0        und Hi-Nibble isolieren
f729   4a            lsr               4 Bits wieder
f72a   4a            lsr               an die vier untersten
f72b   4a            lsr               Positionen (0 bis 3)
f72c   4a            lsr               schieben und
f72d   aa            tax               den Binärwert in das entsprechende
f72e   bd  7f  f7    lda   $f77f,x     5 Bit GCR-Äquivalent umwandeln
f731   18            clc               Carry-Bit löschen
f732   6a            ror               Bit 0 des GCR-Bytes ins Carry
f733   05  58        ora   $58         die vier Bits mit den vorigen Bits summieren
f735   91  30        sta   ($30),y     und das dritte GCR-Byte in den Puffer schreiben
f737   c8            iny               Pufferzeiger erhöhen
f738   6a            ror               Bit aus Carry-Flag wieder in Akku (Position 7)
f739   29  80        and   #$80        und restliche Bits eliminieren
f73b   85  59        sta   $59         neuen Byteteil zwischenspeichern
f73d   a5  54        lda   $54         und drittes Byte für Umwandlung wieder holen
f73f   29  0f        and   #$0f        diesmal das lo-Nibble isolieren
f741   aa            tax               und
f742   bd  7f  f7    lda   $f77f,x     in 5 Bit GCR-Äquivalent umwandeln
f745   0a            asl               die 5 Bits auf die Positionen 2 bis 6
f746   0a            asl               schieben
f747   29  7c        and   #$7c        und restliche Bits eliminieren
f749   05  59        ora   $59         Bit 7 des letzten Teils summieren und
f74b   85  59        sta   $59         Wert merken; jetzt fehlen noch Bit 0 und 1
f74d   a5  55        lda   $55         viertes Byte für Umwandlung holen
f74f   29  f0        and   #$f0        und Hi-Nibble isolieren
f751   4a            lsr               dann die 4 Bits wieder
f752   4a            lsr               an die Positionen
f753   4a            lsr               0 bis 3 im Byte
f754   4a            lsr               schieben
f755   aa            tax               und
f756   bd  7f  f7    lda   $f77f,x     in 5 Bit GCR-Äquivalent umwandeln
f759   6a            ror               Bit 0 ins Carry-Flag schieben
f75a   66  5a        ror   $5a         und danach an Position 7 in Zwischenspeicher
f75c   6a            ror               Bit 1 ins Carry-Flag und
f75d   66  5a        ror   $5a         danach an Position 7 in Zwischenspeicher
f75f   6a            ror               Bit 2 ins Carry und
f760   66  5a        ror   $5a         als Bit 7 zu Bit 5 und 6 des Zwischenspeichers
f762   29  03        and   #$03        restliche 2 Bits im Akku isolieren
f764   05  59        ora   $59         und zu den vorigen 6 Bits summieren
f766   91  30        sta   ($30),y     viertes GCR-Byte in Puffer schreiben
f768   c8            iny               Pufferzeiger erhöhen
f769   d0  04        bne   $f76f       weiter, wenn Puffer noch nicht voll
f76b   a5  2f        lda   $2f         sonst auf aktuellen Puffer
f76d   85  31        sta   $31         umschalten; Ausweichpuffer voll
f76f   a5  55        lda   $55         viertes Byte für Umwandlung wieder holen
```

```
f771   29  0f       and   #$0f        und Lo-Nibble isolieren
f773   aa           tax               und
f774   bd  7f  f7   lda   $f77f,x     in 5 Bit GCR-Äquivalent umwandeln
f777   05  5a       ora   $5a         mit vorigen 3 Bits summieren und
f779   91  30       sta   ($30),y     fünftes GCR-Byte in den Puffer schreiben
f77b   c8           iny               Pufferzeiger erhöhen
f77c   84  34       sty   $34         und merken
f77e   60           rts               Ende
--------------------------------------------------------------------------
f77f                                  GCR-Äquivalente für die Umwandlung eines 4 Bit Nibbles in
                                      ein 5 Bit GCR-Nibble.
f77f 0a = 01010  für %0000            $00
f780 0b = 01011  für %0001            $01
f781 12 = 10010  für %0010            $02
f782 13 = 10011  für %0011            $03
f783 0e = 01110  für %0100            $04
f784 0f = 01111  für %0101            $05
f785 16 = 10110  für %0110            $06
f786 17 = 10111  für %0111            $07
f787 09 = 01001  für %1000            $08
f788 19 = 11001  für %1001            $09
f789 1a = 11010  für %1010            $0a
f78a 1b = 11011  für %1011            $0b
f78b 0d = 01101  für %1100            $0c
f78c 1d = 11101  für %1101            $0d
f78d 1e = 11110  für %1110            $0e
f78e 15 = 10101  für %1111            $0f
--------------------------------------------------------------------------
f78f                                  Wandelt den gesamten aktiven Puffer von Binär (4 Bit)-
                                      Werten in GCR-(5 Bit)-Werte um. Der entstehende Überschuß
                                      an Bytes wird dabei im Ausweichpuffer von $01bb bis $01ff
                                      abgelegt.
f78f   a9  00       lda   #$00        Indizes und Adressen Lo löschen:
f791   85  30       sta   $30         Pufferadresse Lo
f793   85  2e       sta   $2e         Pufferadresse Lo für umwandlung
f795   85  36       sta   $36         Pufferzeiger
f797   a9  bb       lda   #$bb        Zeiger auf Ausweichpuffer
f799   85  34       sta   $34         setzen
f79b   85  50       sta   $50         setzen
f79d   a5  31       lda   $31         Pufferadresse Hi
f79f   85  2f       sta   $2f         kopieren
f7a1   a9  01       lda   #$01        Pufferadresse Hi
f7a3   85  31       sta   $31         auf Ausweichpuffer stellen
f7a5   a5  47       lda   $47         Kennzeichen $07 für Datenblock
f7a7   85  52       sta   $52         als erstes Byte für Umwandlung setzen
f7a9   a4  36       ldy   $36         Pufferzeiger holen
f7ab   b1  2e       lda   ($2e),y     Byte aus Puffer
f7ad   85  53       sta   $53         für Umwandlung setzen
f7af   c8           iny               Pufferzeiger erhöhen
f7b0   b1  2e       lda   ($2e),y     Byte aus Puffer
f7b2   85  54       sta   $54         für Umwandlung setzen
f7b4   c8           iny               Pufferzeiger erhöhen
f7b5   b1  2e       lda   ($2e),y     Byte aus Puffer
f7b7   85  55       sta   $55         für Umwandlung setzen
f7b9   c8           iny               Pufferzeiger erhöhen
f7ba   84  36       sty   $36         und merken
f7bc   20  d0  f6   jsr   $f6d0       4 Bytes umwandeln und in Puffer schreiben
f7bf   a4  36       ldy   $36         Pufferzeiger holen
```

```
f7c1   b1  2e       lda   ($2e),y     Byte aus Puffer holen
f7c3   85  52       sta   $52         erstes Byte für Umwandlung
f7c5   c8           iny               Pufferzejger erhöhen
f7c6   f0  11       beq   $f7d9       verzweige, wenn ganzer Puffer voll
f7c8   b1  2e       lda   ($2e),y     Byte aus Puffer holen
f7ca   85  53       sta   $53         zweites Byte für Umwandlung
f7cc   c8           iny               Pufferzeiger erhöhen
f7cd   b1  2e       lda   ($2e),y     Byte as Puffer holen
f7cf   85  54       sta   $54         drittes Byte für Umwandlung
f7d1   c8           iny               Pufferzeiger erhöhen
f7d2   b1  2e       lda   ($2e),y     Byte aus Puffer holen
f7d4   85  55       sta   $55         viertes Byte für Umwandlung
f7d6   c8           iny               Pufferzeiger erhöhen
f7d7   d0  e1       bne   $f7ba       weiter umwandeln, bis Puffer voll
f7d9   a5  3a       lda   $3a         Prüfsumme über Datenblock
f7db   85  53       sta   $53         als zweites Byte für Umwandlung
f7dd   a9  00       lda   #$00        $00
f7df   85  54       sta   $54         als drittes und
f7e1   85  55       sta   $55         als viertes Byte für Umwandlung
f7e3   4c  d0  f6   jmp   $f6d0       4 Bytes umwandeln und in Puffer schreiben; Ende
-------------------------------------------------------------------------------
f7e6                                  Wandelt 5 GCR-codierte Werte aus dem Puffer in 4 Binärwerte
                                      um und speichert diese dann nach $52 bis $55. Pufferzeiger
                                      in $34.
f7e6   a4  34       ldy   $34         Pufferzeiger holen
f7e8   b1  30       lda   ($30),y     Byte aus Puffer
f7ea   29  f8       and   #$f8        die 5 höchsten Bits isolieren
f7ec   4a           lsr               und an die
f7ed   4a           lsr               5 niedrigsten Positionen (0 bis 4)
f7ee   4a           lsr               schieben
f7ef   85  56       sta   $56         ert merken (1. GCR-Nibble)
f7f1   b1  30       lda   ($30),y     Byte nochmals holen
f7f3   29  07       and   #$07        und die drei niederwertigen Bits isolieren
f7f5   0a           asl               und auf die Positionen
f7f6   0a           asl               2 bis 4 schieben
f7f7   85  57       sta   $57         ert merken; es fehlen noch die Bits 0 und 1
f7f9   c8           iny               Pufferzeiger erhöhen
f7fa   d0  06       bne   $f802       verzweige, wenn noch Bytes im Puffer
f7fc   a5  4e       lda   $4e         sonst Adresse Hi für Ausweichpuffer
f7fe   85  31       sta   $31         setzen
f800   a4  4f       ldy   $4f         und Adresse Lo als Index nehmen
f802   b1  30       lda   ($30),y     zweites GCR-Byte aus Puffer holen
f804   29  c0       and   #$c0        die zwei höchstwertigen Bits (6,7) isolieren
f806   2a           rol               und an die
f807   2a           rol               Positionen 0 und 1
f808   2a           rol               schieben
f809   05  57       ora   $57         zu den drei restlichen Bits summieren
f80b   85  57       sta   $57         und abspeichern (2. GCR-Nibble)
f80d   b1  30       lda   ($30),y     zweites Byte noch einmal holen
f80f   29  3e       and   #$3e        die Bits 1 bis 5 isolieren
f811   4a           lsr               und an die Positionen 0 bis 4 schieben
f812   85  58       sta   $58         5 Bits merken (3. GCR-Nibble)
f814   b1  30       lda   ($30),y     zweites Byte wiederum holen
f816   29  01       and   #$01        Bit 0 isolieren
f818   0a           asl               und an
f819   0a           asl               die Position
f81a   0a           asl               4 im Byte
f81b   0a           asl               schieben
```

```
f81c   85  59       sta   $59         Bit merken
f81e   c8           iny               Pufferzeiger auf nächstes Byte
f81f   b1  30       lda   ($30),y     drittes GCR-Byte aus Puffer
f821   29  f0       and   #$f0        und die vier obersten Bits isolieren
f823   4a           lsr               diese 4 Bits
f824   4a           lsr               an die
f825   4a           lsr               untersten vier Positionen (0-3)
f826   4a           lsr               schieben
f827   05  59       ora   $59         mit Bit 4 verknüpfen
f829   85  59       sta   $59         und 5 Bit-Nibble abspeichern (4. GCR-Nibble)
f82b   b1  30       lda   ($30),y     drittes GCR-Byte noch einmal aus Puffer
f82d   29  0f       and   #$0f        und Lo-Nibble isolieren
f82f   0a           asl               Bits an die Positionen 1 bis 4 schieben
f830   85  5a       sta   $5a         Wert merken
f832   c8           iny               Pufferzeiger auf nächstes Byte
f833   b1  30       lda   ($30),y     viertes GCR-Byte aus Puffer
f835   29  80       and   #$80        und Bit 7 isolieren
f837   18           clc               Carry löschen
f838   2a           rol               und Bit 7 über Carry
f839   2a           rol               auf Position 0 rollen
f83a   29  01       and   #$01        Bit 0 isolieren
f83c   05  5a       ora   $5a         und zu den vorigen 4 Bits hinzufügen
f83e   85  5a       sta   $5a         Wert wieder merken (5. GCR-Nibble)
f840   b1  30       lda   ($30),y     viertes GCR-Byte nochmal aus Puffer holen
f842   29  7c       and   #$7c        Bits 2 bis 6 isolieren
f844   4a           lsr               und an die
f845   4a           lsr               Positionen 0 bis 4 schieben
f846   85  5b       sta   $5b         Wert merken (6- GCR-Nibble)
f848   b1  30       lda   ($30),y     viertes GCR-Byte wieder holen
f84a   29  03       and   #$03        Bits 0 und 1 isolieren
f84c   0a           asl               und an die
f84d   0a           asl               Positionen 3 und 4
f84e   0a           asl               schieben
f84f   85  5c       sta   $5c         Bits merken
f851   c8           iny               Pufferzeiger auf nächstes Byte
f852   d0  06       bne   $f85a       verzweige, wenn Puffer noch nicht fertig
f854   a5  4e       lda   $4e         Pufferadresse Hi des Ausweichpuffers
f856   85  31       sta   $31         setzen
f858   a4  4f       ldy   $4f         Pufferadresse La des Ausweichpuffers als Index
f85a   b1  30       lda   ($30),y     fünftes GCR-Byte aus Puffer holen
f85c   29  e0       and   #$e0        die obersten 3 Bits isolieren
f85e   2a           rol               und an die
f85f   2a           rol               untersten 3
f860   2a           rol               Positionen 0 bis 2
f861   2a           rol               schieben
f862   05  5c       ora   $5c         die zwei vorigen Bits dazunehmen
f864   85  5c       sta   $5c         und Wert merken (7. GCR-Nibble)
f866   b1  30       lda   ($30),y     fünftes GCR-Byte noch einmal aus Puffer
f868   29  1f       and   #$1f        und die fünf untersten Bits isolieren
f86a   85  5d       sta   $5d         und merken (8. GCR-Nibble)
f86c   c8           iny               Pufferzeiger erhöhen
f86d   84  34       sty   $34         und merken
f86f   a6  56       ldx   $56         1. GCR-Nibble
f871   bd  a0  f8   lda   $f8a0,x     Binär-Äquivalent Hi holen
f874   a6  57       ldx   $57         2. GCR-Nibble
f876   1d  c0  f8   ora   $f8c0,x     und Binär-Äquivalent Lo addieren
f879   85  52       sta   $52         1. Binärbyte merken
f87b   a6  58       ldx   $58         3. GCR-Nibble
```

