Karsten Schramm

# Die Floppy 1541

Alles über die Programmierung
der VC 1541 vom Eröffnen einer
Datei bis zu Eingriffen in die
Arbeitsweise des DOS.
Lernen Sie Ihren eigenen
Programmierschutz und Ihr
schnelles Ladeprogramm selbst
zu entwickeln.

Mit komplettem kommentierten ROM-Listing

Ein Markt & Technik Buch

# 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.

Alle enthaltenen Dateilistings in Anhang V wurden mit PETCAT (aus dem VICE Paket, http://www.viceteam.org/) aus Original-Dateien erzeugt.

Das ROM-Listing in Anhang II 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.

Karsten Schramm

# Die Floppy 1541

Alles über die Programmierung der VC 1541 vom Eröffnen einer Datei bis zu Eingriffen in die Arbeitsweise des DOS.
Lernen Sie Ihren eigenen Programmierschutz und Ihr schnelles Ladeprogramm selbst zu entwickeln.

Markt & Technik Verlag

CIP-Kurztitelaufnahme der Deutschen Bibliothek

**Schramm, Karsten:**
Die Floppy 1541: alles über d. Programmierung d. VC 1541
vom Eröffnen e. Datei bis zu Eingriffen in d. Arbeitsweise d. DOS /
Karsten Schramm. —
Haar bei München: Markt-und-Technik-Verlag, 1985.
ISBN 3-89090-098-4

Die Informationen im vorliegenden Buch werden ohne Rücksicht auf einen eventuellen Patentschutz veröffentlicht.
Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt.
Bei der Zusammenstellung von Texten und Abbildungen wurde mit größter Sorgfalt vorgegangen.
Trotzdem können Fehler nicht vollständig ausgeschlossen werden. Verlag, Herausgeber und Autoren können
für fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine
Haftung übernehmen.
Für Verbesserungsvorschläge und Hinweise auf Fehler sind Verlag und Herausgeber dankbar.

Alle Rechte vorbehalten, auch die der fotomechanischen Wiedergabe und der Speicherung in elektronischen Medien.
Die gewerbliche Nutzung der in diesem Buch gezeigten Modelle und Arbeiten ist nicht zulässigig.

»VC 1541« ist eine Produktbezeichnung der Commodore Büromaschinen GmbH, Frankfurt, die ebenso wie der Name
Commodore Schutzrecht genießt. Der Gebrauch bzw. die Verwendung bedarf der Erlaubnis
der Schutzrechtsinhaberin.

15 14 13 12 11 10 9 8 7 6 5 4 3 2

89 88 87 86 85

ISBN 3-89090-098-4

© 1985 by Markt & Technik, 8013 Haar bei München
Alle Rechte vorbehalten
Einbandgestaltung: Grafikdesign Heinz Rauner
Druck: Schoder, Gersthofen
Printed in Gennany

# Vorwort

Lieber Leser,

mit diesem Buch halten Sie ein Werk in Ihren Händen, das Ihnen die Programmierung der Floppystation 1541 nahebringen soll.

Sind Sie Anfänger auf dem Gebiet, erfahren Sie alles Wichtige, das Sie zum Verständnis der Funktionsweise der Floppy benötigen. Sind Sie schon fortgeschrittener Programmierer, erhalten Sie mit diesem Buch ein leistungsfähiges Nachschlagewerk, das Ihnen in vielen Situationen und Fragen weiterhelfen wird.

Die grundlegende Bedienung der Floppy, das heißt, die Handhabung des Ladens und Speicherns von Programmen sollte Ihnen schon vertraut sein. Dieses Buch wird Sie dann in Techniken einführen, die es Ihnen erlauben, Ihre Floppy effektiv einzusetzen und schließlich sogar eigene Eingriffe in den Ablauf des DOS vorzunehmen.

Ich wünsche Ihnen nun viel Spaß bei der Lektüre dieses Buches und bei der Erforschung der 1541 und deren Möglichkeiten.

Karsten Schramm

# Inhaltsverzeichnis


**1 Die Floppy 1541 ..... 11**
1.1 DOS -Was ist das? ..... 13
1.2 Kontakt mit der 1541 ..... 14
1.3 Der Kommandokanal ..... 15

**2 Datenspeicherung mit der 1541 ..... 21**
2.1 Das Inhaltsverzeichnis auf Diskette ..... 23
2.2 Programm-(PRG)-Files auf der 1541 ..... 24
2.3 Die sequentielle (SEQ) Datenspeicherung ..... 25
2.4 Datenspeicherung in USR-Files ..... 28
2.5 Datenspeicherung mit relativen (REL) Files ..... 29

**3 Das Diskettenformat der 1541 ..... 33**
3.1 Der Aufbau einer neu formatierten Diskette ..... 35
3.1.1 Aufbau einer Spur ..... 37
3.1.2 Aufbau eines Sektors ..... 38
3.1.3 Verkettung von Sektoren ..... 38
3.2 Aufbau der BAM ..... 39
3.2.1 Das Formatkennzeichen ..... 40
3.2.2 Die Block Availability Map ..... 41
3.2.3 Der Name einer Diskette ..... 42
3.2.4 Die ID einer Diskette ..... 43
3.3 Aufbau des Directory ..... 43
3.3.1 Fileeinträge im Directory ..... 44
3.4 Aufbau von Programm-(PRG)-Files ..... 48
3.5 Aufbau von sequentiellen (SEQ) Files ..... 48
3.6 Aufbau von User-(USR)-Files ..... 49
3.7 Aufbau von relativen (REL) Files ..... 49
3.7.1 Aufbau der Datenblöcke ..... 50
3.7.2 Aufbau der Side-Sektor-Blocks ..... 50
3.8 Aufbau von deleted (DEL) Files ..... 52

**4 Direktzugriff auf die Floppy 1541 ..... 55**
4.1 Was ist Direktzugriff? ..... 57
4.2 Die Direktzugriffsbefehle ..... 57
4.2.1 Öffnen eines Direktzugriffskanals ..... 57
4.2.2 Der BLOCK-READ-(B-R)-Befehl ..... 59
4.2.3 Der BUFFER-POINTER-(B-P)-Befehl ..... 61
4.2.4 Der BLOCK-WRITE-(B-W)-Befehl ..... 61
4.2.5 Der BLOCK-ALLOCATE-(B-A)-Befehl ..... 62

4.2.6 Der BLOCK-FREE-(B-F)-Befehl ..... 63
4.2.7 Der MEMORY-READ-(M-R)-Befehl ..... 64
4.2.8 Der MEMORY-WRITE-(M-W)-Befehl ..... 65
4.2.9 Der MEMORY-EXECUTE-(M-E)-Befehl ..... 66
4.2.10 Der BLOCK-EXECUTE-(B-E)-Befehl ..... 67
4.3 Die USER-(U)-Befehle ..... 68
4.4 Ein Sonderling: der &-Befehl ..... 69

**5 Die Speicherorganisation der 1541 ..... 73**
5.1 Der RAM-Bereich der 1541 ..... 75
5.1.1 Die Zeropage ..... 77
5.1.2 Die Page 1 ..... 77
5.1.3 Die Seite 2 ..... 78
5.1.4 Die Pufferspeicher der 1541 ..... 78
5.2 Die beiden Schnittstellenbausteine der 1541 ..... 79
5.2.1 Der Buskontroller (BC) VIA 6522 ..... 79
5.2.2 Der Diskcontroller (DC) VIA 6522 ..... 81
5.3 Der ROM-Bereich der 1541 ..... 82
5.3.1 Die Aufgaben des DOS ..... 82
5.3.2 Die Arbeitsweise des 6502 Mikroprozessors ..... 83
5.3.3 Die Arbeit im Hauptprogramm ..... 85
5.3.4 Die Arbeit im Diskcontroller-Modus ..... 85
5.3.5 Die Technik der Jobschleife ..... 86

**6 Das Ausführen von Programmen *im* Pufferspeicher ..... 89**
6.1 Aufruf von Unterprogrammen aus dem ROM ..... 91
6.2 Nutzung des DOS Hauptprogramms ..... 91
6.3 Nutzung der Jobschleife ..... 93
6.3.1 Die Steuerung des Laufwerksmotors ..... 95
6.3.2 Die Steuerung des Steppermotors ..... 95
6.3.3 Lesen eines Blocks in den Pufferspeicher ..... 96
6.3.4 Schreiben eines Blocks auf Diskette ..... 97

**7 Die Aufzeichnung von Daten auf der Diskette ..... 99**
7.1 Aufbau eines Sektors auf der Diskette ..... 101
7.1.1 Aufbau des Blockheaders ..... 102
7.1.2 Aufbau eines Datenblocks ..... 103
7.2 Die Technik beim Schreiben auf Diskette ..... 104
7.2.1 Die SYNC-Markierungen ..... 104
7.2.2 Die GCR-Kodierung ..... 107
7.3 Das Schreiben von Bits auf Diskette ..... 111
7.3.1 Funktionsweise eines Schreib-/Lesekopfes ..... 111
7.3.2 Speichern von Daten auf Diskette ..... 114
7.3.3 Lesen der Daten von Magnetschichten ..... 115

7.3.4 Timing beim Schreiben und Lesen ..... 117

**8 Wiederherstellen zerstörter Disketten ..... 119**
8.1 Das Wiederherstellen gelöschter Files ..... 121
8.2 Retten einer Diskette nach der Formatierung ..... 122
8.2.1 Retten nach der kurzen Formatierung ..... 122
8.2.2 Retten nach der langen Formatierung ..... 124
8.3 Lesen von fehlerhaften Files ..... 124
8.3.1 Rettung bei Soft-Errors ..... 125
8.3.2 Rettung bei Hard-Errors ..... 126
8.4 Retten von physikalisch zerstörten Disketten ..... 127

**9 Funktionsweise von Softwareschutz auf Disketten ..... 129**
9.1 Schreiben von definierten Fehlern auf Diskette ..... 131
9.2 Verändern der Reihenfolge der Sektoren ..... 132
9.3 Verändern der Abstände zwischen den Sektoren ..... 133
9.4 Das Arbeiten mit illegalen Spuren ..... 133

**10 Der serielle Bus der 1541 ..... 135**
10.1 Die Arbeitsweise des seriellen Bus ..... 138
10.2 Spooling von Diskette ..... 140
10.3 Dem seriellen Bus Beine gemacht ..... 143

**11 Die Hardware der 1541 ..... 147**
11.1 Das Laufwerk der 1541 ..... 149
11.2 Eingriffe in die Platine bei der 1541 ..... 150
11.3 Tips zur Behandlung der Floppy ..... 153

**12 Fehler *im* DOS 2.6 der 1541 ..... 155**

**13 Die 1541 *im* Vergleich zu den anderen CBM-Floppies ..... 159**

**Anhang I: RAM-Belegung der 1541 ..... 163**

**Anhang II: DOS-Listing der 1541 ..... 175**

**Anhang III: Liste des gesamten Befehlssatzes mit Kurzbeschreibung ..... 379**

**Anhang IV: Liste der Fehlermeldungen des DOS 2.6 ..... 387**

**Anhang V: Programmlistings ..... 395**

**Stichwortverzeichnis ..... 423**

# 1
# Die Floppy 1541

# 1 Die Floppy 1541

Mit der Floppy 1541 und ihrer Vorgängerin, der 1540, hat Commodore als erste Computerfirma auch dem Hobbyanwender für seine Privatzwecke die schnelle Massenspeicherung ermöglicht. Es ist ja noch gar nicht lange her, daß dieses Privileg nur den Großanwendern vorbehalten war; Diskettenstationen waren für den Hobbybetrieb einfach zu teuer, und so mußte sich der Heimcomputerfan mit Kassettenrekordern begnügen.