```
f87d   bd  a0  f8    lda   $f8a0,x     Binär-Äquivalent Hi holen
f880   a6  59        ldx   $59         4. GCR-Nibb,e
f882   1d  c0  f8    ora   $f8c0,x     und Binär-Äquivalent Lo addieren
f885   85  53        sta   $53         2. Binärbyte merken
f887   a6  5a        ldx   $5a         5. GCR-Nibble
f889   bd  a0  f8    lda   $f8a0,x     Binär-Äquivalent Hi holen
f88c   a6  5b        ldx   $5b         6. GCR-Nibble
f88e   1d  c0  f8    ora   $f8c0,x     und Binär-Äquivalent Lo addieren
f891   85  54        sta   $54         3. Binärbyte merken
f893   a6  5c        ldx   $5c         7. GCR-Nibble
f895   bd  a0  f8    lda   $f8a0,x     Binär-Äquivalent Hi holen
f898   a6  5d        ldx   $5d         8. GCR-Nibble
f89a   1d  c0  f8    ora   $f8c0,x     und Binär-Äquivalent Lo addieren
f89d   85  55        sta   $55         4. Binärbyte merken
f89f   60            rts               Ende
-------------------------------------------------------------------------
f8a0                                   Höherwertige Nibbles für die Umwandlung von von GCR-(5
                                       Bit)-Werten in Binär-(4 Bit)-Werte. Das GCR-Nibble dient
                                       dabei als Index und erreicht theoretische Werte von 0 bis
                                       31. An den Stellen mit $ff in der Tabelle kann keine
                                       Umwandlung erfolgen, da der GCR-Wert illegal ist.
f8a0 ff ff ff ff ff ff ff ff
f8a8 ff 80 00 10 ff c0 40 50
f8b0 ff ff 20 30 ff fO 60 70
f8b8 ff 90 a0 b0 ff d0 e0 ff
-------------------------------------------------------------------------
f8c0                                   Gleiche Tabelle, wie $f8a0. Nur handelt es sich hier um die
                                       niederwertigen Nibbles.
f8c0 ff ff ff ff ff ff ff ff
f8c8 ff 08 00 01 ff 0c 04 05
f8d0 ff ff 02 03 ff 0f 06 07
f8d8 ff 09 0a 0b ff 0d 0e ff
-------------------------------------------------------------------------
f8e0                                   Wandelt die GCR-(5 Bit)-Werte aus dem Ausweichpuffer $0100
                                       bis $01ff in Binär-(4 Bit)-Werte um und legt diese im
                                       aktuellen Puffer ab.
f8e0   a9  00        lda   #$00        Indizes und Adressen löschen:
f8e2   85  34        sta   $34         Pufferzeiger
f8e4   85  2e        sta   $2e         Pufferadresse Lo
f8e6   85  36        sta   $36         Pufferzeiger
f8e8   a9  01        lda   #$01        Adresse für Ausweichpuffer Hi
f8ea   85  4e        sta   $4e         setzen
f8ec   a9  ba        lda   #$ba        Adresse Lo (ergibt $01ba)
f8ee   85  4f        sta   $4f         ebenfalls setzen
f8f0   a5  31        lda   $31         Pufferadresse Hi
f8f2   85  2f        sta   $2f         kopieren
f8f4   20  e6  f7    jsr   $f7e6       5 GCR-Bytes aus Puffer in 4 Binärbytes wandeln
f8f7   a5  52        lda   $52         erstes Binärbyte
f8f9   85  38        sta   $38         als Kennzeichen für Datenblock setzen
f8fb   a4  36        ldy   $36         Pufferzeiger holen
f8fd   a5  53        lda   $53         zweites Binärbyte
f8ff   91  2e        sta   ($2e),y     in Puffer schreiben
f901   c8            iny               Pufferzeiger erhöhen
f902   a5  54        lda   $54         drittes Binärbyte
f904   91  2e        sta   ($2e),y     in Puffer schreiben
f906   c8            iny               Pufferzeiger erhöhen
```

```
f907   a5  55       lda   $55         viertes Binärbyte
f909   91  2e       sta   ($2e),y     in Puffer schreiben
f90b   c8           iny               Pufferzeiger erhöhen
f90c   84  36       sty   $36         merken
f90e   20  e6  f7   jsr   $f7e6       5 GCR-Bytes in 4 Binärbytes umwandeln
f911   a4  36       ldy   $36         Pufferzeiger holen
f913   a5  52       lda   $52         erstes Binärbyte
f915   91  2e       sta   ($2e),y     in Puffer schreiben
f917   c8           iny               Pufferzeiger erhöhen
f918   f0  11       beq   $f92b       verzweige, wenn Puffer fertig
f91a   a5  53       lda   $53         zweites Binärbyte
f91c   91  2e       sta   ($2e),y     in Puffer schreiben
f91e   c8           iny               Pufferzeiger erhöhen
f91f   a5  54       lda   $54         drittes Binärbyte
f921   91  2e       sta   ($2e),y     in Puffer schreiben
f923   c8           iny               Pufferzeiger erhöhen
f924   a5  55       lda   $55         viertes Binärbyte
f926   91  2e       sta   ($2e),y     in Puffer schreiben
f928   c8           iny               Pufferzeiger erhöhen
f929   d0  e1       bne   $f90c       verzweige, wenn Puffer noch nicht voll
f92b   a5  53       lda   $53         zweites Binärbyte
f92d   85  3a       sta   $3a         als Prüfsumme setzen
f92f   a5  2f       lda   $2f         Pufferadresse Hi
f931   85  31       sta   $31         wieder zurücksetzen
f933   60           rts               Ende
--------------------------------------------------------------------------------
f934                                  Routine konvertiert die Bytes des aktuellen Blockheaders in
                                      GCR-codierte Bytes und legt diese ab $24 in der Zeropage
                                      ab.
f934   a5  31       lda   $31         Pufferadresse Hi
f936   85  2f       sta   $2f         setzen
f938   a9  00       lda   #$00        Pufferadresse Hi auf Zeropage
f93a   85  31       sta   $31         setzen
f93c   a9  24       lda   #$24        Pufferzeiger auf $24
f93e   85  34       sta   $34         setzen
f940   a5  39       lda   $39         Konstante $08; Kennzeichen für Blockheader
f942   85  52       sta   $52         als erstes Binärbyte für Umwandlung
f944   a5  1a       lda   $1a         Prüfsumme des Blockheaders
f946   85  53       sta   $53         als zweites Binärbyte für Umwandlung
f948   a5  19       lda   $19         Sektornummer des Blocks
f94a   85  54       sta   $54         als drittes Byte für Umwandlung
f94c   a5  18       lda   $18         Tracknummer des Blocks
f94e   85  55       sta   $55         als viertes Byte für Umwandlung
f950   20  d0  f6   jsr   $f6d0       4 Binärbytes in 5 GCR-Bytes umwandeln
f953   a5  17       lda   $17         ID 2 des Blockheaders
f955   85  52       sta   $52         als erstes Byte für Umwandlung
f957   a5  16       lda   $16         ID 1 des Blockheaders
f959   85  53       sta   $53         als zweites Byte für Umwandlung
f95b   a9  00       lda   #$00        Null als Füllwert für
f95d   85  54       sta   $54         Byte drei und
f95f   85  55       sta   $55         Byte vier
f961   20  d0  f6   jsr   $f6d0       4 Binärbytes in 5 GCR-Bytes umwandeln
f964   a5  2f       lda   $2f         Pufferadresse Hi
f966   85  31       sta   $31         wieder setzen
f968   60           rts               Ende
```

```
------------------------------------------------------------------------
f969                                    Ausgang aus der Jobschleife mit Übergabe der Fehlernummer
                                        in A.
f969    a4  3f      ldy   $3f          Puffernummer für Job
f96b    99  00  00  sta   $0000,y      Rückmeldung in Jobspeicher
f96e    a5  50      lda   $50          noch GCR-Bytes vorhanden?
f970    f0  03      beq   $f975        verzweige, wenn nein
f972    20  f2  f5  jsr   $f5f2        sonst restliche GCR-Bytes in Binär umwandeln
f975    20  8f  f9  jsr   $f98f        Zähler für Ausschaltverzögerung des Drivemotors setzen, um
                                        das Drive nach einer Nachlaufzeit abzuschalten
f978    a6  49      ldx   $49          Stackpointer zurückholen
f97a    9a          txs                und setzen
f97b    4c  be  f2  jmp   $f2be        zurück zur Jobabfrage
------------------------------------------------------------------------
f97e                                    Laufwerksmotor einschalten. Nach dem Einschalten wird noch
                                        0.4 sekunden gewartet, bis das Laufwerk für den Betrieb
                                        freigegeben wird, da der Motor eine Hochlaufzeit benötigt.
                                        Die Be:riebsbereitschaft des Laufwerks wird durch Bit 7 in
                                        $20 signalisiert (Bit 7 dann 0).
f97e    a9  a0      lda   #$a0         Flag fur Motor im Anlaufen aber noch nicht
f980    85  20      sta   $20          auf Endgeschwindigkeit setzen
f982    ad  00  1c  lda   $1c00        Steuerport des Diskcontrollers
f985    09  04      ora   #$04         Bit für Motor setzen
f987    8d  00  1c  sta   $1c00        Motor anschalten
*1 f98a a9  32      lda   #$32         Zähler für Anlaufverzögerung des Motors
*0 f98a a9  7d      lda   #$7d         Zähler für Anlaufverzögerung des Motors
f98c    85  48      sta   $48          auf 0.4 Sekunden (50 IRQs) setzen
f98e    60          rts                Ende
------------------------------------------------------------------------
f98f                                    Laufwerksmotor ausschalten, nachdem ein Zähler für die
                                        Ausschaltverzögerung in $48 und $35 die 12.3 Sekunden
                                        dauernde Verzögerung heruntergezählt hat.
f98f    a6  3e      ldx   $3e          Drivenummer für Jobschleife
f991    a5  20      lda   $20          Flag für Motor im Ausschaltmodus
f993    09  10      ora   #$10         setzen, um
f995    85  20      sta   $20          Motor nach Verzögerung abzuschalten
f997    4c  2b  a6  jmp   $a62b        Verzögerungszähler setzen; Ende
f99a    ea          nop                unbenutzter Platzrest aus den
f99b    ea          nop                1541 ROMs
------------------------------------------------------------------------
f99c                                    Diese Routine ist die zentrale Steuerroutine des
                                        Diskcontrollers im DOS (1541-Modus). Sie existiert noch
                                        einmal für den 1571-Modus und ist für die Laufwerks- und
                                        Steppermotorsteuerung verantwortlich. Sie bildet den
                                        jeweiligen IRQ-Abschluß nach Durchlauf der Jobschleife und
                                        wird alle 8 ms aufgerufen.
f99c    ad  07  1c  lda   $1c07        Timer neu setzen
f99f    8d  05  1c  sta   $1c05        IRQ-Status löschen
f9a2    ad  00  1c  lda   $1c00        anhand der Schreibschutzlichtschranke auf
f9a5    29  10      and   #$10         einen eventuellen Diskettenwechsel testen
f9a7    c5  1e      cmp   $1e          hat Wechsel stattgefunden?
f9a9    85  1e      sta   $1e          neuen Status merken
f9ab    4c  34  a6  jmp   $a634        Flags setzen; Diskette zentrieren; Rücksprung
f9ae    ea          nop                unbenutzter Speicherplatz, der
```

```
f9af   ea           nop             aus den alten 1541 ROMs
f9b0   ea           nop             übriggeblieben ist
f9b1   ad  fe  02   lda   $02fe      Flag für Kopftransport
f9b4   f0  15       beq   $f9cb     verzweige, wenn Kopf auf Track
f9b6   c9  02       cmp   #$02      Kopf gerade auf Track positioniert?
f9b8   d0  07       bne   $f9c1     verzweige, wenn nein
f9ba   a9  00       lda   #$00      Flag für 'Kopf auf Track'
f9bc   8d  fe  02   sta   $02fe     setzen
f9bf   f0  0a       beq   $f9cb     unbedingter Sprung
f9c1   85  4a       sta   $4a       Kopf steht auf Halbspur; also muß Kopf einen
f9c3   a9  02       lda   #$02      halben Track bewegt und
f9c5   8d  fe  02   sta   $02fe     Flag für 'Kopf gerade positioniert'
f9c8   4c  2e  fa   jmp   $fa2e     gesetzt werden
f9cb   a6  3e       ldx   $3e       Drivenummer für Jobschleife
f9cd   30  07       bmi   $f9d6     verzweige, wenn Drive inaktiv
f9cf   a5  20       lda   $20       Flags für Drivestatus
f9d1   a8           tay             merken
f9d2   c9  20       cmp   #$20      Drivemotor an und bereit?
f9d4   d0  03       bne   $f9d9     verzweige, wenn nein
f9d6   4c  be  fa   jmp   $fabe     zurück aus IRQ-Programm
f9d9   c6  48       dec   $48       Verzögerungszähler für Drivemotor vermindern
f9db   d0  1d       bne   $f9fa     verzweige, wenn ungleich 0
f9dd   98           tya             Flags für Drivestatus zurückholen
f9de   10  04       bpl   $f9e4     verzweige, wenn Drive bereit
f9e0   29  7f       and   #$7f      sonst Flag für Drive nicht bereit
f9e2   85  20       sta   $20       löschen
f9e4   29  10       and   #$10      Motor in Ausschaltphase?
f9e6   f0  12       beq   $f9fa     verzweige, wenn nein
f9e8   c6  35       dec   $35       Verzögerung Hi für Ausschaltphase vermindern
f9ea   d0  0e       bne   $f9fa     verzweige, wenn ungleich 0
f9ec   ea           nop             Rest aus 1541-ROM
f9ed   20  70  87   jsr   $8770     Drivemotor abschalten
f9f0   a9  ff       lda   #$ff      Flag für aktives Drive
f9f2   85  3e       sta   $3e       löschen
f9f4   a9  00       lda   #$00      Flags für Drivestatus
f9f6   85  20       sta   $20       löschen
f9f8   f0  dc       beq   $f9d6     unbedingter Sprung
f9fa   98           tya             Flags für Drivestatus zurückholen
f9fb   29  40       and   #$40      soll Steppermotor aktiviert werden?
f9fd   d0  03       bne   $fa02     verzweige, wenn ja
f9ff   4c  be  fa   jmp   $fabe     zurück aus IRQ-Programm
fa02   6c  62  00   jmp   ($0062)   Sprung auf aktuelle Steppermotorroutine:
                                    $fa05 - Initialisierung des Steppermodus
                                    $fa3b - kurzer Steppermodus
                                    $fa4e - Beenden des Steppermodus
                                    $fa7b - Anfahren des Steppermotors
                                    $fa97 - schneller Steppermodus
                                    $faa5 - Abbremsen des Motors
--------------------------------------------------------------------------
fa05                                Initialisierung der Steppermotorsteuerung.
fa05   a5  4a       lda   $4a       Anzahl der Stepperschritte (2 pro Track)
fa07   10  05       bpl   $fa0e     verzweige, wenn Bewegung nach innen (Track 35)
fa09   49  ff       eor   #$ff      Komplement und Absolutwert
fa0b   18           clc             der Stepperschritte
fa0c   69  01       adc   #$01      bilden
fa0e   c5  64       cmp   $64       Distanz größer als $c8 (200) Schritte?
fa10   b0  0a       bcs   $fa1c     wenn ja, dann schneller Steppermodus
fa12   a9  3b       lda   #$3b      sonst Adresse
```