Jetzt sieht die Sache allerdings anders aus, und viele von Ihnen, liebe Leser, werden sich Ihre Floppystation gar nicht mehr wegdenken können.

So werden wir denn damit beginnen, die Floppystation zu erforschen, und langsam aber sicher werden Sie erkennen, daß Sie es hier mit mehr als nur einem einfachen Massenspeicher zu tun haben. Den Freaks unter Ihnen sei empfohlen, die nun folgenden Kapitel für Einsteiger zu übergehen und sich zu den Abschnitten für Fortgeschrittene zu begeben (Kapitel 4 ff.).

## *1.1 DOS - Was ist das?*

Sicherlich werden Sie schon oft über die Bezeichnung 'DOS' oder 'Disk Operating System' gestolpert sein und sich gefragt haben, was dieser Ausdruck eigentlich zu bedeuten hat. Nun, was für Ihren Commodore 64 oder VC 20 das Betriebssystem und der BASIC-Interpreter ist, das ist für die Floppystation das DOS, was auf deutsch etwa 'Diskettenbetriebssystem' heißt. Wenn Sie nun Ihre Floppy zum Arbeiten einschalten, dann beginnt dort intern, genau wie im Computer, ein Programm abzulaufen, das im ROM gespeichert ist. Dieses Programm ist das DOS. Es sorgt dafür, daß Sie mit Ihrer 1541 überhaupt arbeiten können, indem es die Befehle entgegennimmt, auf Fehler durchsucht und gegebenenfalls ausführt....

Oho! Das hört sich doch genauso an, wie eine Beschreibung der Abläufe im Computer. Genau! Sie haben soeben eine Eigenschaft aller Commodore Floppies kennengelernt. Wir haben es hier nicht nur mit einfachen Laufwerken zu tun, sondern mit sogenannten Floppystationen oder 'intelligenten' Laufwerken. Diese Geräte sind mehr als nur einfache Massenspeicher; es sind vollständige Computer, mit eigenen Mikroprozessoren, Speichern und Betriebssystemen. Diese Bauart hat sowohl Vor- als auch Nachteile:

Vorteile:

- Ausführung von Befehlen, ohne die Rechenzeit des Computers zu beanspruchen.
- Es wird kein Speicher im Computer benötigt.
- Es können mehrere Computer auf eine Floppystation zugreifen.
- Befehle können unabhängig vom Computer ausgeführt werden.

Nachteile:

- Das DOS ist unveränderlich im ROM gespeichert.
- Direkte Manipulation auf Diskette sind schwierig.

## *1.2 Kontakt mit der 1541*

Die Unabhängigkeit der Floppystation bringt natürlich auch gewisse Probleme, außer den schon erwähnten Nachteilen, mit sich. So sind beim Dialog zwischen Computer und Diskettenstation einige bestimmte Regeln zu beachten. Da die 1541 nicht einfach als Speicherstelle behandelt werden kann, wie das beim Bildschirmspeicher der Fall ist, müssen wir einen anderen Weg gehen, um Zugang zur Diskette zu bekommen.

Die Floppystation ist über einen sogenannten seriellen Bus mit dem Computer verbunden. Diese Verbindung erlaubt den Anschluß vieler Peripheriegeräte an den Computer, wobei jedem Gerät eine Adresse, ähnlich einer Hausnummer, zugeordnet wird. Mit Hilfe dieser Adresse kann vom Anwender ein beliebiges Peripheriegerät angesprochen werden; die Commodore Floppies haben üblicherweise den Wert 8. Schickt der Computer eine 8 als Gerätenummer auf den

Bus, geht die Floppy in den Bereitschaftszustand und erwartet weitere Zeichen, zum Beispiel:

- LOAD, SAVE und VERIFY Kommandos.
- Floppy-Befehle, die über den Kommandokanal geschickt werden.
- Daten, die über Datenkanäle gesendet werden.

Die BASIC-Anweisungen in der ersten Zeile bedürfen keiner weiteren Erläuterung; die beiden folgenden Zeilen sollen uns jedoch im Verlauf dieses Buches noch intensiver beschäftigen, wobei wir mit dem Kommandokanal beginnen wollen.

## *1.3 Der Kommandokanal*

Die 1541 stellt uns mehrere sogenannte Kanäle zur Verfügung, mit denen wir mit ihr kommunizieren können und zwar Kanäle mit den Nummern von 0 bis 15. Die Kanäle 0 und 1 werden hierbei vom DOS bei LOAD und SAVE benötigt. Die Nummern 2 bis 14 stehen dem Anwender als Datenkanäle zur freien Verfügung und erlauben den Informationsaustausch zwischen Floppy und Computer. Der Kanal mit der Nummer 15 schließlich ist der Kommandokanal, auf dem Meldungen und Befehle übergeben werden.

Uns soll jetzt vorerst nur dieser Kommandokanal interessieren, da wir mit seiner Hilfe gleich einen Befehl an die Floppy schicken werden.

Schalten Sie also schon einmal ihren Computer und die Floppystation ein..... ach übrigens, bestimmt werden auch Sie schon vom Handbuch der 1541 verwirrt worden sein, da sich noch nicht einmal die deutsche und die englische Fassung darüber einig sind, welches der beiden Geräte, Floppy oder Computer, denn nun zuerst eingeschaltet werden soll. Was dieses Dilemma betrifft, so kann ich Sie beruhigen; Es ist völlig egal, welches der beiden Geräte Sie zuerst einschalten. Dies ist auch einleuchtend, nachdem Sie schon erfahren haben, daß sowohl die Floppystation, als auch die Zentraleinheit eigene Computer sind und somit keine 'Buspriorität' eingehalten werden muß.

Sinnvoller ist allerdings das Einschalten aller Peripheriegeräte - auch der Floppy - vor dem Computer, da dieser beim Starten ein RESET-Signal über den Bus schickt und so alle angeschlossenen Geräte in den Ausgangszustand versetzt.

Aber jetzt wieder zum eigentlichen Thema dieses Kapitels. Während Sie sich vielleicht noch überlegen, in welcher Reihenfolge Sie Ihr System starten, möchte ich schon einmal auf die Bedienung des Kommandokanals von BASIC aus eingehen.

Generell sieht der Ablauf folgendermaßen aus:

- Öffnen eines Files mit dem OPEN-Befehl.
- Senden von Kommandos mit PRINT#.
- Empfangen von Daten mit GET# oder INPUT#.
- Schließen des Files mit CLOSE.

Die Syntax des OPEN-Befehls lautet:

OPEN File#, Geräte#, Kanal#

Dabei bedeuten:

File# - logische Filenummer (1-127)
Geräte# - die Gerätenummer (in unserem Fall normalerweise 8)
Kanal# - die Sekundäradresse oder Kanalnummer (0-15)

Die Syntax des PRINT#-Befehls lautet:

PRINT# File#,Datal;Data2;....

Dabei bedeuten:

File# - logische Filenummer (1-127)
Data - sind die Zeichen, die zur Floppy geschickt werden sollen. Sie bestehen aus:
1) Zeichenketten in Anführungszeichen ("")
2) Zahlen oder Variablen
3) CHR$-Werten

Die Syntax von INPUT# und GET# lautet;

INPUT# File#, Variablel, Variable2, ....

GET# File#, Variablel, Variable2, ....

Dabei bedeutet:

File# - logische Filenummer (1-127)
Variable - eine Textvariable, die die eingelesene Zeichenkette aufnehmen soll.

Die Syntax von CLOSE ist:

CLOSE File#

Dabei bedeutet:

File# - logische Filenummer (1-127)

Damit genug der Theorie und nun zur Praxis:

OPEN 1,8,15

Mit diesem Befehl eröffnen wir ein File mit der Nummer 1, adressieren die Floppystation mit der Nummer 8 und reservieren uns dort den Kommandokanal für den weiteren Dialog (Sekundäradresse 15).

Da Ihnen der N-Befehl zum Formatieren einer neuen Diskette aus dem Handbuch bestens bekannt ist, möchte ich Sie bitten, eine leere Diskette zu nehmen und diese in das Laufwerk einzuführen. Anschließend senden wir genannten Befehl zur Floppy:

PRINT#1,"N:NEUE DISKETTE,ND"

Haben Sie die Zeile richtig eingetippt, hören Sie jetzt, wie das Laufwerk anläuft und die Diskette zu formatieren beginnt; der eigentliche Sinn des Formatierens wird in Kapitel 3 erörtert. (Im Anhang dieses Buches finden Sie eine Aufstellung aller im Commodore Handbuch besprochenen Befehle in einer kurzen Zusammenstellung; genauer werden wir später auf die Direktzugriffsbefehle eingehen).

Da wir ja glücklicherweise über ein "intelligentes" Laufwerk verfügen, meldet sich der Computer sofort nach der Befehlsübergabe wieder mit READY, und wir können dort weiterarbeiten, während die Floppystation noch mit der Ausführung beschäftigt ist.

Nun kann die Floppy Kommandos entgegennehmen und ausführen. Was aber viele nicht wissen, ist die Tatsache, daß die 1541 auch über einen sehr großen Vorrat an Rückmeldungen verfügt, die dem Anwender meistens verlorengehen, da das Commodore-BASIC ohne Erweiterungen nicht für deren Anzeige ausgelegt ist. Eine solche Meldung erfolgt nach jeder Befehlssausführung, und anhand dieser kann der Anwender sehr schnell feststellen, was für ein Fehler, zum Beispiel bei Blinken der Leuchtdiode, vorliegt.

Da diese Meldungen auch über den Kommandokanal zurückgeschickt werden, ist es für uns kein Problem, diese anzuzeigen, nachdem die Floppy ihre Arbeit beendet hat:

```
10 OPEN1,8,15
20 GET#1,A$:PRINT A$;:IF ST <>64 THEN 20
30 CLOSE1
```

Mit diesem kleinen Programm, das Sie übrigens in jedes größere BASIC-Programm als Unterprogramm einfügen können, wird der Kommandokanal der Floppystation ausgelesen und auf dem Bildschirm angezeigt. Die Zeile 20 könnte selbstverständlich auch lauten:

```
20 INPUT#1,A,B$,C,D:PRINT A;B$;C;D
```

Wurde das Formatieren der Diskette ordnungsgemäß abgeschlossen, meldet sich die Floppy mit

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

Andernfalls ist beim Formatieren ein Fehler aufgetreten, und es erscheint eine der im Anhang aufgeführten und ausführlich beschriebenen Fehlermeldungen. An unserem Programm wird Ihnen sicher aufgefallen sein, daß die GET#-Schleife zum Holen der Meldung die Statusvariable ST benutzt. Diese Variable nimmt immer den Wert 64 an, wenn die Floppy eine vollständige Dateneinheit gesendet hat. An der folgenden Tabelle können Sie die Belegung des Statuswertes ablesen, der in der Speicherstelle 144 ($90) gespeichert ist, oder den Sie über ST aufrufen können:

| Statusbit | Ursache, wenn Bit = 1 |
|---|---|
| 0 | Time-Out beim Schreiben |
| 1 | Time-Out beim Lesen |
| 2 | Block beim Laden zu kurz (Tape) |
| 3 | Block beim Laden zu lang (Tape) |
| 4 | Fehler beim Lesen (Tape) |
| 5 | Fehler in Prüfsumme (Tape) |
| 6 | EOI - keine weiteren Daten mehr |
| 7 | EOT - Ende der Übertragung |

Tabelle 2.1 Belegung der Statusbits

Wie Sie sehen, ist über den Kommandokanal ein vollständiger Dialog mit der Floppystation möglich. Wenn Sie mit dem auf der Commodore DEMO-Diskette vorhandenen "DOS 5.1" arbeiten, ist das Auslesen des Fehlerkanals sehr viel einfacher, und auch die übrige Bedienung der Floppy wird stark erleichtert.