```
fa14   85  62       sta   $62        $fa3b für
fa16   a9  fa       lda   #$fa       langsamen Steppermodus
fa18   85  63       sta   $63        setzen
fa1a   d0  12       bne   $fa2e      unbedingter Sprung
fa1c   e5  5e       sbc   $5e        Schritte für Schnellmodus berechnen;
fa1e   e5  5e       sbc   $5e        dafür Anzahl der Anfahr- und Bremsschritte
fa20   85  61       sta   $61        (jeweils 4) von den gesamten abziehen
fa22   a5  5e       lda   $5e        Zahl der Anfahrschritte
fa24   85  60       sta   $60        setzen
fa26   a9  7b       lda   #$7b       Adresse $fa7b
fa28   85  62       sta   $62        für schnellen
fa2a   a9  fa       lda   #$fa       Stepprmodus
fa2c   85  63       sta   $63        setzen
fa2e   a5  4a       lda   $4a        Kopfbewegung nach innen?
fa30   10  31       bpl   $fa63      Kopf 1 Step nach innen bewegen, wenn ja
fa32   4c  45  ff   jmp   $ff45      Nullanschlag prüfen; Bewegung nach außen
fa35   ea           nop              unbenutzter Speicherrest
fa36   ea           nop              durch Abänderung des
fa37   ea           nop              1541 ROM
fa38   4c  69  fa   jmp   $fa69      Kopfbewegung 1 Step nach außen
-------------------------------------------------------------------------------
fa3b                                 Langsamer Steppermodus für kurze Wege beim
                                     Kopfpositionieren.
fa3b   a5  4a       lda   $4a        Anzahl der zu fahrenden Schritte
fa3d   d0  ef       bne   $fa2e      verzweige, wenn noch nicht Null
fa3f   a9  4e       lda   #$4e       Adresse $fa4e für
fa41   85  62       sta   $62        Beenden der
fa43   a9  fa       lda   #$fa       Kopfpositionierung
fa45   85  63       sta   $63        setzen
fa47   a9  05       lda   #$05       5 Steps für Anhalten des
fa49   85  60       sta   $60        Kopfes setzen
fa4b   4c  be  fa   jmp   $fabe      zurück aus IRQ-Programm
-------------------------------------------------------------------------------
fa4e                                 Beendigung der Kopfpositionierung; Zurücksetzen der Flags
                                     für die Kopfpositionierung.
fa4e   c6  60       dec   $60        Zähler zum Abbremsen des Motors minus 1
fa50   d0  6c       bne   $fabe      verzweige, wenn noch nicht fertig
fa52   a5  20       lda   $20        Flags für Drivestatus
fa54   29  bf       and   #$bf       Bit 6 löschen, da Steppermotor nicht mehr in
fa56   85  20       sta   $20        Betrieb ist; Flags setzen
fa58   a9  05       lda   #$05       Adresse auf $fa05
fa5a   85  62       sta   $62        zur Rückkehr
fa5c   a9  fa       lda   #$fa       aus den Stepperroutinen
fa5e   85  63       sta   $63        setzen
fa60   4c  be  fa   jmp   $fabe      zurück aus IRQ-Programm
-------------------------------------------------------------------------------
fa63                                 Serviceroutine für Steppermotor. Hier wird die eigentliche
                                     Bewegung des Kopfes gesteuert. Dies geschieht nach
                                     folgendem Prinzip: Der Steppermotor verfügt übr 2 Spulen,
                                     die durch die Bits 0 und 1 im Steuerport des Disk-
                                     controllers gesteuert werden. Erhalten die Bits
                                     nacheinander die erte: 00/01/10/11/00/..., so wird der Kopf
                                     nach innen bewegt. Das pas- siert beim Einsprung von $fa63
                                     mit 'INX'. Erhalten die Bits jedoch die Werte:
                                     11/10/01/00/11/..., so wird der Kopf nach außen bewegt
                                     (Richtung Track 0). Das passiert beim Einsprung ab $fa69,
                                     der z.B. von der Routine bei $ff7e mit 'DEX' erfolgt.
```

```
fa63   c6  4a       dec   $4a         Zähler für Steps vermindern
fa65   ae  00  1c   ldx   $1c00       Steuerport lesen
fa68   e8           inx               Bitfolge 00/01/10/11/... herstellen

fa69   8a           txa               und Wert übergeben
fa6a   29  03       and   #$03        Bits 0 und 1 isolieren
fa6c   85  4b       sta   $4b         und abspeichern
fa6e   ad  00  1c   lda   $1c00       Steuerport lesen
fa71   29  fc       and   #$fc        Bits 0 und 1 löschen
fa73   05  4b       ora   $4b         neuen Zustand der Bits einfügen
fa75   8d  00  1c   sta   $1c00       und an Diskcontroller ausgeben
fa78   4c  be  fa   jmp   $fabe       zurück aus IRQ-Programm; Ende der Jobschleife
-------------------------------------------------------------------------
fa7b                                  Anfahren des Steppermotors im schnellen Steppermodus
                                      steuern. Für den schnellen Steppermodus wird dabei die IRQ-
                                      Periodik von 8 ms auf (8192-(Inhalt von $5f*256))/1000) ms
                                      heruntergesetzt. Bei einem Wert von 4 in $5f heißt das,
                                      eine Zyklendauer von 7 ms, statt wie bisher 8 ms.
fa7b   38           sec               Subtraktion vorbereiten
fa7c   ad  07  1c   lda   $1c07       Timer Hi für Jobschleife um den Anfahrfaktor 4
fa7f   e5  5f       sbc   $5f         für den Steppermotor verringern, um damit ein
fa81   8d  05  1c   sta   $1c05       schnelleres Timing zur Kopfbewegung zu bekommen
fa84   c6  60       dec   $60         Anzahl der Anfahrschritte minus 1
fa86   d0  0c       bne   $fa94       verzweige, wenn noch nicht 0
fa88   a5  5e       lda   $5e         jetzt schon den Zähler für das Abbremsen
fa8a   85  60       sta   $60         setzen
fa8c   a9  97       lda   #$97        Adresse $fa97
fa8e   85  62       sta   $62         für den schnellen
fa90   a9  fa       lda   #$fa        Steppermodus
fa92   85  63       sta   $63         setzen
fa94   4c  2e  fa   jmp   $fa2e       Kopf um einen Step bewegen
-------------------------------------------------------------------------
fa97                                  Routine zur Steuerung des schnellen Stepper. modus. Dieser
                                      Modus wird durch ein Heraufsetzen der IRQ-Frequenz
                                      initialisiert, was eine schnellere Kopfbewegung zur Folge
                                      hat, da der Schreib-/Lesekopf der 1571 genau jeden IRQ.
                                      Durchgang einen Step bewegt wird.
fa97   c6  61       dec   $61         Schrittzähler minus 1
fa99   d0  f9       bne   $fa94       weitermachen, wenn noch nicht Null
fa9b   a9  a5       lda   #$a5        Adresse $faa5
fa9d   85  62       sta   $62         zum Abbremsen des
fa9f   a9  fa       lda   #$fa        Steppermotors
faa1   85  63       sta   $63         setzen
faa3   d0  ef       bne   $fa94       unbedingter Sprung zur Kopfbewegung
-------------------------------------------------------------------------
faa5                                  Routine zum Abbremsen des Steppermotors durch Verringern
                                      der IRQ-Taktfrequenz.
faa5   ad  07  1c   lda   $1c07       Timer Hi durch Addition des Abbremswertes
faa8   18           clc               wieder auf die
faa9   65  5f       adc   $5f         normale Geschwindigkeit bringen
faab   8d  05  1c   sta   $1c05       und neu starten
```

```
faae   c6  60       dec   $60         Zähler für Bremsen vermindern
fab0   d0  e2       bne   $fa94       weitermachen, wenn noch nicht 0
fab2   a9  4e       lda   #$4e        Adresse $fa4e
fab4   85  62       sta   $62         zur Beendigung
fab6   a9  fa       lda   #$fa        des schnellen Steppermodus
fab8   85  63       sta   $63         setzen
faba   a9  05       lda   #$05        Bremsfaktor zum Anhalten auf 5
fabc   85  60       sta   $60         setzen
fabe   ad  0c  1c   lda   $1c0c       BYTE READY-Leitung
fac1   29  fd       and   #$fd        zurücksetzen; Status wieder
fac3   8d  0c  1c   sta   $1c0c       herstellen
fac6   60           rts               zum Hauptinterruptprogramm; Ende des IRQ
-------------------------------------------------------------------------
fac7                                  Jobroutine zum Formatieren einer Diskette im 1541-Modus.
                                      Diese Routine wird durch die Formatierungsroutine ab $c8c6
                                      initialisiert und über einen Vektor im RAM bei $0600 (jmp
                                      $fac7) angesprungen; der Rest wird über die Jobschleife
                                      gesteuert, indem während jeden Durchlaufs ein Track
                                      formatiert wird.
fac7   a5  51       lda   $51         Formatierung bereits im Gange?
fac9   10  2a       bpl   $faf5       verzweige, wenn ja
facb   a6  3d       ldx   $3d         Drivenummer für Jobschleife
facd   a9  60       lda   #$60        Bit 5 und 6 zum Bereitmachen des Laufwerks
facf   95  20       sta   $20,x       setzen
fad1   a9  01       lda   #$01        Track 1 für Beginn an Jobschleife
fad3   95  22       sta   $22,x       übergeben und Track (ist auch Flag) für
fad5   85  51       sta   $51         laufende Formatierung setzen
fad7   a9  a4       lda   #$a4        46 Tracks nach außen fahren (entspricht
fad9   85  4a       sta   $4a         -92 Schritten); BUMP ausführen
fadb   ad  00  1c   lda   $1c00       Schrittphase für Kopfbewegung auf
fade   29  fc       and   #$fc        00 (Bit 0 und 1)
fae0   8d  00  1c   sta   $1c00       setzen
fae3   a9  0a       lda   #$0a        10 Fehlversuche erlauben
fae5   8d  20  06   sta   $0620       Zähler setzen
fae8   a9  a0       lda   #$a0        erster Schätzwert für halbe Kapazität
faea   8d  21  06   sta   $0621       $0621 eines Tracks auf 4000 Bytes
faed   a9  0f       lda   #$0f        ($0621/0622 = $OfaO)
faef   8d  22  06   sta   $0622       $0622 setzen
faf2   4c  9c  f9   jmp   $f99c       Jobschleife; Kopf positionieren; Drive starten
faf5   a0  00       ldy   #$00        Index in Jobspeicher
faf7   d1  32       cmp   ($32),y     neuer Track gleich letztem Track?
faf9   f0  05       beq   $fb00       verzweige, wenn ja
fafb   91  32       sta   ($32),y     neue Tracknummer übernehmen
fafd   4c  9c  f9   jmp   $f99c       und Kopf positionieren; zur Jobschleife
fb00   ad  00  1c   lda   $1c00       Steuerport für Diskcontroller lesen
fb03   29  10       and   #$10        WRITE PROTECT?
fb05   d0  05       bne   $fb0c       verzweige, wenn nein
fb07   a9  08       lda   #$08        Nummer der Fehlermeldung
fb09   4c  d3  fd   jmp   $fdd3       '26, WRITE PROTECT ON' ausgeben
fb0c   20  a3  fd   jsr   $fda3       Track löschen; mit SYNC beschreiben
fb0f   20  c3  fd   jsr   $fdc3       $0621/0622 mal $ff schreiben
fb12   a9  55       lda   #$55        GCR-Leerbyte (0)
fb14   8d  01  1c   sta   $1c01       zum Diskcontroller
fb17   20  c3  fd   jsr   $fdc3       und $0621/0622 mal schreiben
fb1a   20  00  fe   jsr   $fe00       auf Lesen umschalten
fb1d   20  56  f5   jsr   $f556       SYNC-Signal abwarten
fb20   a9  40       lda   #$40        Timer 1 auf
```

```
fb22   0d  0b  18    ora   $180b       'free running mode'
fb25   8d  0b  18    sta   $180b       setzen
fb28   a9  62        lda   #$62        98 Taktzyklen entsprechen ca. 0.1 ms
fb2a   8d  06  18    sta   $1806       sta $1806 als Timerwert setzen
fb2d   a9  00        lda   #$00        Timer Hi auf 0
fb2f   8d  07  18    sta   $1807       Timer setzen
fb32   8d  05  18    sta   $1805       Timer starten
fb35   a0  00        ldy   #$00        Zähler zurücksetzen
fb37   a2  00        ldx   #$00        Zähler zurücksetzen
fb39   2c  00  1c    bit   $1c00       auf Beginn der SYNC-Zone
fb3c   30  fb        bmi   $fb39       warten
fb3e   2c  00  1c    bit   $1c00       auf Ende der SYNC-Zone
fb41   10  fb        bpl   $fb3e       warten
fb43   ad  04  18    lda   $1804       Interrupts für Timerstart löschen
fb46   2c  00  1c    bit   $1c00       Schleife zur Messung des Nicht-SYNC
fb49   10  11        bpl   $fb5c       Bereichs
fb4b   ad  0d  18    lda   $180d       IFR lesen
fb4e   0a            asl               Timerbit nach Position 7 schieben
fb4f   10  f5        bpl   $fb46       warten, wenn Timer noch nicht 0
fb51   e8            inx               Zähler Lo erhöhen
fb52   d0  ef        bne   $fb43       weitermachen
fb54   c8            iny               wenn Null, auch Zähler Hi erhöhen
fb55   d0  ec        bne   $fb43       weitermachen, wenn nicht abgelaufen
fb57   a9  02        lda   #$02        Nummer der Fehlermeldung
fb59   4c  d3  fd    jmp   $fdd3       '20, READ ERROR' ausgeben
fb5c   86  71        stx   $71         Zählerstand als Wert für die Länge
fb5e   84  72        sty   $72         des $55-Bereichs merken
fb60   a2  00        ldx   #$00        Zähler wieder
fb62   a0  00        ldy   #$00        zurücksetzen
fb64   ad  04  18    lda   $1804       Interrupts für Timerstart löschen
fb67   2c  00  1c    bit   $1c00       Schleife zur Messung des SYNC
fb6a   30  11        bmi   $fb7d       Bereichs
fb6c   ad  0d  18    lda   $180d       IFR lesen
fb6f   0a            asl               Timerbit nach Position 7 schieben
fb70   10  f5        bpl   $fb67       warten, wenn Timer noch nicht 0
fb72   e8            inx               Zähler Lo erhöhen
fb73   d0  ef        bne   $fb64       weitermachen
fb75   c8            iny               wenn Null, auch Zähler Hi erhöhen
fb76   d0  ec        bne   $fb64       weitermachen, bis Überlauf
fb78   a9  02        lda   #$02        Nummer der Fehlermeldung
fb7a   4c  d3  fd    jmp   $fdd3       '20, READ ERROR' ausgeben
fb7d   38            sec               Differenz der bei den
fb7e   8a            txa               Meßwerte
fb7f   e5  71        sbc   $71         bi lden, um
fb81   aa            tax               den Größenunterschied beider Felder
fb82   85  70        sta   $70         zu erhalten
fb84   98            tya               Ergebnis
fb85   e5  72        sbc   $72         nach $70/71
fb87   a8            tay               bringen und
fb88   85  71        sta   $71         merken
fb8a   10  0b        bpl   $fb97       verzweige, wenn Differenz größer Null
fb8c   49  ff        eor   #$ff        Bytes invertieren
fb8e   a8            tay               und den
fb8f   8a            txa               Absolutwert
fb90   49  ff        eor   #$ff        der
fb92   aa            tax               Differenz
fb93   e8            inx               bi lden
fb94   d0  01        bne   $fb97       verzweige, wenn ungleich Null
```