# 2
# Datenspeicherung mit der 1541

# 2 Datenspeicherung mit der 1541

Dieses Kapitel ist der Datenspeicherung gewidmet, die nicht nur aus dem Abspeichern von Programmen besteht, sondern auch der Verwaltung großer Datenmengen, beispielsweise Kundenkarteien, dient und so dem Anwender ein weiteres großes Anwendungsgebiet seines Heimcomputers erschließt.

Die Datenspeicherung ist zwar auch mit einem Kassettenrekorder möglich; dieser kann es jedoch, was Geschwindigkeit und Komfort betrifft, nicht im geringsten mit einer Floppystation aufnehmen. In der Tat werden Sie in diesem Kapitel Möglichkeiten der Datenverwaltung kennenlernen, die Sie bestimmt kaum für möglich gehalten haben, zumal es Commodore wieder einmal prächtig verstand, die Fähigkeiten seines Produktes geheimzuhalten.

## *2.1 Das Inhaltsverzeichnis auf Diskette*

Bevor wir in die Datenspeicherung einsteigen, möchte ich einmal kurz auf das Inhaltsverzeichnis einer Diskette, das Directory, eingehen. Diese Eigenschaft der Anlage eines Directory fällt dem Einsteiger wohl als erstes angenehm auf, wenn er von der Datasette auf die Floppystation wechselt. Dank dieser Einrichtung hat man den Inhalt einer Diskette ständig zur Verfügung und kann auf jedes Programm individuell zugreifen, ohne erst andere Daten überlesen zu müssen.

Außer den Namen der einzelnen Files wird deren Länge, Typ und schließlich noch der verbleibende Platz auf der Diskette angegeben; so hat man immer den Überblick und erspart sich das lästige 'Spekulieren', wenn man auf Programmsuche ist.

Mit der Befehlsfolge

```
LOAD"$",8
```

wird das Directory einer Floppystation der Nummer 8 in den Speicher des Computers gelesen und kann so mit LIST angezeigt werden.

**ACHTUNG!!!** BASIC-Programme im Speicher werden überschrieben!

Da das Directory von der Floppy genauso wie ein herkömmliches File behandelt wird, kann der Anwender dessen Inhalt in einer Datei problemlos verwalten. Diese Eigenschaft ermöglicht das Auslesen des Directory vom Programm aus, ohne es zu zerstören. Ein Listing dazu finden Sie im Anhang dieses Buches.

## *2.2 Programm-(PRG)-Files* auf *der 1541*

Tippen Sie einmal folgende Zeile ein:

```
10 REM DIES IST EIN TEST
```

Danach speichern Sie dieses 'Programm' mit SAVE"TEST",8 auf Ihre neu formatierte Diskette und laden das Directory. Das sehen wir uns nun an; es sollte ungefähr so aussehen:

```
0 "NEUE DISKETTE " ND 2A
1  "TEST"             PRG
663 BLOCKS FREE.
```

Sie sehen den Namen des eben abgespeicherten Testprogramms und dahinter die Angabe des Filetyps 'PRG'.

Dieses Kürzel ist die Bezeichnung für mit SAVE abgespeicherte Programme und sagt Ihnen, daß Sie diesen Titel mit LOAD wiedereinlesen können.

Außer diesem Filetyp gibt es noch ein paar weitere, auf die wir gleich eingehen werden.

## 2.3 Die sequentielle (SEQ) Datenspeicherung

Diese Art der Datenspeicherung ist die Standardmethode für das Verwalten großer Datenmengen. Wollen Sie zum Beispiel eine Kartei führen, werden Sie selbst mit einem Commodore 64 schnell an die Grenzen des Hauptspeichers stoßen und sich nach einer externen Speichermöglichkeit umsehen müssen.

Zu diesem Zweck des Ablegens von Daten ist die sequentielle Datenspeicherung die einfachste und auch für den Anfänger schnell beherrschbare Methode, wenn auch nicht gerade die schnellste. Doch dazu später mehr.

Im Prinzip geht man hier den gleichen Weg wie bei der Programmspeicherung; die Daten werden der Reihe nach (sequentiell) auf Diskette geschrieben und können ebenso wieder in den Speicher des Computers eingelesen werden. Diese Methode der Speicherung kennen Sie vielleicht schon vom Arbeiten mit der Kassette her, da sie im Prinzip identisch abläuft; die Floppystation bietet aber auch hier wieder einige Vorteile gegenüber der Arbeit mit der Datasette:

1) Der Datenzugriff erfolgt um einiges schneller.

2) Späteres Anfügen von Daten ohne vorheriges Überlesen der gesamten Datei möglich.

Zur Verdeutlichung wollen wir auf unserer Testdiskette eine sequentielle Datei eroffnen; zuvor jedoch erst die allgemeine Syntax beim Eröffnen einer Datei auf Diskette:

```
OPEN File#,Geräte#,Kanal#,"Filename,Filetyp,Betriebsart"
```

Hierbei bedeuten:

```
File#       - die logische Filenummer (1-127)
Geräte#     - die Geräteadresse (hier normalerweise 8)
Kanal#      - die Kanalnummer (hier 2-14)
Filename    - Name des Files (maximal 16 Zeichen)
Filetype    - Es können sämtliche Filetypen geöffnet werden;
              hierbei ist:
                  P     das Kürzel für PRG-File
                  S     das Kürzel für SEQ-File
```

U das Kürzel für USR-File
L das Kürzel für REL-File

Betriebsart - Modus, in dem ein File geöffnet werden soll:
R File zum Lesen öffnen
W File zum Schreiben öffnen
A File für Append (Anhängen von Daten) öffnen

Da die Datenverwaltung mit dem Computer ein sehr komplexes Gebiet ist, welches allein schon ganze Bücher füllen würde, mochte ich mich an dieser Stelle auf das allernötigste, das zum Verständnis der Datenspeicherung notwendig ist, beschränken.

Eröffnen wir also jetzt eine Testdatei mit dem Namen "SEQ-DATEI":

```
OPEN1,8,2,"SEQ-DATEI,S,W"
```

Wie aus der Zeile hervorgeht, haben wir ein sequentielles File zum Schreiben geöffnet. Jetzt wollen wir etwas in diese Datei hineinschreiben:

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

Danach schließen wir die Datei mit

```
CLOSE1
```

Dieses Schließen einer geöffneten Datei vor dem Wechseln einer Diskette oder anderen Diskettenoperationen ist äußerst wichtig, da das offengebliebene File sonst als ungültig deklariert wird, und die Floppy bei dem Versuch, dieses File später einmal zu lesen, eine Fehlermeldung ausgibt. Auch ein nachträgliches Schließen dieses Files ist nicht möglich. Im Directory erkennt man ein nicht geschlossenes File an einem Sternchen (*) vor dem Filetyp.

Sollte Ihnen dennoch einmal dieser Fehler unterlaufen, so verzweifeln Sie nicht. Solange Sie nicht den VALIDATE-Befehl anwenden, wird Ihrem File nichts passieren. Im Anhang finden Sie ein Programm, das Ihnen das File mit einem Trick ordnungsgemäß schließt.

Haben wir unsere Testdatei ordnungsgemäß geschlossen, sehen wir uns das Inhaltsverzeichnis unserer Diskette an und entdecken, daß dort unser Testfile mit einem 'SEQ' als Filetyp hinzugekommen ist.

Das Einlesen des abgespeicherten Inhalts müssen wir mit Hilfe eines kleinen Programms vollziehen, da INPUT und GET im Direktmodus nicht gestattet sind;

```
10 OPEN1,8,2,"SEQ-DATEI,S,R"
20 INPUT#1,A$
30 PRINTA$
```

Nach einem RUN erscheint ein uns bekannter Text;

```
DIES IST EIN SEQUENTIELLES FILE
```

Jetzt hat es nicht so fatale Folgen, wenn wir den CLOSE-Befehl vergessen; die Floppy erinnert uns aber mit einer dauernd brennenden Leuchtdiode an unser Vergehen, und so wollen wir das Versäumte schnell nachholen:

```
CLOSE1
```

Sie haben jetzt die eigentliche Funktion der Leuchtdiode am Laufwerk der 1541 kennengelernt: sie brennt grundsätzlich, wenn ein Zugriff auf ein File erfolgt; ein LOAD oder SAVE ist ja letztendlich auch nichts anderes als das Eröffnen eines PRG-Files und dann das Schreiben beziehungsweise Lesen von Daten.

Nun aber noch zur dritten Filebetriebsart, dem Append. Hier ist es uns erlaubt, ein einmal zum Schreiben geöffnetes und wieder geschlossenes File abermals zu öffnen, um weitere Daten hinzuzufügen:

```
OPEN1,8,2,"SEQ-DATEI,S,A"
```

Wieder geht die Leuchtdiode am Laufwerk an und zeigt uns an, daß der Zugriff stattfinden kann:

```
PRINT#1,"DIES IST EINE VERLAENGERUNG"
CLOSE1
```

Würden Sie jetzt erneut die Daten auslösen, was Sie nun schon können müßten, wäre es allerdings erforderlich, den INPUT#-Befehl zweimal anzuwenden, da wir ja auch zweimal mit PRINT# etwas in die Datei hineingeschrieben haben.

Zum Abspeichern von Daten mit PRINT# einmal ein paar grundsätzliche Bemerkungen:

Wie beim PRINT-Befehl auf dem Bildschirm, ist auch beim PRINT#-Befehl zur Peripherie zu beachten, daß nach jedem abgeschlossenen Befehl automatisch ein CHR$(13) ('RETURN') mit übergeben wird. Dies kann man vermeiden, indem man jedesmal nach dem zu übergebenden Text ein Semikolon ';' schreibt. Es wird dann der Zeilenvorschub genau wie auf dem Bildschirm unterdrückt:

```
PRINT#1,"BEISPIEL MIT SEMIKOLON";
```

Liest man die Daten mit INPUT#, erfolgt der Lesevorgang immer bis zum nächsten 'RETURN'. Dies ist wichtig zu wissen, wenn beim Abspeichern von Daten in einem File das 'RETURN' durch die oben genannte Methode unterdrückt wurde. Ist der Text nämlich länger als 255 Zeichen, muß mit GET# eingelesen werden, da sonst ein '?STRING TOO LONG ERROR' die Folge ist.

Dies sollte genügen, um Sie mit der sequentiellen Datenspeicherung vertraut zu machen. Wie Sie sicherlich schon bemerkt haben, können auch PRG-Files zum direkten Auslesen durch INPUT# oder GET# auf oben genannte Art geöffnet werden; diese Möglichkeit ist die Grundlage für Filekopierprogramme.