```
fb96   c8          iny             danach
fb97   98          tya             prüfen, ob
fb98   d0  04      bne   $fb9e     Differenz
fb9a   e0  04      cpx   #$04      kleiner 4?
fb9c   90  18      bcc   $fbb6     verzeige, enn ja
fb9e   06  70      asl   $70       Wert verdoppeln
fba0   26  71      rol   $71       und zum
fba2   18          clc             ert für
fba3   a5  70      lda   $70       die Länge der
fba5   6d  21  06  adc   $0621     $0621 Bereiche
fba8   8d  21  06  sta   $0621     ($0621/0622)
fbab   a5  71      lda   $71       addieren
fbad   6d  22  06  adc   $0622     Hi-Anteil des ertes
fbb0   8d  22  06  sta   $0622     ebenfalls addieren und merken
fbb3   4c  0c  fb  jmp   $fb0c     Messung mit neuen Werten beginnen
fbb6   a2  00      ldx   #$00      Zähler ieder
fbb8   a0  00      ldy   #$00      zurücksetzen
fbba   b8          clv             Flag für BYTE READY löschen
fbbb   ad  00  1c  lda   $1c00     Steuerport des Diskcontrollers lesen
fbbe   10  0e      bpl   $fbce     verzeige, enn SYNC noch anliegt
fbc0   50  f9      bvc   $fbbb     sonst Byte einlesen
fbc2   b8          clv             Flag wieder löschen
fbc3   e8          inx             Zähler Lo erhöhen
fbc4   d0  f5      bne   $fbbb     weitermachen
fbc6   c8          iny             wenn Überlauf, auch Zähler Hi erhöhen
fbc7   d0  f2      bne   $fbbb     weitermachen, bis Zähler abgelaufen
fbc9   a9  03      lda   #$03      Nummer der Fehlermeldung
fbcb   4c  d3  fd  jmp   $fdd3     '21, READ ERROR' ausgeben
fbce   8a          txa             Zählerwert
fbcf   0a          asl             mal 2
fbd0   8d  25  06  sta   $0625     und in $0624/0625 abspeichern
fbd3   98          tya             Zählerert Hi
fbd4   2a          rol             mit Übertrag mal 2
fbd5   8d  24  06  sta   $0624     und ebenfalls abspeichern
fbd8   a9  bf      lda   #$bf      Timer 1
fbda   2d  0b  18  and   $180b     Steuerbit löschen;
fbdd   8d  0b  18  sta   $180b     Timer anhalten
fbe0   a9  66      lda   #$66      Gesamtanzahl aller
fbe2   8d  26  06  sta   $0626     Bytes
fbe5   a6  43      ldx   $43       berechnen,
fbe7   a0  00      ldy   #$00      die auf
fbe9   98          tya             diesen Track
fbea   18          clc             passen und
fbeb   6d  26  06  adc   $0626     somit
fbee   90  01      bcc   $fbf1     geschrieben
fbf0   c8          iny             werden
fbf1   c8          iny             müssen
fbf2   ca          dex             Zähler vermindern
fbf3   d0  f5      bne   $fbea     und weiter addieren
fbf5   49  ff      eor   #$ff      Absolutwert bilden
fbf7   38          sec             wenn Zähler auf Null
fbf8   69  00      adc   #$00      Übertrag addieren
fbfa   18          clc             Anzahl der Bytes in
fbfb   6d  25  06  adc   $0625     den Blockzwischenräumen
fbfe   b0  03      bcs   $fc03     berechnen
fc00   ce  24  06  dec   $0624     Zählwert Hi vermindern
fc03   aa          tax             und
fc04   98          tya             den
```

```
fc05   49  ff       eor   #$ff        Absolutwert bilden
fc07   38           sec               danach
fc08   69  00       adc   #$00        den Übertrag addieren
fc0a   18           clc               und
fc0b   6d  24  06   adc   $0624       testen, ob Spurkapazität überschritten wurde
fc0e   10  05       bpl   $fc15       verzweige, wenn nein
fc10   a9  04       lda   #$04        Nunrner der Fehlermeldung
fc12   4c  d3  fd   jmp   $fdd3       '22, READ ERROR' ausgeben
fc15   a8           tay               Gesamtanzahl der Bytes
fc16   8a           txa               in den Zwischenräumen
fc17   a2  00       ldx   #$00        geteilt durch die
fc19   38           sec               Anzahl der Sektoren
fc1a   e5  43       sbc   $43         ergibt die Anzahl
fc1c   b0  03       bcs   $fc21       der Bytes pro
fc1e   88           dey               Zwischenraum
fc1f   30  03       bmi   $fc24       weitermachen
fc21   e8           inx               Zähler erhöhen
fc22   d0  f5       bne   $fc19       weitermachen
fc24   8e  26  06   stx   $0626       Zahl der Bytes merken
fc27   e0  04       cpx   #$04        Anzahl kleiner als 41
fc29   b0  05       bcs   $fc30       verzweige, wenn nein
fc2b   a9  05       lda   #$05        Nummer der Fehlermeldung
fc2d   4c  d3  fd   jmp   $fdd3       '23, READ ERROR' ausgeben
fc30   18           clc               und
fc31   65  43       adc   $43         Rest der Division plus Anzahl der Sektoren
fc33   8d  27  06   sta   $0627       merken
fc36   a9  00       lda   #$00        Zähler für die Sektoren pro Track
fc38   8d  28  06   sta   $0628       setzen
fc3b   a0  00       ldy   #$00        Zeiger in Puffer
fc3d   a6  3d       ldx   $3d         Drivenummer für Job
fc3f   a5  39       lda   $39         Kennzeichen S08 für Blockheader
fc41   99  00  03   sta   $0300,y     in Puffer schreiben
fc44   c8           iny               ein Byte für die Prüfsumme
fc45   c8           iny               überspringen
fc46   ad  28  06   lda   $0628       Nummer des entsprechenden Sektors
fc49   99  00  03   sta   $0300,y     in den Puffer schreiben
fc4c   c8           iny               nächstes Byte
fc4d   a5  51       lda   $51         aktuelle Tracknummer der Formatierung
fc4f   99  00  03   sta   $0300,y     in den Puffer schreiben
fc52   c8           iny               nächstes Byte
fc53   b5  13       lda   $13,x       ID 2
fc55   99  00  03   sta   $0300,y     in den Puffer schreiben
fc58   c8           iny               nächstes Byte
fc59   b5  12       lda   $12,x       ID 1
fc5b   99  00  03   sta   $0300,y     in den Puffer schreiben
fc5e   c8           iny               nächstes Byte
fc5f   a9  0f       lda   #$0f        $Of als Füllbyte
fc61   99  00  03   sta   $0300,y     zweimal
fc64   c8           iny               in den
fc65   99  00  03   sta   $0300,y     Puffer schreiben
fc68   c8           iny               nächstes Byte
fc69   a9  00       lda   #$00        Startwert für Summe
fc6b   59  fa  02   eor   $02fa,y     Prüfsumme
fc6e   59  fb  02   eor   $02fb,y     über alle
fc71   59  fc  02   eor   $02fc,y     Parameter
fc74   59  fd  02   eor   $02fd,y     berechnen
fc77   99  f9  02   sta   $02f9,y     und in Puffer schreiben
fc7a   ee  28  06   inc   $0628       Nummer des Sektors erhöhen
```

```
fc7d   ad  28  06    lda   $0628       und
fc80   c5  43        cmp   $43         auf maximale Sektornummer testen
fc82   90  bb        bcc   $fc3f       nächsten Blockheader, wenn nicht erreicht
fc84   98            tya               Endeposition im Puffer
fc85   48            pha               merken
fc86   e8            inx               $00 + 1 (nnötig; eigentlich NOP)
fc87   8a            txa               als 'dummy' (Leerinhalt)
fc88   9d  00  05    sta   $0500,x     in den Puffer schreiben
fc8b   e8            inx               und
fc8c   d0  fa        bne   $fc88       gesamten Puffer vollschreiben
fc8e   a9  03        lda   #$03        Pufferadresse Hi auf $03
fc90   85  31        sta   $31         setzen
fc92   20  30  fe    jsr   $fe30       Pufferinhalt nach GCR umwandeln
fc95   68            pla               Endeposition im Puffer zurückholen
fc96   a8            tay               und wieder als Index
fc97   88            dey               minus 1; Pufferinhalt ab $0300,Y
fc98   20  e5  fd    jsr   $fde5       nach $0344,Y verschieben; GCR-Bytes in den
fc9b   20  f5  fd    jsr   $fdf5       freigewordenen Bereich davor schieben
fc9e   a9  05        lda   #$05        Pufferadresse Hi $05
fca0   85  31        sta   $31         setzen
fca2   20  e9  f5    jsr   $f5e9       Prüfsumme über Datenblock berechnen
fca5   85  3a        sta   $3a         und abspeichern
fca7   20  8f  f7    jsr   $f78f       Datenblock in GCR-Code umwandeln
fcaa   a9  00        lda   #$00        Zeiger auf Blockheader
fcac   85  32        sta   $32         setzen
fcae   20  0e  fe    jsr   $fe0e       Track löschen; DC auf Schreibbetrieb
fcb1   a9  ff        lda   #$ff        SYNC-Markierung
fcb3   8d  01  1c    sta   $1c01       auf Diskette
fcb6   a2  05        ldx   #$05        schreiben, d.h. 5 mal
fcb8   50  fe        bvc   $fcb8       den Wert $ff
fcba   b8            clv               auf die Diskette
fcbb   ca            dex               bringen
fcbc   d0  fa        bne   $fcb8       und weitermachen
fcbe   a2  0a        ldx   #$0a        10 Bytes
fcc0   a4  32        ldy   $32         Pufferzeiger (für Blockheader)
fcc2   50  fe        bvc   $fcc2       Schreiben abwarten
fcc4   b8            clv               Flag löschen
fcc5   b9  00  03    lda   $0300,y     Blockheader aus Puffer
fcc8   8d  01  1c    sta   $1c01       auf die
fccb   c8            iny               Diskette
fccc   ca            dex               schreiben und zwar 10 GCR-Bytes
fccd   d0  f3        bne   $fcc2       weiter, bis alle Bytes geschrieben wurden
fccf   a2  09        ldx   #$09        9 'Leerbytes'
fcd1   50  fe        bvc   $fcd1       schreiben
fcd3   b8            clv               Flag löschen
fcd4   a9  55        lda   #$55        'Leerbyte'
fcd6   8d  01  1c    sta   $1c01       zum Diskcontroller
fcd9   ca            dex               nächstes Byte
fcda   d0  f5        bne   $fcd1       alle 9 Bytes schreiben
fcdc   a9  ff        lda   #$ff        Wert für SYNC-Markierung
fcde   a2  05        ldx   #$05        5 Bytes
fce0   50  fe        bvc   $fce0       als SYNC-Markierung
fce2   b8            clv               auf
fce3   8d  01  1c    sta   $1c01       die Diskette
fce6   ca            dex               schreiben
fce7   d0  f7        bne   $fce0       weitermachen, bis fertig
fce9   a2  bb        ldx   #$bb        Index in Ausweichpuffer
fceb   50  fe        bvc   $fceb       Schreibvorgang abwarten
```

```
fced   b8          clv             Flag löschen
fcee   bd  00  01  lda   $0100,x    Byte aus Ausweichpuffer
fcf1   8d  01  1c  sta   $1c01      auf Diskette schreiben
fcf4   e8          inx             nächstes Byte
fcf5   d0  f4      bne   $fceb     weitermachen, bis 70 Byte auf Diskette
fcf7   a0  00      ldy   #$00      Index für aktuellen Puffer
fcf9   50  fe      bvc   $fcf9     Schreibvorgang abwarten
fcfb   b8          clv             Flag löschen
fcfc   b1  30      lda   ($30),y   Byte aus Puffer holen
fcfe   8d  01  1c  sta   $1c01      und auf Diskette schreiben
fd01   c8          iny             weitermachen,
fd02   d0  f5      bne   $fcf9     bis der gesamte Puffer geschrieben wurde
fd04   a9  55      lda   #$55      'Leerbyte'
fd06   ae  26  06  ldx   $0626     Größe der Lücken holen
fd09   50  fe      bvc   $fd09     Schreibvorgang abwarten
fd0b   b8          clv             Flag löschen
fd0c   8d  01  1c  sta   $1c01      Byte auf Diskette schreiben
fd0f   ca          dex             nächstes Byte,
fd10   d0  f7      bne   $fd09     bis Lücke fertig
fd12   a5  32      lda   $32       Zeiger auf Blockheader in Puffer
fd14   18          clc             mit
fd15   69  0a      adc   #$0a      10 addieren
fd17   85  32      sta   $32       zeigt jetzt auf nächsten Blockheader
fd19   ce  28  06  dec   $0628     nächste Sektornummer
fd1c   d0  93      bne   $fcb1     weitermachen, wenn noch nicht alle geschrieben
fd1e   50  fe      bvc   $fd1e     Schreibvorgang abwarten
fd20   b8          clv             Flag wieder löschen
fd21   50  fe      bvc   $fd21     Schreibvorgang abwarten
fd23   b8          clv             Flag wieder löschen
fd24   20  00  fe  jsr   $fe00     Diskcontroller auf Lesen umschalten
fd27   a9  c8      lda   #$c8      200 Leseversuche
fd29   8d  23  06  sta   $0623     setzen
fd2c   a9  00      lda   #$00      Pufferadresse Lo
fd2e   85  30      sta   $30       setzen
fd30   a9  03      lda   #$03      Pufferadresse Hi auf $03
fd32   85  31      sta   $31       setzen
fd34   a5  43      lda   $43       Anzahl der Sektoren pro Track
fd36   8d  28  06  sta   $0628     merken
fd39   20  56  f5  jsr   $f556     SYNC-Signal abwarten
fd3c   a2  0a      ldx   #$0a      10 Bytes für Blockheader
fd3e   a0  00      ldy   #$00      Pufferzeiger
fd40   50  fe      bvc   $fd40     Byte lesen
fd42   b8          clv             Flag für BYTE READY löschen
fd43   ad  01  1c  lda   $1c01      Byte vom Diskcontroller holen
fd46   d1  30      cmp   ($30),y   und mit Speicherinhalt vergleichen
fd48   d0  0e      bne   $fd58     verzweige, wenn ungleich
fd4a   c8          iny             Zeiger auf nächstes Byte
fd4b   ca          dex             Zähler vermindern
fd4c   d0  f2      bne   $fd40     weitermachen, bis 10 Bytes verglichen
fd4e   18          clc             Pufferadresse
fd4f   a5  30      lda   $30       um 10 erhöhen; zeigt jetzt auf den
fd51   69  0a      adc   #$0a      nächsten Blockheader
fd53   85  30      sta   $30       Pufferadresse setzen
fd55   4c  62  fd  jmp   $fd62     weitermachen; alles ok
fd58   ce  23  06  dec   $0623     Fehlerzähler minus 1
fd5b   d0  cf      bne   $fd2c     weitermachen, wenn ungleich 0
fd5d   a9  06      lda   #$06      Nummer der Fehlermeldung
fd5f   4c  d3  fd  jmp   $fdd3     '24, READ ERROR' ausgeben
```

```
fd62   20  56  f5    jsr   $f556       SYNC.Signal abwarten
fd65   a0  bb        ldy   #$bb        Zeiger in Ausweichpuffer
fd67   50  fe        bvc   $fd67       Byte lesen
fd69   b8            clv               Flag für BYTE READY löschen
fd6a   ad  01  1c    lda   $1c01       Byte vom Diskcontroller holen
fd6d   d9  00  01    cmp   $0100,y     mit Pufferinhalt vergleichen
fd70   d0  e6        bne   $fd58       verzweige, wenn ungleich
fd72   c8            iny               sonst Zeiger auf nächstes Byte
fd73   d0  f2        bne   $fd67       und weitermachen
fd75   a2  fc        ldx   #$fc        Zähler für Anzahl der Bytes im Datenpuffer
fd77   50  fe        bvc   $fd77       Byte lesen
fd79   b8            clv               Flag löschen
fd7a   ad  01  1c    lda   $1c01       Byte vom Diskcontroller holen
fd7d   d9  00  05    cmp   $0500,y     und mit Pufferinhalt vergleichen
fd80   d0  d6        bne   $fd58       verzweige, wenn ungleich
fd82   c8            iny               Zeiger auf nächstes Byte
fd83   ca            dex               Zähler vermindern
fd84   d0  f1        bne   $fd77       weitermachen; nächstes Byte vergleichen
fd86   ce  28  06    dec   $0628       Sektorzähler vermindern
fd89   d0  ae        bne   $fd39       nächsten Sektor testen
fd8b   e6  51        inc   $51         Tracknummer für Formatierung plus 1
fd8d   a5  51        lda   $51         und mit
fd8f   c9  24        cmp   #$24        Maximalwert 36 vergleichen
fd91   b0  03        bcs   $fd96       verzweige, wenn größer gleich
fd93   4c  9c  f9    jmp   $f99c       sonst Kopf positionieren und weitermachen
fd96   a9  ff        lda   #$ff        Flag für Formatierung beendet
fd98   85  51        sta   $51         setzen
fd9a   a9  00        lda   #$00        Flag für GCR-Bytes im Puffer
fd9c   85  50        sta   $50         löschen
fd9e   a9  01        lda   #$01        Nummer der Rückmeldung
fda0   4c  69  f9    jmp   $f969       '00, OK' ausgeben; Ende
--------------------------------------------------------------------------
fda3                                   Spur durch Beschreiben mit 10240 $ff-Bytes löschen.
fda3   ad  0c  1c    lda   $1c0c       PCR lesen
fda6   29  1f        and   #$1f        und
fda8   09  c0        ora   #$c0        auf Schreibbetrieb
fdaa   8d  0c  1c    sta   $1c0c       umschalten
fdad   a9  ff        lda   #$ff        Port für Schreib-/Lesekopf
fdaf   8d  03  1c    sta   $1c03       auf Ausgang
fdb2   8d  01  1c    sta   $1c01       Byte zum Diskcontroller
fdb5   a2  28        ldx   #$28        und
fdb7   a0  00        ldy   #$00        10240 mal
fdb9   50  fe        bvc   $fdb9       Byte auf Diskette schreiben
fdbb   b8            clv               Flag löschen
fdbc   88            dey               nächstes Byte
fdbd   d0  fa        bne   $fdb9       schreiben
fdbf   ca            dex               und weitermachen
fdc0   d0  f7        bne   $fdb9       bis 10240 maL geschrieben
fdc2   60            rts               Ende
--------------------------------------------------------------------------
fdc3                                   Eine in $0621/0622 gegebene Anzahl von BYTE READY Signalen
                                       abwarten.
fdc3   ae  21  06    ldx   $0621       Index Lo
fdc6   ac  22  06    ldy   $0622       Index Hi
fdc9   50  fe        bvc   $fdc9       BYTE READY abwarten
fdcb   b8            clv               Flag löschen
fdcc   ca            dex               Index Lo vermindern
```

```
fdcd   d0  fa       bne   $fdc9       und weitermachen
fdcf   88           dey               Index Hi vermindern
fdd0   10  f7       bpl   $fdc9       und weitermachen
fdd2   60           rts               Ende
-------------------------------------------------------------------------
fdd3                                  Fehlerausgang beim Formatieren
fdd3   ce  20  06   dec   $0620       Zähler für Versuche minus 1
fdd6   f0  03       beq   $fddb       Fehler, wenn Null
fdd8   4c  9c  f9   jmp   $f99c       sonst weitermachen
fddb   a0  ff       ldy   #$ff        Flag für Ende der Formatierung
fddd   84  51       sty   $51         setzen
fddf   c8           iny               Y=00
fde0   84  50       sty   $50         Flag für GCR-Bytes im Puffer löschen
fde2   4c  69  f9   jmp   $f969       Ende; Fehlernummer in A
-------------------------------------------------------------------------
fde5                                  Schiebt die ersten 69 ($45) Bytes in Puffer 0 um 69
                                      Positionen nach oben, um am Anfang Platz zu bekommen.
fde5   b9  00  03   lda   $0300,y     Byte vom Anfang holen
fde8   99  45  03   sta   $0345,y     und weiter oben ablegen
fdeb   88           dey               Zähler vermindern
fdec   d0  f7       bne   $fde5       und weitermachen
fdee   ad  00  03   lda   $0300       erstes Byte am Anfang
fdf1   8d  45  03   sta   $0345       ebenfalls nach oben schieben
fdf4   60           rts               Ende
-------------------------------------------------------------------------
fdf5                                  Holt den Inhalt des Ausweichpuffers in den aktuellen
                                      Puffer.
fdf5   a0  44       ldy   #$44        Zeiger in Ausweichpuffer
fdf7   b9  bb  01   lda   $01bb,y     Byte aus Ausweichpuffer
fdfa   91  30       sta   ($30),y     in aktuellen Puffer übernehmen
fdfc   88           dey               Zeiger vermindern
fdfd   10  f8       bpl   $fdf7       und weitermachen
fdff   60           rts               Ende
-------------------------------------------------------------------------
fe00                                  Diskcontroller auf Lesen umschalten.
fe00   ad  0c  1c   lda   $1c0c       PCR lesen
fe03   09  e0       ora   #$e0        und auf Lesebetrieb
fe05   8d  0c  1c   sta   $1c0c       umschalten
fe08   a9  00       lda   #$00        Richtungsregister für Schreib-/Lesekopf
fe0a   8d  03  1c   sta   $1c03       auf Eingang
fe0d   60           rts               Ende
-------------------------------------------------------------------------
fe0e                                  überschreibt eine Spur mit 10240 $55 Bytes.
fe0e   ad  0c  1c   lda   $1c0c       PCR lesen
fe11   29  1f       and   #$1f        und
fe13   09  c0       ora   #$c0        auf Schreibbetrieb
fe15   8d  0c  1c   sta   $1c0c       umschalten
fe18   a9  ff       lda   #$ff        Richtungsregister für Schreib-/Lesekopf
fe1a   8d  03  1c   sta   $1c03       auf Ausgang setzen
fe1d   a9  55       lda   #$55        Byte
fe1f   8d  01  1c   sta   $1c01       an Diskcontroller
fe22   a2  28       ldx   #$28        und
fe24   a0  00       ldy   #$00        10240 mal
fe26   50  fe       bvc   $fe26       Byte schreiben
fe28   b8           clv               Flag löschen
fe29   88           dey               Zähler vermindern
fe2a   d0  fa       bne   $fe26       und weitermachen
```

```
fe2c   ca           dex              Zähler Hi vermindern
fe2d   d0  f7       bne   $fe26       und weitermachen
fe2f   60           rts              Ende
-------------------------------------------------------------------------
fe30                                 Wandelt die Bytes des Blockheaders von Binär-(4 Bit)-Form
                                     in die GCR-(5 Bit)-Form um und schreibt diese Werte in den
                                     Ausweichpuffer.
fe30   a9  00       lda   #$00       Puffer adressen und Indizes löschen:
fe32   85  30       sta   $30        Puffer adresse Lo
fe34   85  2e       sta   $2e        Pufferadresse Lo
fe36   85  36       sta   $36        Pufferzeiger
fe38   a9  bb       lda   #$bb       Pufferzeiger für Ausweichpuffer
fe3a   85  34       sta   $34        setzen
fe3c   a5  31       lda   $31        Pufferadresse Hi des aktuellen Puffers
fe3e   85  2f       sta   $2f        übernehmen
fe40   a9  01       lda   #$01       Pufferadresse Hi
fe42   85  31       sta   $31        setzen
fe44   a4  36       ldy   $36        Pufferzeiger holen
fe46   b1  2e       lda   ($2e),y    Byte aus Puffer
fe48   85  52       sta   $52        für Umwandlung
fe4a   c8           iny              nächstes Byte
fe4b   b1  2e       lda   ($2e),y    Byte aus Puffer
fe4d   85  53       sta   $53        für Umwandlung
fe4f   c8           iny              nächstes Byte
fe50   b1  2e       lda   ($2e),y    Byte aus Puffer
fe52   85  54       sta   $54        für Umwandlung
fe54   c8           iny              nächstes Byte
fe55   b1  2e       lda   ($2e),y    Byte aus Puffer
fe57   85  55       sta   $55        für Umwandlung
fe59   c8           iny              nächstes Byte
fe5a   f0  08       beq   $fe64      verzweige, wenn Ende
fe5c   84  36       sty   $36        Pufferzeiger merken
fe5e   20  d0  f6   jsr   $f6d0      4 Bytes in 5 GCR-Bytes umwandeln
fe61   4c  44  fe   jmp   $fe44      weitermachen
fe64   4c  d0  f6   jmp   $f6d0      4 Bytes in 5 GCR-Bytes umwandelnj Ende
-------------------------------------------------------------------------
fe67                                 System-IRQ-Routinej wird über den Hardwarevektor
                                     angesprungen.
fe67   6c  a9  02   jmp   ($02a9)    Sprung zu den Routinen $9d88 oder $9dde
-------------------------------------------------------------------------
fe6a ff ...                          leerer Speicherbereich
fe84 ... ff                          aus 1541 ROM[1]
-------------------------------------------------------------------------
fe85                                 Werte für Organisation des Directory.
fe85 12                              Spurnummer des Directory (18)
fe86 04                              Anzahl der Bytes pro Spur in der BAM (4)
fe87 04                              BAM steht in 18,0 ab Position 4
fe88 90                              Anfang des Diskettennamens bei Position 144
-------------------------------------------------------------------------
fe89                                 ASCII-Codes der einzelnen Floppy-Befehle.
fe89 56                              V = VALIDATE oder COLLECT Befehl
fe8a 49                              I = INITIALIZE Befehl
fe8b 44                              D = DUPLICATE Befehl (n.v.)
fe8c 4d                              M = MEMORY Befehle
fe8d 42                              B = BLOCK Befehle
fe8e 55                              U = USER Befehle
fe8f 50                              P = POSITION oder RECORD Befehl
fe90 26                              & = &-Befehl
```

[1] @ST: Der Speicherbereich $FE6A .. $FE84 ist nicht wirklich leer, er enthält Reste der IRQ-Routine aus der 1541. Wird aber nicht benutzt.

```
fe91 43                              C = COPY Befehl
fe92 52                              R = RENAME Befehl
fe93 53                              S = SCRATCH Befehl
fe94 4e                              N = NEW oder HEADER Befehl
-------------------------------------------------------------------------
fe95                                 Adressen der einzelnen Floppy-Befehle.

Adresse Lo   Adresse Hi              Adresse Befehl

fe95 84      fea1 ed                 $ed84 = VAL !DA TE
fe96 05      fea2 d0                 $d005 = INITIALIZE
fe97 c1      fea3 c8                 $c8c1 = DUPLICATE (n.v.)
fe98 f8      fea4 ca                 $caf8 = MEMORY
fe99 1b      fea5 cc                 $cc1b = BLOCK
fe9a 5c      fea6 cb                 $cb5c = USER
fe9b 07      fea7 e2                 $e207 = POSITION
fe9c a3      fea8 e7                 $e7a3 = &-Befehl
fe9d f0      fea9 c8                 $c8f0 = COPY
fege 88      feaa ca                 $ca88 = RENAME
fe9f 23      feab c8                 $c823 = SCRATCH
fea0 0d      feac ee                 $ee0d = NEW
-------------------------------------------------------------------------
fead                                 Bitmuster für Befehlssyntax.
fead 51   %01010001                  Kopieren eines Files
feae dd   %11011101                  Umbenennen eines Files
feaf 1c   %00011100                  Löschen eines Files
feb0 ge   %10011110                  Formatieren einer Diskette
feb1 1c   %00011100                  Laden eines Files
-------------------------------------------------------------------------
feb2                                 ASCII-Bytes der Filebetriebsarten.
feb2 52                              R = READ; Lesen eines Files
feb3 57                              W = WRITE; Schreiben eines Files
feb4 41                              A = APPEND; Anhängen von Daten im File
feb5 4d                              M = MODIFY; Lesen eines offengebliebenen Files
-------------------------------------------------------------------------
feb6                                 Kürzel der Filetypen.
feb6 44                              D = DELETED File
feb7 53                              S = SEQUENTIAL File
febB 50                              P = PROGRAM File
feb9 55                              U = USER File
feba 4c                              L = RELATIVE File
-------------------------------------------------------------------------
febb Bezeichnung der Filetypen im Directory.
febb 44   fec0 45   fec5 4c          DELeted file
febc 53   fec1 45   fec6 51          SEQuential file
febd 50   fec2 52   fec7 47          PRoGram file
febe 55   fec3 53   fec8 52          USeR file
febf 52   fec4 45   fec9 4c          RELative file
-------------------------------------------------------------------------
feca                                 Maskenbytes für LED am Laufwerk.
feca 08                              LED-Maskenbyte für Laufwerk 0
fecb 00                              LED-Maskenbyte für Laufwerk 1 (n.v.)
-------------------------------------------------------------------------
fecc                                 Werte für Fehlermeldungen, die über den Prozessorstatus an
                                     andere Routinen übergeben. werden.
fecc 00                              Z-Flag gesetzt
fecd 3f                              kein Flag gesetzt
```

```
fece 7f                                 V-Flag gesetzt
fecf bf                                 N-Flag gesetzt
fed0 ff                                 N-Flag und V-Flag gesetzt
-------------------------------------------------------------------------
fed1                                    Maximalzahl der Sektoren für einen Track.
fed1 11                                 17 Sektoren auf Spur 35-31
fed2 12                                 18 Sektoren auf Spur 30-25
fed3 13                                 19 Sektoren auf Spur 24-18
fed4 15                                 21 Sektoren auf Spur 17-01
-------------------------------------------------------------------------
fed5                                    Format-Zeichen.
fed5 41                                 'A'; Formatkennzeichern der DOS-Version
fed6 04                                 Anzahl der Zonen auf der Diskette mit verschiedenen
                                        Sektorzahlen pro Spur
-------------------------------------------------------------------------
fed7                                    Spurnummern, bei denen ein Zonenwechsel und ein Wechsel des
                                        Timings beim Schreiben und Lesen stattfindet.
fed7 24                                 Spur 36 = Ende der Zone 31-35 (4)
fed8 1f                                 Spur 31 = Ende der Zone 25-30 (3)
fed9 19                                 Spur 25 = Ende der Zone 18-24 (2)
feda 12                                 Spur 18 = Ende der Zone 01-17 (1)
-------------------------------------------------------------------------
fedb                                    Steuerbytes für die Kopfdejustierung bei Leseproblemen. Der
                                        Kopf wird dabei eine Halbspur nach innen, dann zwei
                                        Halbspuren nach außen und wieder eine Halbspur nach innen
                                        bewegt. Dann steht er wieder normal auf der Spur.
fedb 01                                 ein Halbschritt nach innen
fedc ff                                 ein Halbschritt nach außen
fedd ff                                 ein Halbschritt nach außen
fede 01                                 ein Halbschritt nach innen
fedf 00                                 Endekennzeichen
-------------------------------------------------------------------------
fee0                                    Hi-Bytes der Pufferadressen.
fee0 03                                 Puffer 0 ($0300-03ff)
fee1 04                                 Puffer 1 ($0400-04ff)
fee2 05                                 Puffer 2 ($0500-05ff)
fee3 06                                 Puffer 3 ($0600-06ff)
fee4 07                                 Puffer 4 ($0700-07ff)
fee5 07                                 Puffer 5 ($0700-07ff)
-------------------------------------------------------------------------
fee6 ff                                 ehemalige ROM-Prüfsumme; jetzt leer.
-------------------------------------------------------------------------
fee7                                    Sprungvektor des NMI; zeigt auf $eb22 und führt den
                                        'kurzen' RESET ohne Speichertest aus.
fee7   6c  65  00    jmp   ($0065)      Sprung auf $eb22 (normalerweise)
-------------------------------------------------------------------------
feea                                    LED am Laufwerk einschalten; Datenrichtungsreregister für
                                        LED setzen.
feea   8d  00  1c    sta   $1c00        LED einschalten (A enthält bei Einsprung 8)
feed   8d  02  1c    sta   $1c02        entsprechendes Pin auf Ausgang
fef0   4c  7d  ea    jmp   $ea7d        und weiter in Fehlerroutine
-------------------------------------------------------------------------
fef3                                    Verzögerung für seriellen Bus.
fef3   8a            txa                Inhalt von X merken
fef4   a2  05        ldx   #$05         Wert für Verzögerung
fef6   ca            dex                25 us Verzögerung
```