Weitere Anwendungen der Datenspeicherung und Anregungen zur eigenen Programmiertätigkeit finden Sie im Anhang.

## *2.4 Datenspeicherung in USR-Files*

Die USER-(USR)-Files stellen eine Abart der sequentiellen Files dar, sind in Aufbau und Bedienung jedoch identisch (von ein paar zusätzlichen Aufgaben einmal abgesehen). Sie bilden zum Beispiel die Grundlage für das sogenannte 'Spooling'. Unter Spooling versteht man beispielsweise die Ausgabe von Daten auf einem Drucker, ohne daß der Computer blockiert wird. Dieser Trick funktioniert mit einer Floppy auf die Art, daß die entsprechenden Daten, zum Beispiel ein Listing, auf Diskette abgespeichert werden, und dann ein Programm nur zwischen Floppy und Drucker abläuft, welches die

Drucksteuerung übernimmt. (Wie das funktioniert, erfahren Sie in den Kapiteln für Fortgeschrittene).

Glücklicherweise versetzt uns der CMD-Befehl in die Lage, alle Bildschirmausgaben auf den Drucker oder auf die Floppy umzuleiten, so daß das Ablegen eines Programmlistings in einem File zu einem Kinderspiel wird:

```
10 OPEN1,8,2,"LISTING,U,W"
20 CMD1
30 LIST

CLOSE1
```

Dieses Programm listet sich selbst in ein USR-File namens 'LISTING', und Ihnen bleibt nur noch, dieses File anschließend von Hand zu schließen.

## *2.5 Datenspeicherung mit relativen (REL) Files*

Dies ist die letzte Möglichkeit der Datenspeicherung und zugleich auch die schnellste und komplizierteste. Mit ihr sollte sich nur beschäftigen, wer schon Erfahrung in der Arbeit mit der Floppy hat und einen möglichst schnellen Datenzugriff benötigt. Die relative Datenspeicherung wird in den Handbüchern der 1541 nicht einmal erwähnt, obwohl sie gegenüber der sequentiellen Datenspeicherung enorme Vorteile besitzt. Das einzige Problem, mit dem der Anwender hier zu kämpfen hat, ist die umständliche Handhabung einer relativen Datei, da sie eigentlich für das BASIC 4.0 der großen Commodore Computer ausgelegt ist. Dennoch meine ich, daß es sich lohnt, diese komfortable Methode der Datenverarbeitung kennenzulernen.

Bei der sequentiellen Datenverwaltung hatten wir immer das Problem, daß wir bei einer sehr langen Datei alle Daten durchlesen mußten, um an eine bestimmte Stelle zu gelangen. Die relative Datenspeicherung geht hier einen anderen Weg:

Es wird davon ausgegangen, daß jeder Datensatz, der bestimmte Informationen enthält, nur eine bestimmte Länge hat. Jeder dieser Datensätze wird jetzt durchnumeriert, das heißt, Sie können durch Angabe der jeweiligen Nummer auf jeden beliebigen Datensatz direkt zugreifen, was die Geschwindigkeit um einiges erhöht.

Das Anlegen einer Datei geht jetzt allerdings etwas anders vor sich:

- Öffnen der Datei nach bekanntem Schema. Dabei wird die Datensatzlänge angegeben.
- Der allerletzte Datensatz wird markiert.
- Die Datei wird ordnungsgemäß geschlossen.

Um dieses Prinzip zu verdeutlichen, legen wir auf unserer Testdiskette eine relative Datei an:

```
OPEN1,8,2,"REL-FILE,L,"+CHR$(100)
```

Wie Sie sehen, ist jetzt anstelle der Betriebsart des Files die Datensatzlänge in Form eines CHR$-Codes angegeben. In unserem Fall beträgt diese Länge 100 Zeichen, wobei darauf zu achten ist, daß diese 100 Zeichen inklusive des 'RETURN', das am Ende jedes PRINT# gesendet wird, zu zählen sind. Die tatsächliche Datensatzlänge reduziert sich deshalb auf 99 Zeichen.

Als nächstes wollen wir den letzten Datensatz markieren. Dies erfolgt durch Freigeben desselben. Mit freigeben ist hier gemeint, daß der Datensatz mit CHR$(255)-Bytes vollgeschrieben wird, da er erst dann zum Zugriff zugelassen ist. Das Freigeben eines Datensatzes hat zur Folge, daß auch alle Datensätze mit niedrigeren Nummern bis zum letzten beschriebenen freigegeben werden, was seine Zeit benötigt. Um diese Angelegenheit ein für allemal zu erledigen, geben wir den allerletzten Datensatz frei.

Für diesen Zweck verwenden wir erstmals den Positionier-(P)-Befehl; dabei sollten Sie nicht vergessen, daß es sich um einen Befehl handelt, der über den Kommandokanal geschickt werden muß. Die Syntax lautet:

```
PRINT# File#,"P"CHR$(Kanal#)CHR$(NrLo)CHR$(NrHi)CHR$(Stelle)
```

Hierbei bedeuten:

| | |
|---|---|
| File# | – die logische Filenummer (1–127) |
| Kanal# | – die Kanalnummer (2–14) |
| NrLo | – das niederwertige Byte der Anzahl der Datensätze |
| NrHi | – das höherwertige Byte der Anzahl der Datensätze |
| Stelle | – die Nummer des Zeichens in einem Datensatz, auf das positioniert werden soll. |

Die beiden Werte NrLo und NrHi sind die zwingende Aufspaltung der Nummer des Datensatzes, da in einem CHR$-Statement nur Werte bis maximal 255 vorkommen dürfen, wir aber durchaus 1000 Datensätze haben können. Mit den folgenden Formeln lassen sich die Werte aus der Gesamtanzahl berechnen:

```
NrHi = INT(Gesamtanzahl/256)
NrLo = Gesamtanzahl-NrHi*256
```

Als Beispiel mag die folgende Zeile dienen:

```
OPEN2,8,15
PRINT#2,"P"CHR$(2)CHR$(34)CHR$(3)CHR$(23)
CLOSE2
```

Wir positionieren auf einen Datensatz der Nummer 3*256+34 = 802 und zwar auf das 23. Zeichen.

Aber jetzt zu unserer Datei. Wir wollen die gesamten Datensätze freigeben, sagen wir 200 Stück:

```
OPEN2,8,15
PRINT#2,"P"CHR$(2)CHR$(200)CHR$(0)CHR$(1)
PRINT#1,CHR$(255)
CLOSEl:CLOSE2
```

Das kann eine Weile dauern. Ein Grund, warum es jedesmal beim Neueröffnen einer Datei sofort erfolgen sollte.

Das Blinken der Leuchtdiode am Laufwerk der 1541 sollte Sie übrigens nicht irritieren. Es tritt dann auf, wenn Sie nur den P-Befehl, aber noch nicht das CHR$(255)-Byte senden. Der Grund liegt in der Tatsache, daß Sie auf einen Datensatz positionieren wollen, der noch nicht angelegt ist. Das Anlegen erfolgt anschließend sofort durch das Senden des Bytes. Verwunderlich ist außerdem, daß die LED während des Freigebens nicht brennt; aber auch das ist normal und sollte Sie nicht stören.

Nach dem Schließen der Datei sollten Sie jetzt ein bißchen mit Lesen und Schreiben experimentieren, damit Sie sich an die Anwendung gewöhnen und sich die Beispielprogramme im Anhang zu Gemüte fuhren können. Vergessen Sie aber nicht, daß Sie vor einem Schreibzugriff jedesmal erst mit dem P-Befehl an die Stelle gehen müssen, die beschrieben werden soll. Das gleiche gilt für einen Lesezugriff.

Ein Schreibbeispiel:

```
OPEN 1,8,2,"REL-DATEI,L"CHR$(100)
OPEN 2,8,15
PRINT#2,"P"CHR$(2)CHR$(30)CHR$(0)CHR$(1)
PRINT#1,"DIES IST EIN SCHREIBBEISPIEL";
```

Das Schließen der Datei erfolgt analog zur sequentiellen Datei. Vergessen Sie dabei auch den Kommandokanal nicht.

Genauso, wie wir in einen Datensatz hineinschreiben, können wir auch wieder herauslesen:

```
10 OPEN1,8,2,"REL-DATEI,L"CHR$(100)
20 OPEN2,8,15
30 PRINT#2,"P"CHR$(2)CHR$(30)CHR$(0)CHR$(1)
40 A$=""
50 FORX=1T0100
60 GET#1,X$:IFX$=CHR$(255)THEN PRINTA$;GOTO 80
70 A$=A$+X$:NEXT X
80 CLOSE1:CLOSE2
```

In diesem Programm benutzen wir zum Lesen eine GET#-Schleife, die mit dem Vorgang aufhört, sobald sie auf das 'Leerkennzeichen' CHR$(255) stoßt.

Wegen der Große des Pufferspeichers kann bei der 1541 immer nur eine relative Datei auf einmal geöffnet sein. Es kann allerdings noch zusätzlich ein sequentielles File geöffnet werden. Arbeiten Sie nur mit sequentiellen Files, können Sie 3 Dateien gleichzeitig offen halten.

# 3
# Das Diskettenformat der 1541

# 3 Das Diskettenformat der 1541

Sicherlich haben Sie sich schon einmal gefragt, was das soll: Formatieren. Was passiert in der Floppy, wenn sie diesen über eine Minute dauernden Vorgang beginnt, wobei sie jede Spur der Diskette abfährt? Warum 'rattert' die Floppy während des Formatierens? Nun, wir werden in diesem Abschnitt zwar nicht auf die internen Vorgänge der Floppy eingehen - dies soll später geschehen - aber wir wollen untersuchen, wie eine Diskette nach der Formatierung aussieht und was das Ganze für einen Sinn hat.

## *3.1 Der Aufbau einer neu formatierten Diskette*

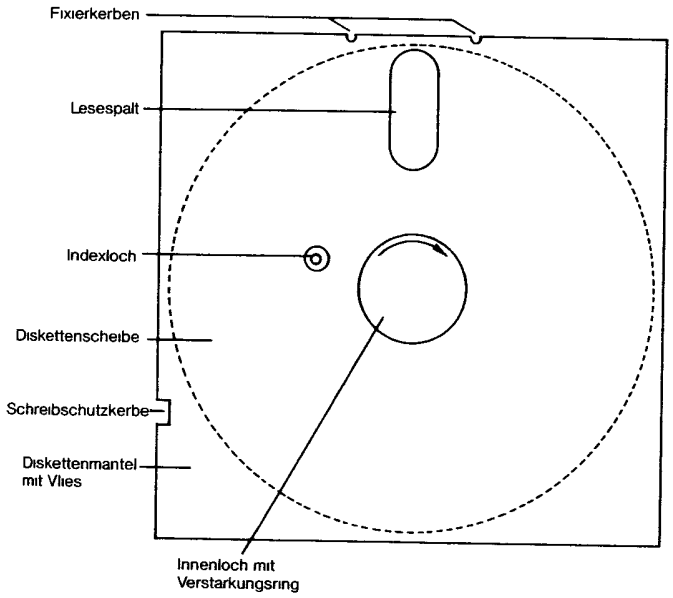



Bild 3.1 Darstellung einer Diskette im Schema

Es gibt insgesamt zwei Arten der Datenspeicherung auf Diskette. Die sogenannte hardsektorierte und die softsektorierte Datenspeicherung.