```
fef7   d0  fd       bne   $fef6       ablaufen lassen
fef9   aa           tax               X wieder zurückholen
fefa   60           rts               Ende
--------------------------------------------------------------------------
fefb                                   CLOCK OUT auf Lo und DATA OUT auf Hi setzen.
fefb   20  ae  e9    jsr   $e9ae       CLOCK OUT-Leitung auf Lo setzen
fefe   4c  9c  e9    jmp   $e99c       DATA OUT-Leitung auf Hi setzen; Ende
--------------------------------------------------------------------------
ff01                                   Einsprung vom UI-Befehl. Hier kann die Floppy durch 'UI+'
                                       auf 64er und durch 'UI-' auf VC20 Modus umgeschaltet
                                       werden.
ff01   ad  02  02    lda   $0202       drittes Zeichen aus Befehlsstring
ff04   c9  2d        cmp   #$2d        mit '-' (Minus) vergleichen
ff06   f0  05        beq   $ff0d       verzweige, wenn Minus (UI-)
ff08   38            sec               sonst
ff09   e9  2b        sbc   #$2b        mit '+' (Plus) vergleichen
ff0b   d0  da        bne   $fee7       verzweige, wenn kein Plus
ff0d   85  23        sta   $23         Flag für schnelleren Busbetrieb setzen
ff0f   60            rts               Ende
--------------------------------------------------------------------------
ff10                                   RESET für Datenrichtungsregister A im Buscontroller .
ff10   8e  03  18    stx   $1803       Wert in Datenrichtungsregister
ff13   a9  02        lda   #$02        Wert für Port B
ff15   4c  5a  a6    jmp   $a65a       zur RESET-Routine für 1571-Bausteine
--------------------------------------------------------------------------
ff18                                   RESET für Datenrichtungsregister B in Buscontroller.
ff18   a9  1a        lda   #$1a        Wert für DDRB
ff1a   8d  02  18    sta   $1802       setzen
ff1d   4c  a7  ea    jmp   $eaa7       und zurück zur RESET-Routine
--------------------------------------------------------------------------
ff20                                   Auf Hi der DATA IN-Leitung warten; Timer setzen.
ff20   ad  00  18    lda   $1800       Port lesen
ff23   29  01        and   #$01        Bit für DATA IN isolieren
ff25   d0  f9        bne   $ff20       warten, bis Bit 0 (Leitung Hi) wird
ff27   a9  01        lda   #$01        Wert für Timer
ff29   8d  05  18    sta   $1805       Timer setzen
ff2c   4c  df  e9    jmp   $e9df       Ende
--------------------------------------------------------------------------
ff2f                                   Vorbereitung für Formatieren einer Diskette. Nach Testen
                                       der gewünschten Formtierung wird entweder im 1541 oder im
                                       1571-Modus formatiert.
ff2f   a9  ff        lda   #$ff        Flag für 'Formatierung startet'
ff31   85  51        sta   $51         setzen
ff33   ad  0f  18    lda   $180f       auf 1541-Modus (1 Mhz)
ff36   29  20        and   #$20        testen
ff38   d0  03        bne   $ff3d       verzweige, wenn Floppy im 1571-Modus (2 Mhz)
ff3a   a9  24        lda   #$24        sonst 36 Spuren für Formatierung
ff3c   2c            .byte $2c         nächsten Befehl überspringen
ff3d   a9  47        lda   #$47        71 Spuren für Formatierung
*1 ff3f 8d  ac  02   sta   $02ac       setzen
*0 ff3f 8d  d7  fe   sta   $fed7       nicht setzen (ROM)
ff42   4c  79  a7    jmp   $a779       zur Formatierung...
```

```
------------------------------------------------------------------------
ff45                                    Routine für die Steppermotorsteuerung bei der Bewegung des
                                        Kopfes nach außen. Es wird hierbei laufend auf das
                                        Erreichen der Spur 0 getestet, um den Kopf rechtzeitig (vor
                                        dem Anschlag) stoppen zu können.
ff45   98           tya                 Inhalt des Y-Registers
ff46   48           pha                 retten
ff47   a0  64       ldy   #$64          100 Versuche; Verzögerung für Stepdauer
ff49   ad  0f  18   lda   $180f         Steuerport lesen
ff4c   6a           ror                 Bit für Nullanschlag-LED ins Carry
ff4d   08           php                 und Status merken
ff4e   ad  0f  18   lda   $180f         Steuerport lesen
ff51   6a           ror                 Bit für Nullanschlag-LED
ff52   6a           ror                 auf Position 7 schieben
ff53   28           plp                 und Status zurückholen
ff54   29  80       and   #$80          Bit 7 testen
ff56   90  04       bcc   $ff5c        verzweige, wenn Null beim ersten Test
ff58   10  1d       bpl   $ff77        verzweige, wenn kein Nullanschlag beim 2. Test
ff5a   30  02       bmi   $ff5e        unbedingter Sprung; Stepperbewegung abwarten
ff5c   30  19       bmi   $ff77        Spur Null erst beim zweiten Test erreicht
ff5e   88           dey                 Kopfbewegung noch nicht beendet?
ff5f   d0  e8       bne   $ff49        weitertesten, bis Kopf stillsteht
ff61   b0  14       bcs   $ff77        verzweige, wenn Nullanschlag nicht erreicht
ff63   ad  00  1c   lda   $1c00        Null erreicht; Kopfstatus prüfen
ff66   29  03       and   #$03          Bits isolieren
ff68   d0  0d       bne   $ff77        verzweige, wenn Stepper aktiv
ff6a   a5  7b       lda   $7b          sonst Testen auf Sonderstatus bei Kopfbewegung
ff6c   d0  09       bne   $ff77        verweise, wenn ja; Kopf weiter bewegen
ff6e   68           pla                 Y-Register
ff6f   a8           tay                 zurückholen
ff70   a9  00       lda   #$00          Null für Anzahl der Schritte
ff72   85  4a       sta   $4a          setzen und
ff74   4c  be  fa   jmp   $fabe        Kopfbewegung damit stoppen
ff77   68           pla                 Y-Register
ff78   a8           tay                 zurückholen
ff79   e6  4a       inc   $4a          Anzahl der Schritte plus 1 (entspricht minus 1)
ff7b   ae  00  1c   ldx   $1c00        Kopfstatus für Spulensteuerung holen
ff7e   ca           dex                 nächsten Teilschritt für Steppermotor setzen
ff7f   4c  38  fa   jmp   $fa38        zur Kopfsteuerung
------------------------------------------------------------------------
ff82                                    Floppy auf 1541-Modus stellen.
ff82   20  59  f2   jsr   $f259        Diskcontroller initialisieren
ff85   a9  05       lda   #$05          Schreib-/Leseabstand der Sektoren für
ff87   85  3c       sta   $3c          MFM wählen; bei 1541 nicht benutzt
ff89   a9  88       lda   #$88          Interruptverktor für den Aufruf
ff8b   8d  a9  02   sta   $02a9        der Jobschleife auf $9d88 für
ff8e   a9  9d       lda   #$9d          die Jobschleife der 1541
ff90   8d  aa  02   sta   $02aa        setzen
ff93   a9  24       lda   #$24          36 als maximale Sektoranzahl
*1 ff95 8d  ac  02   sta   $02ac        setzen
*0 ff95 8d  d7  fe   sta   $fed7        nicht setzen (ROM)
ff98   18           clc                 Flag für Seite 0 der Diskette
ff99   4c  f3  93   jmp   $93f3        setzen; Diskseite setzen; Ende
------------------------------------------------------------------------
ff9c                                    Drive für Betrieb vorbereiten.
ff9c   85  ff       sta   $ff          Drivestatus setzen
ff9e   4c  00  c1   jmp   $c100        LED einschalten; Ende
```