Sicherlich ist Ihnen schon einmal das sogenannte Indexloch, eine kleine Öffnung in der Magnetscheibe neben dem Loch in der Mitte der Diskette, aufgefallen. Dies ist bei den von Ihnen normalerweise verwendeten Disketten nur ein Loch. Es gibt jedoch auch Disketten mit vielen dieser Indexlocher. Es handelt sich hierbei um hardsektorierte Disketten.

Hardsektoriert arbeitende Laufwerke bestimmen anhand dieser Löcher die Position der Magnetscheibe, damit Daten gefunden oder zusammenhängend geschrieben werden können.

Softsektoriert arbeitende Laufwerke verwenden nur das eine Loch zur Positionsbestimmung und schreiben sich die übrigen erforderlichen Markierungen direkt auf Diskette. Den Vorgang des Markensetzens nennt man Formatieren.

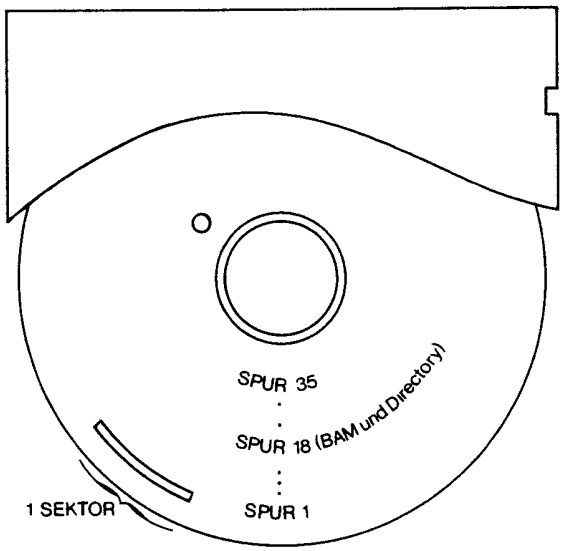



Bild 3.2 Belegung einer formatierten Diskette

### 3.1.1 Aufbau einer Spur

Wie Sie aus Bild 3.2 erkennen können, ist eine auf der 1541 formatierte Diskette in 35 Spuren (meistens wird der englische Ausdruck 'Track' verwendet) aufgeteilt, wobei Track 1 ganz außen und Track 35 ganz innen auf der Diskette liegt.

Jeder Track einer Diskette ist wiederum in kleinere Einheiten, sogenannte Sektoren, untergliedert. Diese Sektoren sind von 0 beginnend bis zur maximalen Anzahl durchnumeriert.

Die Zählweise von Null an sollten Sie sich angewöhnen, da sie bei der Arbeit mit dem Computer üblich ist und sehr oft verwendet wird.

Die maximale Anzahl der Sektoren pro Track ist verständlicherweise unterschiedlich, da zum Beispiel Track 1 um einigs länger ist als Track 23. Tabelle 3.1 zeigt diese Aufteilung:

| Tracknummer | Anzahl der Sektoren |
|---|---|
| 01 - 17 | 21 (00-20) |
| 18 - 24 | 19 (00-18) |
| 25 - 30 | 18 (00-17) |
| 31 - 35 | 17 (00-16) |

Tabelle 3.1 Anzahl der Sektoren für jeden Track

Track 1 enthält beispielsweise 21 Sektoren, die entgegen dem Uhrzeigersinn angeordnet sind.

### 3.1.2 Der Aufbau eines Sektors

Sektoren stellen im Gesamtaufbau einer Diskette die kleinste Einheit dar und enthalten jeweils einen Block Daten. Das sind 256 Bytes. Durch die Numerierung auf einer Diskette kann man jeden Block gezielt anwählen, indem man einfach die Spur- und Sektornummer angibt. Zu diesem Zweck besteht jeder Sektor auf der Diskette aus einem Vorspann und dem dazugehörigen Datenblock. Der Vorspann enthält die Nummer des Sektors und der Spur. Der Datenblock besteht aus den schon angesprochenen 256 Bytes an Information.

Das Anwählen und die Verwaltung der Sektoren geschieht im Normalfall vom DOS automatisch, und wir merken gar nichts davon. Die einzige Information, die uns das DOS im Normalmodus mitteilt, steht im Directory und bezeichnet die Anzahl der Blöcke, aus denen ein File besteht.

### 3.1.3 Verkettung von Sektoren

Wie Sie schon wissen, kann ein File aus sehr vielen Blöcken bestehen, und jeder Block ist in einem anderen Sektor untergebracht. Wie erkennt das DOS, welche der insgesamt 683 Blöcke einer Diskette zu jenem bestimmten File gehören, das wir gerade laden wollen?

Die Lösung dieses Problems ist einfach. In Wirklichkeit können wir nämlich in einem Datenblock, obwohl dieser aus 256 Bytes besteht, nur 254 Bytes an Daten abspeichern, da die ersten zwei Bytes eines jeden Datenblocks als sogenannte Linker benutzt werden. Diese Linker dienen der Verkettung der Sektoren und enthalten in Byte 0 die Spur- und in Byte 1 die Sektornummer des nächsten Blocks dieser Datei.

Wenn Sie ein Programm mit 3 Blöcken Länge laden wollen, lädt das DOS den ersten Block und schickt ihn zum Computer; danach "sieht" es anhand des Linkers, wo der zweite Block steht und liest diesen

ebenfalls. Ist der letzte Block gelesen, muß das DOS erkennen, daß es nun aufhören soll. Zu diesem Zweck steht im letzten Datenblock jeweils 0 als Spurnummer. Da diese nicht existiert, bricht das DOS den Vorgang ab; Byte 1 enthält dann die Anzahl der benutzten Bytes im Block.

Die eben beschriebene Verkettung von Blöcken ist bei allen Anwendungen üblich. Sie sollten sich diesen Abschnitt deshalb gut einprägen.

## *3.2 Aufbau der BAM*

So, jetzt wird's interessant. Wir dringen nämlich schon ziemlich tief in das Innenleben der Floppystation ein und wollen uns ansehen, wie das DOS erkennt, welchen Namen eine Diskette hat und ob diese gewechselt wurde, oder woher die Floppy weiß, daß ein Block schon beschrieben ist.

Die Überschrift dieses Kapitels werden Sie vielleicht mit einem leichten Lächeln bedacht haben: BAM? Was soll denn das sein? Nun, BAM ist eine Abkürzung für den englischen Ausdruck 'Block Availability Map', was übersetzt etwa 'Blockbelegungsplan' heißt.

Wie Sie vielleicht schon wissen, ist der Track 18 einer jeden Diskette vom DOS reserviert und steht dem Anwender nicht zur freien Verfügung. Auf dieser Spur werden alle wichtigen Werte (unter anderem auch das Inhaltsverzeichnis einer Diskette) abgespeichert. Sie werden sich vielleicht sogar schon Gedanken über einen eventuellen Druckfehler in diesem Buch gemacht haben, als ich in Kapitel 3.1.3 von 683 Blöcken gesprochen habe, obwohl man bei einer leeren Diskette im Directory nur '664 blocks free" lesen kann. Diese Differenz kommt durch Track 18 zustande, der 19 Sektoren enthält.

Nun aber zur BAM. Dieser Blockbelegungsplan steht in Block 18,0 (Track 18 und Sektor 0) zusammen mit noch einigen anderen Werten.

Tabelle 3.2 zeigt den Inhalt dieses Blocks:

| Bytes von bis | Bedeutung der entsprechenden Bytes |
|---|---|
| 000 - 001 | Linker zum nächsten Block im Directory; steht immer auf 18/01 |
| 002 - 002 | Formatkennzeichen $41 ('A') |
| 003 - 003 | 00; keine Funktion |
| 004 - 143 | Bitmuster der BAM; für jeden Track sind dabei 4 Bytes reserviert |
| 144 - 161 | Diskettenname; Leerstellen sind mit $A0 (160) aufgefüllt |
| 162 - 163 | Diskettenidentifikation (ID) |
| 164 - 164 | $AO (160) 'SHIFT SPACE' |
| 165 - 166 | Formatkennzeichen für Anzeige im Header des Directory; enthält bei der 1541 die Bytes $32 und $41; ASCII-Zeichen '2A' |
| 167 - 170 | $A0 (160) 'SHIFT SPACE' |
| 171 - 255 | normalerweise unbenutzt; kann bei manchen Disketten den Inhalt 'BLOCKS FREE.' Haben |

Tabelle 3.2 Inhalt des Blocks 18,0 einer Diskette

### 3.2.1 Das Formatkennzeichen

Bevor wir in Block 18,0 intensiver einsteigen, möchte ich Sie auf ein Programm im Anhang aufmerksam machen. Es handelt sich um einen Diskettenmonitor/Editor, mit dem Sie das im folgenden Gesagte leicht selbst überprüfen können.

Die Tabelle 3.2 enthält eine Aufstellung des Inhalts von Block 18,0. Sie können erkennen, daß auch hier die ersten beiden Bytes als Linker dienen; sie enthalten üblicherweise 18,1, was einen Zeiger auf den ersten Block des Directory darstellt. Darauf kommen wir im Anschluß zu sprechen.

Byte 2 enthält das Formatkennzeichen der Diskette. Es handelt sich hier um den Wert 65 ($41), der ein 'A' darstellt. Dieses A bekommen Sie zu sehen, wenn Sie sich das Directory einer Diskette anzeigen lassen. Es steht dann hinter dem Diskettennamen '2A'. Was ist nun die Funktion dieses Wertes?

Commodore stellt eine ganze Reihe von Floppystationen her. Angefangen bei der 1541 bis hin zur CBM 8250. Jede dieser Stationen hat ihre eigene Formatierungsroutine und da fast alle Floppys andere Aufzeichnungskapazitäten haben, unterscheiden sich auch deren Formatierungsroutinen und letztendlich deren formatierte Disketten mehr oder weniger voneinander. Die Folge ist, daß zum Beispiel eine 4040 keine Disketten der 8250 Stationen lesen und diese schon gar nicht beschreiben kann.

Damit keine Pannen passieren, ordnet man jeder Floppystation ein eigenes Formatkennzeichen zu. Bei der 1541 ist es ein "A", bei der 8050 ein 'C'. Werden nun Disketten vertauscht, bricht die Floppy ihre Arbeit mit einer Fehlermeldung ab.

### 3.2.2 Die Block Availability Map

Byte 3 des Blocks 18,0 enthält ein Nullbyte und hat bei der 1541 keine Funktion.

Jetzt kommt erst die eigentliche BAM mit ihrem Verzeichnis über alle freien und belegten Blöcke der Diskette. Hierbei sind für jede Spur 4 Bytes reserviert. Das jeweils erste Byte gibt die Zahl der verfügbaren Blöcke dieser Spur an, und die restlichen 3 Bytes bilden eine Gesamtheit von 24 Bits, wobei jedes '1'-Bit einen freien und jedes '0'-Bit einen belegten Block repräsentiert. Eine Aufstellung zeigt Tabelle 3.3.

| Byte | Bitnummer | Funktion der einzelnen Bits |
|---|---|---|
| 0 | 0-7 | Anzahl der freien Sektoren dieses Tracks |
| 1<br>2<br>3 | 0<br>1<br>2<br>3<br>4<br>5<br>6<br>7 | Jeweils ein Bit dieser 3 Bytes zeigt den Zustand eines Sektors dieser Spur an. Dabei bedeutet:<br><br>Bit=1: Sektor frei<br>Bit=0; Sektor belegt |

Tabelle 3.3 Eintrag eines Tracks in der BAM

Die gesamte BAM belegt die Bytes von 4 bis 143, so daß wir mit Byte 144 fortfahren.

### 3.2.3 Der Name einer Diskette

Die Bytes 144 bis 161 enthalten den Namen einer Diskette, der beim Formatieren vom Benutzer festgelegt wird. Die Länge dieses Namens ist maximal 16 Zeichen; bei kürzeren Namen werden die restlichen Bytes mit 'SHIFT SPACE' (entspricht dem Wert 160 oder $A0) aufgefüllt.

### 3.2.4 Die ID einer Diskette

Die Bytes 162 und 163 enthalten die ebenfalls bei der Formatierung festgelegte Diskettenidentifikation oder kurz ID genannt. Das DOS kann mit ihrer Hilfe erkennen, ob eine Diskette gewechselt wurde, da dann automatisch neu initialisiert werden muß. Aus diesem Grund ist es unbedingt nötig, daß jede Diskette eine andere ID erhält. Wurde das Initialisieren nämlich unterbleiben, wäre die BAM im Speicher der Floppy nicht mehr aktuell, und eine Katastrophe beim nächsten Schreibvorgang wäre die Folge.

Byte 164 enthält wieder den Wert 160 ($A0) und die Bytes 165 und 166 sind mit den ASCII-Zeichen '2A' belegt, wobei das 'A' eine Kopie von Byte 2 des Blocks darstellt. Das Zeichen "2" steht für die DOS-Version mit der gearbeitet wird (CBM DOS V2.6).

In den Bytes 167 bis 170 treffen wir wiederum auf den Wert 160 ($AO).

Die restlichen Bytes des Blocks (171 bis 255 oder $AB bis $FF) sind normalerweise nicht von Bedeutung und können unterschiedlichen Inhalt aufweisen. Manchmal steht dort die Meldung 'BLOCKS FREE'.

## *3.3 Aufbau des Directory*

Im vorigen Abschnitt erfuhren Sie etwas über die Daten, die sich das DOS sichert, um eine Fehlbedienung der beschriebenen Disketten auszuschließen. Diese etwas graue Theorie, die uns in Kapitel 3 begleitet, ist leider zwingend notwendig, um die Vorgänge in der Floppy voll zu verstehen und dann eventuell selber in diese Vorgänge einzugreifen, um Manipulationen an der 1541 vorzunehmen.

Jetzt soll uns zum Beispiel das eigentliche Inhaltsverzeichnis der Diskette interessieren. Woher weiß das DOS, an welcher Stelle der Diskette welches Programm mit welchem Namen steht? Alles Fragen, die Sie nach dem Durcharbeiten dieses Abschnittes leicht beantworten können.

| Bytes von bis | Belegung der Bytes im Directory |
|---|---|
| 000 - 001 | Zeiger auf Track und Sektor des nächsten Directoryblocks |
| 002 - 031 | 1. Fileeintrag |
| 034 - 063 | 2. Fileeintrag |
| 066 - 095 | 3. Fileeintrag |
| 098 - 127 | 4. Fileeintrag |
| 130 - 159 | 5. Fileeintrag |
| 162 - 191 | 6. Fileeintrag |
| 194 - 223 | 7. Fileeintrag |
| 226 - 255 | 8. Fileeintrag |

Tabelle 3.4 Inhalt eines Blocks des Directory

In Tabelle 3.4 sehen Sie den Aufbau von Block 18,1, auf den wir schon durch den Linker in 18,0 hingewiesen wurden. Hier steht der erste Block des eigentlichen Directory. Wie üblich bilden die beiden ersten Bytes einen Zeiger. Steht Byte 0 auf Null, ist der betrachtete Block der letzte dieses Directory.

### 3.3.1 Fileeinträge im Directory

Nach dem Linker folgt auch schon der erste Eintrag eines Files. Jeder dieser Einträge besteht aus 30 Bytes und enthält alle wichtigen Informationen über das jeweilige Programm. Tabelle 3.5 zeigt den Aufbau eines solchen Eintrags:

| Bytes von bis | Bedeutung der Bytes im Eintrag |
|---|---|
| 000 - 000 | Filetyp (siehe Tabelle 3. 6.) |
| 001 - 002 | Track- und Sektornummer des ersten Blocks im File |
| 003 - 018 | Filename; Leerstellen sind mit $A0 (160) aufgefüllt |
| 019 - 020 | Track und Sektor des ersten Side-Sektor-Blocks (nur REL Dateien) |
| 021 - 021 | Recordlänge (nur REL Dateien) |
| 022 - 025 | Zwischenspeicher bei DOS-Operationen |
| 026 - 027 | Zwischenspeicher für Track und Sektor des neuen Files beim Überschreiben mit REPLACE '@' |
| 028 - 029 | Länge des Files in Blöcken auf Diskette (L/H) |

Tabelle 3.5 Aufbau eines Fileeintrags im Directory

Das erste Byte eines jeden Eintrages gibt demnach den Filetyp an, der hier vorliegt. Tabelle 3.6 zeigt diese Kennzeichnung:

| Bitnummer | Funktion des Bits |
|---|---|
| 0 | Kennzeichnung des Filetyps |
| 1 | 0000 - DEL 0011 - USR |
| 2 | 0001 - SEQ 0100 - REL |
| 3 | 0010 - PRG |
| 4 | keine Funktion |
| 5 | keine Funktion |
| 6 | Bit=1 zeigt SCRATCH-Schutz an |
| 7 | Bit=0 zeigt offenes File an |

Tabelle 3.6 Belegung der Bits im Filetyp

Wie man sieht, kann der Filetyp durch die Bits 0 bis 2 beschrieben werden, so daß noch Bits für andere Informationen freibleiben. Die Bits 3 bis 5 sind unbenutzt. Bei Bit 6 entdecken Sie eine Funktion, die Ihnen bisher wahrscheinlich unbekannt war: hier kann man ein File vor dem Löschen mit SCRATCH schützen, indem man dieses Bit auf 1 setzt. Im Directory steht daraufhin ein '<'-Zeichen hinter dem Filetyp zur Kennzeichnung der geschützten Datei. Im Anhang finden Sie ein Programm, das Ihnen das Aufbringen des SCRATCH-Schutzes ermöglicht. Auch ein Löschen dieser Funktion, also ein Wiederfreigeben des Files ist mit dem Programm realisierbar.

Aha, werden Sie sagen, wenn sie sich die Funktion von Bit 7 des Bytes ansehen. Es kennzeichnet, ob ein File ordnungsgemäß geschlossen wurde und steht dann auf 1. Können sie sich noch an Kapitel 2 erinnern? Dann werden Sie wissen, daß im Anhang ein Programm existiert, welches ein versehentlich offen gelassenes File nachträglich schließt. Dieses Programm setzt einfach besagtes Bit auf 1. Wie so etwas funktioniert, erfahren Sie in Kapitel 4.

Jetzt aber wieder zurück zu unseren Einträgen im Directory. Wir wollen uns die Belegung der übrigen Bytes einmal genauer ansehen.

Die Bytes 1 und 2 des Eintrags im Directory geben den Track und den Sektor an, in dem der erste Block des Files abgelegt ist.

Die Bytes 3 bis 18 enthalten den Namen des Files, wie schon im Namen der Diskette, ergänzt mit 'SHIFT SPACE' = 160 ($A0).

Die Bytes 19 und 20 des Eintrags werden nur bei relativen Dateien benutzt und enthalten dann Track- und Sektornummer des ersten Side-Sektor-Blocks. Was das ist, erfahren Sie in Kapitel 3.7.

Byte 21 hat ebenfalls nur bei relativen Dateien eine Funktion und enthält dann die Länge der Records (Datensätze; siehe auch Kapitel 2.5).

Da die Bytes 22 bis 25 keine Funktion erfüllen, kommen wir gleich zu den Bytes mit den Nummern 26 und 27. Diese enthalten Track und Sektor des neuen Files, falls das vorherige mit REPLACE ('@:') überschrieben wurde (nur Zwischenspeicherfunktion).

Schließlich noch die Bytes 28 und 29; sie geben die Anzahl der Blöcke in einem File an. Dabei bestimmt Byte 28 den niederwertigen Teil dieser Zahl.

Wie aus der Tabelle 3.4 zu erkennen ist, enthält ein Block auf der Diskette 8 Fileeinträge. Insgesamt sind bis zu 144 Einträge pro Diskette möglich.

Nachdem wir uns nun durch soviel trockene Theorie durchgearbeitet haben, müssen wir feststellen, daß uns das alles im Augenblick relativ wenig angeht, da uns das DOS hier normalerweise alle Arbeit abnimmt. Also wollen wir uns nun einem Gebiet zuwenden, welches uns schon mehr berührt. Es handelt sich um den Aufbau von Files.

Wie sieht so ein sequentielles File, in dem wir wie selbstverständlich Daten abspeichern, überhaupt aus? Was unterscheidet PRG-Files und ÜSR-Files?

Zu diesem Zweck sehen wir uns jeden Filetyp genau an.

## 3.4 Aufbau von Programm-(PRG)-Files

Dieser Filetyp ist uns allen bestens bekannt. In PRG-Files werden Computerprogramme mit "SAVE" abgelegt und mit "LOAD" wieder geladen. Die Verkettung von Blöcken wurde schon in Kapitel 3.1.3 behandelt und gilt auch für diesen Filetyp. Eine andere Sache ist aber die Startadresse des Programms im Speicher des Computers. Wie Sie wissen, lassen sich Programme durch LOAD"Name",8 und durch LOAD"Name",8,1 laden, wobei letztere Möglichkeit dann gebraucht wird, wenn es sich um Maschinenprogramme handelt, die in einem ganz bestimmten Speicherbereich laufen.

In jedem ersten Block eines PRG-Files haben Sie nämlich nicht wie allgemein üblich Platz für 254 Daten, sondern nur für 252 Bytes. Den Grund können Sie sich jetzt schon fast denken: In den Bytes 2 und 3 des Blocks steht die Anfangsadresse des Programms im Computer, und zwar enthält Byte 2 dabei den niederwertigen Teil der Adresse. Laden Sie jetzt ein Programm nur mit Gerätenummer 8, wird diese Adresse ignoriert, und das Programm wird an den Beginn des BASIC-Speichers geladen; andernfalls, das heißt mit Sekundäradresse 1, nimmt der Computer als Ladeadresse die auf der Diskette angegebene und schreibt das Programm zum Beispiel ab 49152 ($C000) in den Speicher.

Der letzte Block eines jeden Programms ist, wie allgemein üblich, dadurch gekennzeichnet, daß er als Linker für die Spur den Wert 0 enthält. Die Sektornummer ist dann keine Sektornummer mehr, sondern enthält die Anzahl der belegten Bytes im letzten Block.

## 3.5 Aufbau von sequentiellen (SEQ) Files

Mit den sequentiellen Files haben wir uns in Kapitel 2.3 schon auseinandergesetzt, nur ging es dort nicht um deren Aufbau, sondern rein um die Methode der Datenspeicherung. Jetzt soll uns interessieren, wie die Floppy die Daten abspeichert, die sie durch den PRINT#-Befehl zugeschickt bekommt.