```
------------------------------------------------------------------------
ffa1                                    Kopf um einen Schritt nach innen oder außen bewegen.
ffa1   85  7b       sta   $7b          Wert für Kopfpositionierung merken
ffa3   4c  76  d6   jmp   $d676        Kopf bewegen; Ende
------------------------------------------------------------------------
ffa6                                    Kopf wieder auf Track positionieren; Wert für Verschiebung
                                        löschen.
ffa6   20  76  d6   jsr   $d676        Kopf wieder auf Track setzen
ffa9   a9  00       lda   #$00         Anzahl der 'Dejustierschritte'
ffab   85  7b       sta   $7b          löschen
ffad   60           rts                Ende
------------------------------------------------------------------------
ffae                                    Pufferzeiger für Kanalnummer setzen.
ffae   a4  82       ldy   $82         Kanalnummer holen
ffb0   4c  de  d3   jmp   $d3de        Pufferzeiger entsprechende setzen; Ende
------------------------------------------------------------------------
ffb3   ff ...                           Leerbereich;
ffe5   ... ff                           wird nicht benutzt.
------------------------------------------------------------------------
ffe6                                    Tabelle der Systemsprungvektoren.
ffe6 c6 c8                              Formatierung einer Diskette   $c8c6
ffe8 8f f9                              Laufwerksmotor ausschalten    $f98f
ffea 5f cd                              U1 UA Vektor; Block lesen     $cd5f
ffec 97 cd                              U2 UB Vektor; Block schreiben $cd97
ffee 00 05                              U3 UC Vektor; Userprogramm    $0500
fffO 03 05                              U4 UD Vektor; Userprogramm    $0503
fff2 06 05                              U5 UE Vektor; Userprogramm    $0506
fff4 09 05                              U6 UF Vektor; Userprogramm    $0509
fff6 0c 05                              U7 UG Vektor; Userprogramm    $050c
fff8 0f 05                              U8 UH Vektor; Userprogramm    $050f
fffa 01 ff                              U9 UI Vektor; NMI-Vektor      $ff01
fffc a0 ea                              U: UJ Vektor; RESET-Vektor    $eaaO
fffe 67 fe                              IRQ-Vektor;                   $fe67
------------------------------------------------------------------------
DOS V3.0 1571 (c) by Commodore Business Machines
ROM-Dokumentation: Copyright Oktober 1985 Karsten Schramm
```

D Die wichtigsten ROM-Adressen

Adresse	Inhalt
$8030	Behandlung der U0-Befehle
$806e	Jobcodes der U0-Routinen
$808e	Adressen der U0-Routinen
$80ce	ATN vom seriellen Bus empfangen (1571)
$8199	DRF senden
$81b2	1571-Bus auf Empfang schalten
$81ce	1571-Bus auf Senden schalten
$81ea	Ausgabe auf 1571-Bus
$823d	Ein Byte auf 1571-Bus ausgeben
$8276	Ein Byte auf 1541-Bus ausgeben
$82c7	Byte vom 1571-Bus holen (nach LISTEN)
$8342	Byte vom 1571-Bus in den aktuellen Puffer übernehmen
$8371	BURST-READ eines MFM-Sektors
$8394	Sektor von Diskette lesen
$83ec	BURST-WRITE eines Sektors vom Puffer auf Disk (MFM+GCR)
$848b	INQUIRE DISK: Initialisierung einer Disk (MFM+GCR)
$84b7	FORMAT-DISK: Formatierung einer Disk (MFM+GCR)
$84f1	SECTOR- INTERLEAVE
$8517	QUERY-DISK: Formatanalyse
$856b	INQUIRE-STATUS: Setzen oder Lesen des Burst-Status
$8581	Burst-Status auf Bus ausgeben
$8596	Übermittelten Burst-Status übernehmen
$85a5	BACKUP (nicht möglich)
$85af	Kopfpositionierung auf einen Track
$85f6	Byte in $46 auf 1571-Bus ausgeben
$861e	Berechnung der nächsten Sektornummer
$864b	Job ausführen mit Retry
$865e	LED ein; Job für Puffer 0 ausführen; LED aus
$869d	Warten auf Änderung des CLOCK-lN-Signals
$86b9	Tabelle der Steuerbytes für 1571-Diskettenzugriffe
$86c8	Adressentabelle für 1571-Steuerroutine
$86e6	Steuerroutine für MFM-Betrieb
$8764	Drivemotor ein
$8770	Drivemotor aus
$877c	LED einschalten
$8788	LED ausschalten
$87b0	Warten, bis Drive bereit ist
$87ba	Kopf auf neuen Track setzen
$87df	Kopf einen Step nach innen steuern
$87e7	Kopf einen Step nach außen bewegen
$8830	Verzögerung 1.3 ms
$883c	Status des MFM-Controllers holen
$884e	Kommando an MFM-Controller übergeben
$8861	Ausführung eines Kommandos abwarten
$886c	Nächster Sektor = Sektor + Interleave
$88f0	Verify eines MFM-Sektors bei der Formatierung
$891b	Track für Formatierung aus Befehlszeile holen
$892a	MFM-Sektorheader lesen mit Retry
$8954	Kopf auf Diskettenseite ($39) aktivieren
$8961	kleinste und größte Sektornummer aus Tabelle

$8986	Interleave aus Sektorfolge errechnen
$89b3	Init des MFM-Controllers auf einem Track
$89ef	BUMP: Kopf auf Track 0
$89fd	Test auf Write Protect
$8a09	Sektorheader für INQUlRE-DISK lesen
$8a27	MFM-Sektorheader lesen
$8a73	Tabellen für MFM-Betrieb
$8a86	Track in MFM formatieren
$8c57	Diskette in MFM formatieren
$8cde	Diskettenseite in MFM formatieren
$8d67	MFM-Sektor von Diskette lesen und ggf. senden
$8df6	BURST-WRITE: MFM-Sektor aus Puffer auf Disk
$8ec6	Verify MFM-Sektor mit Puffer 0
$8f18	MFM-Sektor neu formatieren mit Verify
$8f5f	Sektorfolge für QUERY -DISK feststellen
$8fa4	Routine U0 S: Interleave festlegen
$8faa	Routine U0 R: Leseversuche festlegen
$8fb0	Routine U0 T: ROM-Test durchführen
$8fb3	Routine U0 H: 1541-Diskettenseite wählen
$8fe5	Steuerroutine für die obigen Utilities
$9026	Routine U0 M: Umschaltung 1541/1571-Modus
$9080	FASTLOAD
$9199	Burst-Statusbyte an Computer senden
$91ad	Fehlerbehandlung bei Suche im Directory
$91ca	Internen Lesekanal suchen und belegen
$91da	Pufferzeiger für Job herstellen
$91ea	Befehlszeile auf Drivenummer und ":" testen
$9228	X-Register im FAST-Modus zum Rechner übertragen
$924e	Prüfsumme über das ROM berechnen
$92ba	Jobschleife für 1570/71-Betrieb
$93a2	User-Jobprogramm in Puffer starten
$93b0	BUMP: Kopf auf Track 0
$93d1	Pufferzeiger festlegen und zugehörigen Job holen
$93f3	Kopf auf aktueller Diskseite aktivieren
$9409	Tabelle für Aufzeichnungsrate/ Track
$942c	Anzahl Sektoren/ Track
$944f	SEEK: Sektorheader auf beliebigem Track suchen
$94bc	"Optimalen" Job suchen
$952f	Sektorheader von GCR nach Binär wandeln
$9600	SYNC nach Sektorheader abwarten
$9606	Job Lesen: GCR-Datenblock lesen
$970f	Sektorheader mit gegebenen Parametern suchen
$9754	SYNC abwarten
$976e	Job Schreiben: GCR-Block auf Disk schreiben
$97f9	Ausweichpuffer + akt. Puffer von GCR nach Binär
$9898	Job Verify: Pufferdaten mit Disk vergleichen
$98d9	5 GCR-Bytes in 4 Binärbytes wandeln
$9965	Ausweichpuffer $1bb-$1ff von GCR nach Binär
$99b5	Ausgang der 1571-Jobschleife
$99ca	Diskcontroller. Motorsteuerung
$9a56	Kopfsteuerung
$9ab3	Kopf um Halbspur nach innen
$9adb	Track für Formatierung ausmessen
$9b29	Disk in GCR formatieren
$9b73	Track mit $ff beschreiben
$9b89	Fortsetzung der Formatierroutine
$9d63	Füllbytes ($55) schreiben
$9d88	IRQ-Routine für 1541-Betrieb

$9dde	IRQ-Routine für 1571-Betrieb
$9f0d	Tabellen für Umwandlung GCR nach Binär
$a40d	Formatierung einer Diskette in GCR
$a47e	Verzögerung ca. 30 Taktzyklen
$a483	Verzögerung ca. 70 Taktzyklen
$a48b	Pufferzeiger für BAM retten
$a496	Pufferzeiger für BAM wieder holen
$a4e7	BAM-Belegung eines Sektors feststellen (Seite 2)
$a508	Freie Blöcke eines Tracks nach Freigabe erhöhen
$a51e	Freie Blöcke eines Tracks nach Belegung vermindern
$a534	Pufferzeiger in BAM setzen
$a545	BAM der 2. Seite auf Richtigkeit testen
$a58d	BAM auf Diskette schreiben
$a5db	Tabelle der Positionen der BAM-Einträge/ Track
$a5fe	Erweiterung des &-Befehls für zweiseitige Disks
$a62b	Ausschaltverzögerung für Drivemotor setzen
$a634	Drivemotor ein, wenn Diskette gewechselt wurde
$a65a	Init des Buscontrollers bei RESET
$a667	BAM von Disk lesen
$a6e5	Diskette auf zweiseitige Formatierung prüfen
$a738	BAM-Puffer für 1571-BAM löschen
$a74f	Tracknummern größer 35 in Nummern für Seite 2 wandeln
$a764	Neue BAM im Speicher herstellen
$a779	GCR-Formatierung einer Disk ohne Directory
$a786	RESET für seriellen Bus
$a7b3	ATN-Bedienung des Bus (1571)
$a7c0	Fortsetzung der RESET-Routine
$a7c7	Neue BAM in Speicher generieren (Erweiterung)
$a83e	Block in BAM freigeben
$a874	Block in BAM belegen
$a8a9	Nächsten freien Sektor eines Tracks suchen
$a905	Bitmuster für Sektor aus BAM holen
$a91e	Nächsten freien Sektor feststellen
$a937	Gültigkeit der BAM prüfen
$a951	Freie Blöcke einer Diskette errechnen
$a97f	Blockanzahl einer Datei löschen
$a989	Diskette im 1571-Format formatieren
$a99d	Drivestatus "OK" setzen
$a9a5	Drivestatus auf Y setzen
$a9ac	Serielle Busbedienung nach ATN
$a9b9	Erweiterte Fehlerbearbeitung mit Burst-Status und Klartext
$a9d5	Fehlerbehandlung bei LOAD
$a9f8	Drivestatus auf "inaktiv"
$aa17	Drivestatus holen
$aa25	Flags für Drivestatus holen
$aa2d	USER-Befehle außer U0 ausführen
$aa3f	Fehler nach Job prüfen
$aa4d	Diskette im 1570-Format formatierren
$aa5b	Dateityp auf PRG testen
$aa62	Steuerport lesen
$bf00	Sprungtabelle

Verändertes, aber im wesentlichen erhalten geblietenes 1541-ROM:

$c100	LED am aktuellen Laufwerk einschalten
$c118	LED an Laufwerk 0 ein
$c123	Fehlerstatus-Flags löschen
$c12c	LED-Blinken initialisieren
$c146	Befehlsstring auswerten
$c19d	BefehlsabschluB: OK-Meldung
$c1bd	INPUT-Puffer $200-$228 löschen
$c1c8	Fehler ausgeben; Nummer in A
$c1d1	Befehlsstring auf ":" untersuchen
Sc1ee	Befehlsstring auf Joker untersuchen
$c268	Befehlsstring auf Zeichen in A untersuchen
$c2b3	Init der Kommandotabellen und -zeiger
$c312	Laufwerksnummer feststellen und setzen
$c33c	Drivenummer aus Befehlsstring holen
$c38f	Drivenummer ($7f) umdrehen
$c398	Zeiger für Dateinamen setzen, Dateityp holen
$c3bd	Test auf vorhandene Drivenummer
$c3ca	Vorbereitung zur Suche einer Datei
$c44f	Dateien aus dem Befehlsstring suchen
$c48b	nächsten Dateinamen im Directory suchen
$c4d8	Vergleich Dateieintrag im Directory/ im Befehlsstring
$c5ac	Vorbereitung der Suche im Directory
$c63d	Test auf Diskette im Laufwerk
$c66e	Dateiname aus Befehlsstring in Directory-Puffer
$c688	INPUT-Puffer in anderen Puffer kopieren
$c6a6	Ende des Dateinamens im Befehlsstring suchen
$c6ce	Dateieintrag aus Directory über Lesekanal 17 holen
$c7ac	Namen im Directory löschen
$c7b7	Kopf des Directory für Ausgabe erzeugen
$c806	Zeile mit "BLOCKS FREE" erzeugen
$c823	SCRATCH-Befehl
$c8b6	Dateieintrag im Directory löschen
$c8c1	BACKUP-Befehl (nicht möglich)
$c8f0	COPY-Befehl
$c932	Parameter für Kopieren einer Datei setzen
$c952	Datei in eine Datei kopieren
$c9fa	Internen Kanal zum Lesen einer Datei öffnen
$ca35	Byte über internen Lesekanal 17 holen
$ca53	Routine zum Kopieren relativer Dateien
$ca88	RENAME-Befehl
$cacc	Test auf Vorhandensein einer Datei
$cae7	Test auf Namensgleichheit zweier Dateien
$caf8	MEMORY-Befehle (Steuerung)
$cb20	M-R-Befehl
$cb50	M-W-Befehl
$cb5c	USER-Befehle (Steuerroutine)
$cb84	"#": Öffnen eines Direktzugriffs-Kanals
$cc1b	BLOCK-Befehle (Steueroutine)
$cc63	Adressen der Blockbefehle
$cc6f	Parameter für Blockbefehle holen
$ccal	ASCII aus INPUT-Puffer nach HEX wandeln
$ccf5	B-F-Befehl
$cd03	B-A-Befehl (fehlerhaft)
$cd56	B-R-Befehl
$cd5f	U1-Befehl

$cd73	B-W-Befehl
$cd97	U2-Befehl
$cda3	B-E-Befehl
$cdbd	B-P-Befehl
$cdd2	Direktkanal öffnen; Puffer belegen
$cdf2	Test auf legalen Block und vorhandenen Puffer
$ce0e	Suchen eines Blocks in einer relativen Datei
$ce2c	Position eines Records berechnen
$ce6e	Divisionsroutine
$cf1e	Aktiven Puffer für Diskbetrieb setzen/ neuen suchen
$cf7b	Freien Puffer suchen
$cf8c	Betriebszustand eines Puffers umkehren
$cf9b	Schreiben einer Datei über internen Kanal in Puffer
$cffl	Ein Byte in den aktuellen Puffer schreiben
$d005	INITIALISE-Befehl
$d042	BAM lesen und aktualisieren
$d075	Anzahl freier Blöcke einer Diskette berechnen
$d09b	Lesen eines Blocks
$d0af	Folgeblock einer Verkettung holen
$d0c3	Job: Block lesen (Einsprung)
$d0eb	Suchen und Öffnen eines Lesekanals
$d107	Suchen und Öffnen eines Schreibkanals
$d125	Aktuellen Dateityp auf REL prüfen
$d12f	Puffer- und Kanalnummer holen/ setzen
$d156	Byte aus Datei holen mit Nachladen von Disk
$d19d	Byte in Puffer schreiben mit Schreiben auf Disk, wenn Puffer voll ist
$d1c6	Erhöhen des aktuellen Pufferzeigers
$d1d3	Holen und Setzen der Laufwerksnummer
$d1df	Suchen einer Kanalnummer und des zugehörigen Puffers
$d1df	Schreibkanal
$d1e2	Lesekanal
$d227	Freigeben aller Kanäle außer dem Kommandokanal
$d25a	Puffer und zugeh. Kanal freigeben
$c28e	Suchen eines Puffers
$d2ba	Suchen eines freien Puffers
$d37f	Freien Kanal suchen und belegen
$d39b	Nächstes Byte eines Kanals holen
$d3aa	Nächstes Byte einer Datei holen
$d414	Byte aus Fehlerkanallesen
$d44d	Nächsten Block einer Datei lesen und EOF setzen
$d460	Parameterübergabe an Diskcontroller
$d475	Öffnen eines Lesekanalsj Typ SEQ; Block lesen
$d486	Internen Lesekanal öffnen
$d48d	Belegen und Schreiben des nächsten Directory-Blocks
$d4c8	Aktuellen Pufferzeiger auf Wert in A setzen
$d4da	Beide internen Kanäle (Lesen + Schreiben) schließen
$d4e8	Aktuellen Pufferzeige setzen
$d4f6	Byte aus aktuellem Puffer lesen (A=Nummer des Bytes)
$d506	Track- und Sektornummer für Jobschleife prüfen
$d552	Track- und Sektornummer für Job holen
$d55f	Test auf legale Track- und Sektornummer
$d572	Track- und Sektornummer für "DOS MISMATCH" holen
$d576	Kopfpositionierung relativ zur aktuellen Position
$d57a	Job für einen Puffer setzen
$d586	Jobcodes an Diskcontroller übergeben
$d586	Job Lesen
$d58a	Job Schreiben
$d599	Warten auf Ende des Jobs

$d5a6	Test auf Fehler nach einem Job
$d693	Kopfbewegung um einen Track nach innen oder außen
$d6a6	Steuerung der Anzahl Leseversuche
$d6d0	Parameter für nächsten Job übergeben
$d6e4	Neue Datei in Directory eintragen
$d7b4	OPEN vom seriellen Bus empf. (LOAD,SAVE,SEQ,USR,REL)
$d8f5	Öffnen einer Datei zum Überschreiben
$d7a0	Öffnen einer Datei zum Lesen
$d9e3	Datei zum Schreiben öffnen
$da09	Dateibetriebsart testen/ Parameter setzen
$da2a	APPEND vorbereiten/ Datenteil überlesen
$da55	Directory von Disk lesen
$dac0	Datei mit gegebener Sekundäradresse schließen
$daec	Alle Dateien schließen, wenn Kanal 15 geschlossen wird
Sdb02	Datei mit bestimmter Sekundäradresse schließen
$db62	Letzten Block einer Datei schreiben
$dba5	Directory nach Schreiben einer Datei aktualisieren
$dc46	Kanal mit zwei Puffern zum Lesen öffnen
$dcb6	Parameter zum Öffnen eines Kanals setzen
$dcda	Kanal mit 2 Puffern zum Schreiben öffnen
$dd8d	Byte in Side-Sektor schreiben
$dda6	Dateityp-Flags testen
$ddab	Jobcode auf "Schreiben" prüfen
$ddb7	Test auf aktive Datei
$ddf1	Block schreiben, wenn Puffer "dirty"
$ddfd	Linker für Folgeblock in Puffer schreiben
$de0c	Linker für Folgeblock aus Puffer holen
$de19	Linker für letzten Block setzen
$de2b	Aktuellen Pufferzeiger setzen
$de3b	Track- und Sektornummer nach $80/ $81 bringen
$de50	Schreib- und Lesejobs ausführen
$de95	Nächsten Block einer Datei aus Linker schließen
$dea5	Kopieren eines Puffers in einen anderen
$dec1	Puffer löschen
$ded2	Side-Sektor-Nummer holen
$dedc	Pufferzeiger auf Side-Sektor setzen
$def8	Side-Sektor und Pufferzeiger setzen
$df1b	Jobübergabe an Diskcontroller (A=Puffernummer; X=Nr. aktiver Puffers)
$df4c	Größe einer relativen Datei berechnen
$df66	Zustand eines Side-Sektors im Puffer prüfen
$df93	Nummer des aktiven Puffers holen
$df9e	Aktiven Puffer auf Belegung testen