SEQ-Files stellen den Standardtyp der Datenspeicherung dar und sind deshalb sehr einfach aufgebaut. Die Blöcke sind durch die übliche Verkettung miteinander verbunden. Auch die Endmarkierung ist identisch mit der der PRG-Files.

Der einzige Unterschied zu den PRG-Files besteht im ersten Datenblock, in dem bei den SEQ-Files ganze 254 Bytes Daten abgelegt werden können, da sie keine Anfangsadresse benötigen.

## *3.6 Aufbau von User-(USR)-Files*

Die USR-Files stellen eine Sonderform der SEQ-Files dar. Sie sind im Aufbau mit diesen identisch, dienen jedoch anderen Zwecken, wie das schon erwähnte Umleiten der Bildschirmausgabe, um ein Spooling durchführen zu können.

Auf die Anwendung von USR-Files werden wir noch öfter zu sprechen kommen.

## *3.7 Aufbau von relativen (REL) Files*

Dieser Filetyp ist mit Abstand der komplizierteste, wie schon die Handhabung relativer Dateien vermuten läßt. Das aufwendige an dieser Datenspeicherung ist, daß sie aus mehreren, voneinander unabhängigen, Datensätzen besteht, wobei auf jeden davon schnell und völlig willkürlich zugegriffen werden soll.

### 3.7.1 Aufbau der Datenblöcke

Da die Recordlänge bei relativen Dateien maximal 254 betragen darf, werden Sie schon vermuten, wie ein Datenblock bei relativen Dateien aufgebaut sein muß. Er ist in der Tat identisch mit dem der SEQ-Files. Nun können Sie sagen: Warum denn eine Blockverkettung, wenn sowieso jeder Datensatz einzeln angesprochen wird und nicht länger als 254 Bytes sein darf? Nun, die Sache ist die:

Wenn Datensätze kürzer als 254 Bytes lang sind (zum Beispiel nur 200 Bytes), benutzt das DOS nicht für jeden Datensatz einen eigenen Block, sondern schreibt eben die ersten 54 Bytes des nächsten Datensatzes noch zu den ersten 200 Bytes dazu. Wird jetzt aber der zweite Datensatz angesprochen, muß das DOS wissen, wo der restliche Teil dieses Satzes zu finden ist. Diese Angabe enthalten die ersten beiden Bytes des Blocks.

### 3.7.2 Aufbau der Side-Sektor-Blöcke

Erinnern Sie sich? In Kapitel 3.3 hörten Sie das erstemal etwas über sogenannte Side-Sektor-Blöcke, die bei den relativen Dateien eine Rolle spielen. Hier soll nun der Aufbau dieser 'geheimnisvollen' Einrichtung beschrieben werden, mit deren Hilfe das DOS die Datensätze schnell wiederfindet.

Ein Side-Sektor besteht je nach der Anzahl der Datensätze aus maximal sechs Side-Sektor-Blöcken. Tabelle 3.7 zeigt den Inhalt eines Side-Sektor-Blocks. Diese Blöcke enthalten nun die Track- und Sektornummern sämtlicher Datensätze. Somit muß das DOS nur die Nummer des Datensatzes wissen und kann dann im Handumdrehen auf den entsprechenden Datenblock zugreifen, ohne erst lange suchen zu müssen.

| Bytes von bis | Bedeutung der Bytes |
|---|---|
| 000 - 001 | Track- und Sektornummer des nächsten Side-Sektor-Blocks |
| 002 - 002 | Nummer dieses Side-Sektor-Blocks (0 bis 5) |
| 003 - 003 | Recordlänge der Datei |
| 004 - 015 | Track- und Sektornummern von allen angelegten Side-Sektor-Blöcken |
| 016 - 255 | Zeiger auf 120 Datenblöcke (jeweils Track und Sektor) |

Tabelle 3.7 Inhalte eines Side-Sektor-Blocks

Der Aufbau eines Side-Sektor-Blocks sei im folgenden kurz dargestellt.

Die Bytes 0 und 1 enthalten, wie sollte es auch anders sein, den Zeiger auf den nächsten Side-Sektor-Block, beziehungsweise Byte 0 enthält 0, wenn kein weiterer Block vorhanden ist. Ist kein weiterer Block vorhanden, enthält außerdem Byte 1 die Anzahl der benutzten Bytes dieses Blocks.

Byte 2 enthält die Nummer des Side-Sektor-Blocks.

Die Datensatzlänge, die schon im Fileeintrag zu finden ist, steht nochmal in Byte 3 des Side-Sektor-Blocks.

Die nächsten Bytes der Nummern 4 bis 15 enthalten Track- und Sektornummern der gesamten 6 Side-Sektor-Blöcke, falls diese vorhanden sind.

Ab Byte 16 sind jetzt die Spur- und Sektornummern abgespeichert, unter denen die Datensätze zu finden sind. Es können maximal 120

Datensätze von einem Side-Sektor-Block verwaltet werden. Aus dieser Zahl folgt die maximale Anzahl von Bytes, die eine relative Datei auf der 1541 enthalten kann: 120*6*254 = 182880 Bytes. Das ist mehr als die gesamte Diskettenkapazität einer 1541, da die Floppy auf einer Diskette gerade 174848 Bytes unterbringt.

Um den Zugriff des DOS auf einen bestimmten Datenblock anschaulich zu machen, möchte ich ihn an einem Beispiel verdeutlichen:

Nehmen wir an, wir haben eine relative Datei mit 250 Datensätzen zu je 127 Bytes eröffnet und wollen nun auf den 248. Datensatz zugreifen. Zum Verwalten dieser relativen Datei benötigt das DOS 125 Datenblöcke und 2 Side-Sektor-Blöcke. Beim Zugriff geschieht nun folgendes: Das DOS errechnet die Nummer des Datenblocks aus (248-1/127 = 123.5 (minus 1, da die Zählung immer bei Null beginnt). Unser Datensatz befindet sich also im 123. Datenblock. Da ein Side-Sektor-Block nur 120 Einträge aufnehmen kann, finden wir Track- und Sektornummer unseres Datenblocks in Side-Sektor-Block Nummer 1. Dieser Datenblock wird jetzt anhand der Angabe gelesen und dann die Byteposition unseres Datensatzes anhand der obigen Nachkommastelle mit 254*0.5 = 127 errechnet. Jetzt addieren wir noch den Wert 2 zu unserem Ergebnis, da die beiden ersten Bytes des Blocks keine Daten sind, und wir haben unseren Datensatz ab dem 129. Byte im 123. Datenblock gefunden.

Diese Methode mag ziemlich kompliziert erscheinen, ist jedoch um einiges schneller, als etwaige 123 Blöcke einer sequentiellen Datei nacheinander zu durchsuchen.

## *3.8 Aufbau von deleted (DEL) Files*

Dieser Filetyp existiert in der Anzeige normalerweise nicht, da er ein gelöschtes File angibt. Wird im Directory ein File gelöscht, ersetzt das DOS einfach den Filetyp mit dem Wert 0; eine weitere Anzeige im Inhaltsverzeichnis wird damit unterdrückt. Sie können eine Anzeige aber erzwingen, indem Sie diesen Wert wieder durch 128 ($80) ersetzen. Es erfolgt dann eine DEL-Anzeige im Directory.

Wie man die Floppy derart manipuliert, erfahren Sie im folgenden Kapitel über den Direktzugriff.

# 4
# Direktzugriff auf die Floppy 1541

# 4 Direktzugriff auf die Floppy 1541

## *4.1 Was ist Direktzugriff?*

Der Direktzugriff auf die Floppystation ist der Einstieg zur professionellen Programmierung und Anwendung der 1541. Es handelt sich hierbei um die Ausnutzung eines Befehlssatzes, der dazu dient, zum Beispiel nur einzelne Blöcke auf der Diskette anzusprechen oder eigene Programme in die Floppy einzuspeichern und dort auszuführen.

Die Krönung dieser Programmiertechnik, auf die wir später dann sehr intensiv eingehen werden, wird die direkte Manipulation des DOS sein. Auf diese Weise funktionieren zum Beispiel Programmschutztechniken und schnelle Kopierprogramme.

## *4.2 Die Direktzugriffsbefehle*

Die Befehle, die im folgenden erläutert werden, zählen zu den interessantesten Eigenschaften, die die 1541 besitzt, da wir mit deren Hilfe selbst komplizierte Projekte durchführen können. Es kann durchaus sein, daß Sie mit ein paar Ausdrücken noch Schwierigkeiten haben, da diese noch nicht intensiv genug besprochen wurden. Darauf komme ich noch im Anschluß an die Direktzugriffsbefehle zu sprechen.

### 4.2.1 Öffnen eines Direktzugriffskanals

Als wir in den vorhergehenden Kapiteln Daten auf eine Diskette speichern wollten, mußten wir zuerst eine Datei auf der Diskette öffnen, um die Werte dort abzulegen. Dieses Anlegen einer 'Datei' ist auch beim Direktzugriff erforderlich, nur existiert diese 'Datei' jetzt nicht mehr auf Diskette und heißt auch nicht mehr 'Datei', sondern es handelt sich hier um einen sogenannten Direktzugriffskanal. Dieser Kanal reserviert einen Teil des Speichers in der Floppy, wo die weiteren Vorgänge dann ablaufen werden.

Das Öffnen eines Direktzugriffskanals erfolgt mit dem OPEN-Befehl, dessen Synatx hierfür lautet:

OPEN File#, Geräte#, Kanal#, "#"

Hierbei bedeuten:

File# - die logische Filenummer (1-127)
Geräte# - die Gerätenummer der Floppy (normalerweise 8)
Kanal# - die Nummer des Direktzugriffskanals (2-14)

Wie Sie sicherlich bemerken, ist dies genau die Methode, um eine ganz normale Datei auf der Diskette zu öffnen. Den Ausschlag, daß es sich hier um einen Direktzugriff handelt, gibt nur der 'Filename' in den Anführungszeichen. Aus diesem Grund sollten Sie auch keine Files auf Ihrer Diskette haben, die mit '#' beginnen, da das beim Laden Schwierigkeiten machen könnte.

Wie oben erwähnt, reserviert dieser Befehl einen bestimmten Speicherbereich in der Floppy für die künftigen Aktionen. Diesen Speicherbereich, um den es sich dreht, nennt man Pufferspeicher oder einfach Puffer. Die Floppy hat deren 5, so daß Sie sich unter Umständen aussuchen können, welchen der fünf Puffer Sie benutzen wollen. 'Unter Umständen' deshalb, weil die Floppy einige Pufferspeicher für ihren 'eigenen Bedarf' benötigt und dieses Wählen deshalb nicht immer möglich ist. Wollen Sie dies dennoch, da es auch für später gezeigte Anwendungen wichtig ist, können Sie eine Puffernummer von 0 bis 4 zum Öffnen angeben, zum Beispiel:

OPEN1,8,3,"#3"

Normalerweise sollten Sie aber dem DOS die Wahl des Puffers überlassen, da es dann keine Konflikte geben kann.

Was es genau mit den Puffern auf sich hat erfahren Sie in Kapitel 5.1.

Natürlich ist es Ihnen möglich, festzustellen, welchen Puffer die Floppy für Sie reserviert hat. Dazu genügt es, wenn Sie im Anschluß an das Öffnen des Direktzugriffskanals den Kommandokanal abfragen. Hier wird die Nummer des gewählten Puffers übergeben.