$dfb7	Nummer eines inaktiven Puffers + Kanalnummer holen
$dfc2	Puffer in A freigeben
$dfd0	Nächsten Record einer relatien Datei herstellen
$e03c	Nächsten Record im Puffer generieren
$e07c	Byte in Record-Puffer schreiben
$e0ab	Schreiben eines Records
$e0f3	Recordrest mit Nullen auffüllen
$e105	Puffer "dirty" anzeigen
$e115	Puffer "dirty" zurücksetzen
$e120	Byte aus Recordpuffer holen
$e153	Nächsten Record lesen/ Byte für Ausgabe bereitmachen
$e16e	Letztes Byte im Record markieren
$e1b2	Letztes Recordbyte ungleich Null suchen
$e1cb	Ende des letzten Records feststellen
$e207	POSITION - Befehl
$e257	Record in aktuellen Puffer holen; Folgeblock laden

$e29c	Datenblock in Puffer schreiben
$e2e2	Pufferdaten einzelnen Records zuordnen
$e304	Paßt nächster Record noch in Puffer?
$e31c	Blöcke an relative Datei anhängen
$e44e	Neuen Side-Sektor anlegen
$e4fc	Felermeldungen (ASCII-Tabelle)
$e60a	Fehlerbehandlung nach Job (A==Fehlernummer; X==Puffer)
$e680	Fehler nach TALK vom Bus
$e688	Fehler nach LISTEN
$e69b	Hex in A nach dezimal wandeln
$e6bc	Fehlermeldung in Fehlerpuffer schreiben
$e754	Test auf Kontroll- und ASCII-Codes
$e775	Byte aus Fehlertabelle holen/ Zeiger erhöhen
$e7a3	&-Befehl (UTILITY LOADER)
$e853	IRQ-Routine unter Berücksichtigung eines ATN
$e85b	Bedienung des Bus nach ATN
$e909	Senden von Daten nach TALK
$e99c	DATA OUT auf Hi
$e9a5	DATA OUT auf Lo
$e9ae	CLOCK OUT auf Lo
$e9b7	CLOCK OUT auf Hi
$e9c0	Warten auf Antwort vom Bus
$e9c9	Byte vom Bus holen nach LISTEN
$ea2e	Daten vom seriellen Bus in aktuellen Puffer schreiben
$ea59	Test auf ATN-Modus
$ea6e	Behandlung von Hardwarefehlern
$eaa0	RESET-Routine
$ebe7	Systemwarteschleife
$ec9e	Laden und Aufbereiten des Directory
$ed23	Abschluß des Directory
$ed59	Directoryzeile in Ausgabepuffer schreiben
$ed67	Byte aus Directory holen mit Nachladen von Disk
$ed84	VALIDATE-Befehl
$edb3	Verkettung eines Directoryeintrags nachvollziehen
$ede5	Dateiaufbau anhand der Verkettung nachvollziehen
$eeb7	Neue BAM erzeugen
$eef4	BAM auf Diskette schreiben, wenn "dirty"
$ef3a	BAM lesen und Pufferzeiger setzen
$ef4d	Anzahl freier Blöcke holen
$ef5c	Block in BAM freigeben
$ef88	Flag für "BAM dirty" setzen
$ef90	Block in der BAM als belegt kennzeichnen
$efcf	Index des BAM-Eintrags eines Blocks errechnen
$eff1	AAM, falls notwendig, auf Disk schreiben
$f005	Puffer für BAM löschen
$f011	BAM-Maske in Puffer erzeugen
$f05b	BAM-Masken im Puffer vertauschen
$f0a5	BAM-Maske in richtige Position bringen
$f0df	BAM lesen, falls notwendig
$f10f	Kanalnummer für BAM nach A holen
$f11e	Nächsten freien Sektor nach aktuellem ermitteln
$f173	Nächsten "optimalen" Sektor einer Spur feststellen
$f1a9	Nächsten "optimalen" Sektor suchen und belegen
$f1fa	Nächsten freien Sekor einer Spur suchen
$f220	Gültigkeit der BAM testen
$f24b	Maximalzahl der Sektoren einer Spur ermitteln
$f259	Register-Init für Diskcontroller
$f2b0	IRQ-Routine für Diskcontroller

$f37c	BUMP-Ausführung
$f393	Pufferadresse und -zeiger für Job setzen
$f3b1	Suchen einer Spur (SEEK)
$f423	Suchen des besten Folgejobs
$f497	Headerbytes von GCR nach Binär wandeln
$f4ca	Job Lesen: Blockheader suchen, Block lesen
$f510	Blockheader suchen
$f556	Warten auf SYNC
$f56e	Job Schreiben: Block aus Puffer auf Disk schreiben
$f5e9	Prüfsumme über Datenblock bilden
$f5f2	Ausweich- und Datenpuffer von GCR nach Binär wandeln
$f691	Job Verify: Vergleich aktueller Puffer mit Disk
$f6d0	4 Binärbytes in 5 GCR-Bytes wandeln
$f77f	Tabelle für Wandlung von Binär nach G CR
$f78f	Aktiven Puffer von Binär nach GCR wandeln
$f7e6	5 GCR-Bytes in 4 Binärbytes wandeln
$f8a0	Tabellen für die Wandlung GCR nach Binär
$f8e0	GCR in Ausweichpuffer nach Binär im aktuellen Puffer
$f934	Aktueller Blockheader nach GCR
$f969	Ausgang aus der Jobschleife
$f97e	Drivemotor an + 0.4 sAnIaufzeit
$f98f	Drivemotor aus + 12.3 s Nachlaufzeit
$f99c	Steuerroutitne des Diskcontrollers (1541)
$fa05	Init der Steppermotor-Steuerung
$fa3b	Langsamer Steppermodus für kurze Wege
$fa4e	Kopfpositionierung beenden
$fa63	Service-Routine für Steppermotor
$fa7b	Anfahren Steppermotor in schnellen Raten
$fa97	Steuerroutine für schnellen Steppermodus
$faa0	Abbremsen des Steppermotors
$fac7	Jobroutine: Formatierung im 1541-Format
$fda3	Spur löschen durch Beschreiben mit Sir-Bytes
$fdc3	Anzahl BYTE-READY-Signale abwarten
$fdf5	Inhalt des Ausweichpuffers in den aktuellen Puffer
$fe00	Diskcontroller auf Lesen schalten
$fe0e	Track mit $55 beschreiben
$fe30	Blockheader von Binär nach GCR wandeln
$fe67	Tabellen
$feea	LED einschalten
$fef3	Verzögerung für seriellen Bus
$fefb	CLOCK OUT auf Lo; DATA OUT auf Hi
$ff0l	UI-Befehl: Umschaltung auf 1540-Betrieb
$ff10	RESET für Datenrichtungsregister ABuscontroller
$ff18	RESET für Datenrichtungsregister B Buscontroller
$ff20	DATA IN Hi abwarten
$ff2f	Vorbereitung der Formatierung einer Diskette
$ff45	Steppermotorsteuerung für Bewegung nach außen
$ff82	Floppy auf 1541-Modus stellen
$ff9c	Drive für Betrieb vorbereiten
$ffa1	Kopf einen Schritt nach innen oder außen
$ffe6	Kopf wieder auf Track positionieren
$ffae	Pufferzeiger für Kanalnummer setzen
$ffe6	Sprungvektoren

E Die Statusmeldungen der 1570/71

Nachfolgend eine komplette Liste aller möglichen Status- und Fehlermeldungen der 1570/71. Diese Meldungen bestehen grundsätzlich aus einer Nummer mit der darauf folgenden Klartextmeldung und zwei angehängten Werten, die sich in der Regel auf den Track und den Sektor beziehen, bei dem der Fehler aufgetreten ist. Es gibt jedoch auch ein paar andere Anzeigemöglichkeiten, weshalb die folgende kleine Aufstellung ein paar Variablen zeigt und wofür diese stehen:

tt Tracknummer
ss Sektornummer
ff Anzahl der Files

00 ok,00,00 (alles in Ordnung)

Diese Meldung wird vom DOS ausgegeben, wenn ein Befehl ordnungsgemäß ausgeführt worden ist.

01 files scratched,ff ,00 (keine Fehlermeldung)

Gibt an, wie viele Files auf der Diskette durch den gegebenen Befehl gelöscht wurden.

20 read error,tt,ss (Blockheader nicht gefunden)

Der BIockheader eines Sektors konnte vom Diskcontroller nicht ausfindig gemacht werden. Das liegt entweder an einer ungültigen Sektornummer, einer zerstörten Diskette oder einem geänderten Headerkennzeichen in der Zeropageadresse $0039.

21 read error,tt,ss (SYNC-Markierung nicht gefunden)

Hier wurde die SYNC-Markierung auf einer Spur nicht innerhalb der gegebenen Zeit gefunden. Dieser Fehler tritt bei einer defekten Diskette oder einem dejustierten Schreib- /Lesekopf auf.

22 read error,tt,ss (Datenblock nicht gefunden)

Es wurde der normalerweise nach dem Blockheader folgende Datenblock nicht gefunden. Dieser Fehler tritt auf, wenn zum Beispiel das Datenblock

kennzeichen der Diskette nicht mit dem Wert in der Zeropageadresse $0047 übereinstimmt.

23 read error,tt,ss (Prüfsummenfehler im Datenblock)

Die auf der Diskette abgespeicherte Prüfsumme über den Datenblock stimmt nicht mit der errechneten des DOS überein. Der Serktor konnte zwar in den DOS-Puffer gelesen werden; es besteht jedoch die Gefahr einer defekten Diskette. Diese Fehlermeldung kann auch Erdungsprobleme der Floppy anzeigen.

24 read error,tt,ss (Fehler bei GCR-Codierung)

Beim Recodieren eines eingelesenen Datenblocks oder Blockheaders in das Binärformat sind unerlaubte Bitkombinationen der GCR-Bytes aufgetreten; zum Beispiel mehr als 8 1-Bits oder mehr als 2 0-Bits direkt hintereinander. Bei dieser Fehlermeldung handelt es sich zumeist um Erdungsprobleme der Floppy, die Lesestörungen verursachen.

25 write error,tt,ss (Fehler bei Verify)

Das anschließende Verify eines Sektors nach dem Schreiben ist fehlerhaft verlaufen. Es besteht deshalb der Verdacht auf einen Schreibfehler. Diese Fehlermeldung zeigt normalerweise eine defekte Diskette an.

26 write protect on,00,00 (Schreibschutz aktiviert)

Es wurde versucht, auf eine Diskete zu schreiben, die eine Schreibschutzplakette trägt.

27 read error,tt,ss (Prüfsummenfehler im Blockheader)

Es wurde beim Einlesen eines Blockheaders ein Prüfsummenfehler festgestellt. Hier handelt es sich sehr wahrscheinlich um eine defekte Diskette oder Erdungsprobleme der Floppy, die sich in Lesestörungen äußern.

28 write error,tt,ss (zu langer Block)

Hier wurde nach dem Schreiben eines Datenblocks nicht in der festgelegten Zeit die SYNC-Markierung des folgenden Blockheaders gefunden. Diese Meldung zeigt eine fehlformatierte Diskette (Datenblock hat die folgende SYNC-Markierung überschrieben) oder einen Hardwaredefekt der Floppy an.

29 disk id mismatch,tt,ss (falsche ID im Blockheader)

Es wurde auf eine Diskette zugegriffen, die vorher nicht initialisiert worden ist. Bei dieser Fehlermeldung kann es sich aber auch um die Folge eines zerstörten Blockheaders (Kopierschutz) handeln.

30 syntax error,00,00 (allgemeiner Syntaxfehler)

Hier konnte ein vom Computer geschicktes Kommando nicht interpretiert werden, weil entweder die Parameterübergabe falsch war, oder das gegebene Kommando nicht existiert.

31 syntax error,00,00 (ungültiger Befehl)

Das DOS war hier nicht in der Lage, einen gesendeten Befehl zu interpretieren. Das kann unter Umständen auf Leerzeichen vor dem Befehlsnamen hindeuten. Ein Befehl muß immer am Anfang eines Befehlsstrings stehen.

32 syntax error,00,00 (Befehlszeile zu lang)

Der gesendete Befehlsstring vom Computer paßt nicht in den INPUT-Puffer der Floppy, weil er zu lang ist. Die Länge darf 42 Zeichen nicht überschreiten.

33 syntax error,00,00 (unerlaubte Verwendung eines Jokers)

Es wurde ein Joker (*,?) in einem Dateinamen verwendet, obwohl das für den gegebenen Befehl nicht zulässig ist.

34 syntax error,00,00 (Dateiname nicht gefunden)

Das DOS konnte in der Eingabezeile keinen Dateinamen entdecken. Hierfür ist das Fehlen eines Doppelpunktes ':' verantwortlich.

39 file not found,00,00 (Dateityp USR nicht gefunden)

Bei dem Befehl, eine &-Datei zu starten, wurde keine USR-Datei mit dem gegebenen Dateinamen gefunden.

50 record not present,00,00 (Datensatz nicht gefunden)

Diese Fehlermeldung zeigt an, daß in einer relativen Datei ein Datensatz nicht gefunden worden ist. Ist der folgende Befehl ein Schreibzugriff, so kann diese Fehlermeldung ignoriert werden, da die Erweiterung der relativen Datei automatisch vorgenommen wird.

Bei einem &-Befehl zeigt diese Fehlermeldung eine verkehrte Prüfsumme über den Programmteil an.

51 Overflow in record,00,00 (Record für Datensatz zu klein)

Hier wurde versucht, einen längeren Datensatz in einer relativen Datei unterzubringen, als bei der Eröffnung der Datei festgelegt wurde.

Bei einem &-Befehl zeigt diese Meldung eine verkehrte Längenangabe für einen Programmteil an.

52 file too large,00,00 (Datei zu groß)

Eine Erweiterung einer relativen Datei ist nicht mehr möglich, da diese Erweiterung die Kapazität der Diskette oder des Side-Sektors überschreiten würde.

60 write file open,00,00 (offengeblieben Datei)

Es wurde versucht, auf eine nicht ordnungsgemäß geschlossene Datei einen Lesezugriff durchzuführen.

61 file not open,00,00 (Datei noch nicht geöffnet)

Es wurde versucht, auf eine noch nicht geöffnete Datei zuzugreifen.

62 file not found,00,00 (gewünschte Datei nicht gefunden)

Die zum Öffnen angeforderte Datei ist unter diesem Namen oder unter diesem Dateityp auf der eingelegten Diskette nicht vorhanden.

63 file exists,00,00 (Datei bereits vorhanden)

Es wurde versucht, ein neue Datei mit einem Dateinamen anzulegen, der bereits auf der Diskette exisitert.

64 file type mismatch,00,00 (falscher Dateityp)

Die Angabe des Dateityps im Dateinamen stimmt nicht mit dem Dateityp auf der Diskette überein.

65 no block,tt,ss (zu belegender Block ist bereits belegt)

Hier wurde versucht, einen Block mit dem B-A-Befehl zu belegen, der bereits in der BAM als belegt gekennzeichnet ist. Die Spur- und Sektorangabe zeigen den nächsten verfügbaren freien Block an.

66 illegal track or sector,tt,ss (unerlaubte Sektorangabe)

Es wurde versucht, auf einen Sektor zuzugreifen, dessen Nummer auf der Diskette nicht existiert.

67 illegal track or sector,tt,ss (unerlaubte Sektorangabe)

In einer Datei zeigt der Linker eines Datenblocks auf einen nächsten Datenblock, der auf der Diskette nicht existiert.

70 no channel,00,00 (kein Kanal mehr frei)

Es wurde versucht, einen Kanal zu belegen, der bereits belegt ist, oder es wurde versucht, einen Kanal zu belegen, obwohl schon alle Kanäle belegt sind.

71 dir error,tt,ss (Fehler in der BAM)

Hier stimmt entweder die errechnete Summe freier Blöcke mit der Angabe in der BAM nicht überein, oder die BAM-Angaben widersprechen sich innerhalb der BAM. Dieser Fehler zeigt in der Regel eine zerstörte oder eine veränderte und nicht wieder auf die Diskette zurückgeschriebene BAM an.

72 disk full,00,00 (Diskette voll)

Diese Meldung erscheint entweder, wenn nur noch weniger als drei Blöcke auf der Diskette frei sind, oder wenn das Directory mit 144 Einträgen voll ist.

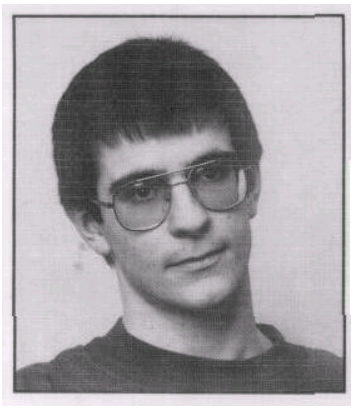
73 cbm dos v3.0 1571,00,00 (Einschalt- und Fehlermeldung)

Diese Meldung tritt nach dem neuen Einschalten der Floppystation auf oder, wenn versucht wurde, auf eine Diskette anderen Commodore-Formats zu schreiben.

74 drive not ready,00,00 (keine Diskette im Laufwerk)

Es befindet sich entweder keine oder keine formatierte Diskette im Laufwerk der Floppystation.

Stichwortverzeichnis



KARSTEN SCHRAMM wurde am 1. 3. 1966 in München geboren. Zur Zeit besucht er noch das Gymnasium. Vor fünf Jahren kam er das erste Mal mit Mikrocomputern in Kontakt. Diese Geräte haben ihn derart fasziniert, daß er ein begeisterter Hobbyanwender von Mikrocomputern geworden ist.
Seine Vorliebe gilt der Programmierung in Maschinensprache, wobei er sich besonders für Floppystationen interessiert.

DIE FLOPPY 1570/1571

Die Floppy 1541 ist unter Commodore-64-Besitzern hinreichend bekannt. Mit Erscheinen des Commodore 128 PC sind weitere externe Datenspeicher verfügbar - die 5¼ -Zoll-Diskettenlaufwerke 1570 und 1571. Gegenüber der Floppy 1541 bieten sie ein völlig neues Floppy-Konzept wie z.B.

- eine höhere Verarbeitungsgeschwindigkeit
- das Lesen von Standard-Diskettenformaten
- Burst-Betriebsarten für schnellen seriellen Busbetrieb.

Wer diese Vorteile für die eigene Programmierung nutzen will, findet hier zum Nachschlagen ein umfassend dokumentiertes ROM-Listing des DOS-Betriebssystems. Neben der systematischen Beschreibung der wichtigsten Hardwarekomponenten der Floppy 1570/1571 wird ein leistungsfähiger Diskettenmonitor als Beispielprogramm zum Abtippen mitgeliefert. Für Einsteiger und Profis ist es ein nützliches Utility, um in die Geheimnisse der Floppy 1570/1571 einzudringen.

Aus dem Inhalt:

- Beschreibung der verschiedenen Diskettenformate
- Die DOS-Befehle
- Erläuterung der unterschiedlichen Betriebsarten (GCR, MFM, Burst)
- Ansprechen der Schnittstellenbausteine (VIA 6522, CIA 6526 etc.)
- Programmierung des DOS in Maschinensprache
- Bekannte DOS-Fehler
- Dateiverwaltung
- Behandlung und Aufbau von Disketten
- Tips und Tricks für die Pflege und Wartung

Besonders wichtig für die 64er-Anwender: Die Floppy 1570/1571 kann in der schnellen Betriebsart auch mit dem C 64 genutzt werden.

Hardwareanforderung:
C 128 oder C 128 D, Floppy 1570/1571

Markt&Technik Verlag Aktiengesellschaft
ISBN 3-89090-185-9

DM 52,—
sFr..47,80 / öS 405,60