Nun aber zu den Direktzugriffsbefehlen, bei denen man zwei Arten unterscheiden muß: die sogenannten BLOCK-Befehle, die sich immer auf einen Block auf der Diskette beziehen und die MEMORY-Befehle, die sich direkt auf den Speicher der Floppy beziehen.

### 4.2.2 Der BLOCK-READ-(B-R)-Befehl

Dieser Befehl liest, wie man schon am Namen erkennt, einen Block von der Diskette in den reservierten Pufferspeicher, wo er dann von Ihnen abgeändert werden kann. Der B-R-Befehl bezieht sich auf sämtliche Blöcke der Diskette, also auch auf Track 18, was Ihnen riesige Möglichkeiten in der Diskettenorganisation eröffnet. Seine Syntax lautet:

B-R Kanal# Laufwerk# Track# Sektor#

Diese Syntax wird jedoch nicht benutzt. Der B-R-Befehl wird durch einen anderen Befehl ersetzt, da er einen Mangel aufweist (siehe Kapitel 12):

U1 Kanal# Laufwerk# Track# Sektor#

Der U1-Befehl gehört zu den User-Befehlen in Kapitel 4.3. Die einzelnen Bezeichnungen bedeuten:

Kanal# - die Nummer des Direktzugriffskanals (2-14)
Laufwerk# - Bei der 1541 immer 0
Track# - Track des zu lesenden Blocks (1-35)
Sektor# - Sektor des zu lesenden Blocks (0-16, 17, 18, 20)

Es sollte beachtet werden, daß die Direktzugriffsbefehle immer über den Kommandokanal geschickt werden, der mit dem Direktzugriffskanal nichts zu tun hat. Haben Sie den gewünschten Block in den Pufferspeicher der Floppy gelesen, können Sie die Daten mit einer GET#-Schleife auslesen.

Hier ein kleines Beispiel:

Dieses BASIC-Programm liest Block 18,0 von der Diskette und zeigt Ihnen die beiden ersten Bytes an, die, wie Sie wissen, die Linker auf den nächsten Block enthalten:

```
10 OPEN1,8,2,"#"
20 OPEN2,8,15
30 PRINT#2,"U1 2 0 18 0"
40 GET#1,A$,B$
50 PRINT ASC(A$),ASC(B$)
```

Starten Sie dieses Programm, erhalten Sie folgendes Bild;

```
18    1
```

Selbstverständlich können Sie Track- und Sektornummer in einer Variablen außerhalb der Anführungszeichen übergeben:

```
PRINT#2,"U1 2 0";TRACK;SEKTOR
```

Das Einlesen der Daten aus dem Puffer mittels GET# hat nur zwei kleine Haken: wenn ein sogenanntes Nullbyte, das heißt CHR$(0) über den Bus kommt, erhält die Variable hinter GET# kein Zeichen sondern einen Leerstring (""), der bei dem ASC-Befehl einen '?ILLEGAL QÜANTITY ERROR' auslöst und deshalb abgefragt werden müßte. Es gibt aber eine andere sehr elegante Methode:

```
40 GET#1,A$
50 PRINT ASC(A$+CHR$(0))
```

Der zweite Haken ist der, daß Sie bisher alle Bytes sequentiell von der Floppy holen mußten. Das heißt, wenn Sie das 200. Byte erhalten wollen, müssen Sie vorher alle anderen 199 Bytes mitlesen. Dieses Problem läßt sich ebenfalls einfach lösen.

### 4.2.3 Der BUFFER-POINTER-(B-P)-Befehl

Der B-P-Befehl gestattet es Ihnen, vorher festzulegen, welches Byte Sie als nächstes aus dem Speicher lesen wollen. Die Syntax ist denkbar einfach:

B-P Kanal# Position#

Hierbei bedeuten:

Kanal# - die Nummer des Direktzugriffskanals (2-14)
Position# - Die Nummer des Bytes, auf das Sie positionieren wollen (0-255)

Sie können sich nun die Bytes herauspicken, die Sie interessieren und dabei beliebig im Speicher der Floppy herumspringen. Gelangen Sie beim Lesevorgang über das 255. Byte hinaus, beginnt der Zeiger wieder mit 0.

### 4.2.4 Der BLOCK-WRITE-(B-W)-Befehl

Hier haben wir das Gegenstuck zum B-R-Befehl. Mit diesem Befehl können Sie den Inhalt des reservierten Puffers (256 Bytes) an jede beliebige Position auf Diskette schreiben. Auch dieser Befehl hat einen Mangel, der durch den Ersatz mit dem U2-Befehl aufgehoben wird. Die Syntax lautet:

U2 Kanal# Laufwerke Track# Sektor#

Hierbei bedeuten:

Kanal# - die Nummer des Direktzugriffskanals (2-14)
Laufwerk# - Bei der 1541 immer 0
Track# - Track auf den der Block geschrieben werden soll (1-35)
Sektor# - Sektor, der beschrieben werden soll (0-16, 17, 18, 20)

Es ist natürlich möglich, einen Puffer mit dem PRINT#-Befehl zu beschreiben und dann auf Diskette zu bringen.

Anhand der Erläuterungen von Block lesen und Block schreiben ist Ihnen sicher schon aufgefallen, daß man nur jeweils ganze Blöcke lesen oder schreiben kann, was in der Eigenschaft der Floppystation liegt, die blockweise und nicht byteweise arbeitet.

Das nachfolgende Beispiel zeigt, wie man etwas in den Pufferspeicher und dann auf Diskette schreibt:

```
10 OPEN1,8,2,"#"
20 OPEN2,8,15
30 PRINT#1,"DIES IST EIN TEST!"
40 PRINT#2,"U2 2 0 1 0"
```

Hier wird der Text 'DIES IST EIN TEST!' in den Block 0 auf Track 1 geschrieben.

### 4.2.5 Der BLOCK-ALLOCATE-(B-A)-Befehl

Wir gehen jetzt einmal davon aus, Sie hätten 10 Blöcke im Direktzugriff auf Diskette geschrieben und diese Diskette dann weggelegt. Jetzt holen Sie die Diskette wieder hervor und schreiben ein Programm mit SAVE darauf. Wenn Sie Pech haben, kann Ihnen jetzt ein Verzweiflungsschrei auch nicht mehr helfen; Sie haben Ihre 10 Blöcke, die Arbeit von Stunden, gelöscht.

Ihr Pech war, daß Sie dem DOS nicht mitgeteilt hatten, daß Sie diese Blöcke beschrieben haben.

Um diesem Übel abzuhelfen gibt es den B-A-Befehl. Mit diesem Kommando ist es möglich, einen Block in der BAM als belegt zu kennzeichnen, damit er bei späteren Schreibvorgängen nicht überschrieben werden kann. Die Syntax lautet:

B-A Laufwerk# Track# Sektor#

Die Bedeutung der Werte ist:

Laufwerk# - Bei der 1541 immer 0
Track# - Track des zu belegenden Blocks (1-35)
Sektor# - Sektornummer des zu belegenden Blocks (0-16, 17, 18, 20)

Wollen wir einen Block belegen, der bereits belegt war, gibt die Floppy eine Fehlermeldung aus: 'NO BLOCK'. Die Track- und die Sektornummer gibt dann den nächsten freien Block auf Diskette an.

Achtung: Eine Diskette, die direktgeschriebene Blöcke enthält, die nicht im Directory verzeichnet und als File miteinander verkettet sind, darf nicht durch den Befehl 'V' 'aufgeräumt' werden, da dieser die Blöcke in der BAM wieder freigibt!

### 4.2.6 Der BLOCK-FREE-(B-F)-Befehl

Der B-F-Befehl ist das genaue Gegenstuck zum B-A-Befehl; er erlaubt das Wiederfreigeben eines Blocks in der BAM. Die Syntax lautet:

B-F Laufwerk# Track# Sektor#

Es bedeutet:

Laufwerk# - Bei der 1541 immer 0
Track# - Track des freizugebenden Blocks (1-35)
Sektor# - Sektornummer des freizugebenden Blocks (0-16, 17, 18, 20)

Die Anwendung dieses Befehls erfolgt analog zum B-A-Befehl.

## 4.2.7 Der MEMORY-READ-(M-R)-Befehl

Als nächstes steht ein MEMORY-Befehl auf dem Programm. Mit dem M-R-Befehl können Sie wie mit dem PEEK-Befehl beim Computer Speicherinhalte lesen. Im Fall der 1541 heißt das, daß Sie sich den gesamten Adreßbereich der Floppystation (das sind 64 KByte) ansehen können.

Die Syntax dieses Befehls ist in den Handbüchern falsch beschrieben und lautet in Wirklichkeit:

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

Dabei bedeuten:

```
ADL         - das niederwertige Byte der Speicheradresse
ADH         - das höherwertige Byte der Speicheradresse
Anzahl      - die Anzahl der zu lesenden Bytes
```

Das Aufspalten einer Speicheradresse zwischen 0 und 65535 in ADL und ADH sollte Ihnen inzwischen geläufig sein; der Wiederholung halber hier noch einmal die Formel:

A ist der Wert der Adresse, dann sind:

```
ADH = INT(A/256)
ADL = A-ADH*256
```

Zur Verdeutlichung wollen wir einmal den Inhalt der Speicherzelle 119 ($0077) auslesen. Diese Speicherstelle enthält die Gerätenummer der Floppy + 32, das heißt, wir müßten im Normalfall den Wert 40 (32+8) erhalten:

```
OPEN 1,8,15
PRINT#1,"M-R"CHR$(119)CHR$(0)CHR$(1)
GET#1,A$
PRINTASC(A$+CHR$(0))
```

Wie Sie aus diesem Programmbeispiel ersehen, werden die Werte, die man mit dem M-R-Befehl aus dem Speicher der Floppy liest, über den Kommandokanal zum Computer geschickt.

## 4.2.8 Der MEMORY-WRITE-(M-W)-Befehl

Na also, hier haben wir nun das Gegenstück zu unserem vorherigen PEEK-Befehl, nämlich den POKE-Befehl für die Floppystation. Wie unschwer zu erraten ist, kann man mit dem M-W-Befehl die Inhalte von Speicherstellen in der Floppy verändern. Die Syntax lautet hier:

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

Die Bedeutungen sind:

```
ADL         - das niederwertige Byte der Anfangsadresse
ADH         - das höherwertige Byte der Anfangsadresse
Anzahl      - die Anzahl der zu schreibenden Bytes
DATA        - Die entsprechend zu schreibenden Daten
```

Wie man sieht, können mit diesem Befehl gleich mehrere Bytes gleichzeitig zur Floppy geschickt und in deren Speicher geschrieben werden. In unserem nächsten Beispiel ändern wir die Gerätenummer unserer 1541 von bisher 8 auf den neuen Wert 9. Diese Einstellung bleibt bis zu einem RESET der Floppy erhalten und sollte immer dann angewendet werden, wenn Sie mit mehreren Laufwerken gleichzeitig arbeiten wollen:

```
OPEN 1,8,15
PRINT#1,"M-W" CHR$(119)CHR$(0)CHR$(2)CHR$(41)CHR$(73)
CLOSE1
```

An dieser Befehlssequenz erkennen Sie sofort, daß es nicht genügt, eine Gerätenummer zu ändern, sondern wir müssen zwei verschiedene Werte in zwei verschiedene Speicherstellen schreiben. Das liegt daran, daß die Floppy sowohl für den Lese- als auch für den Schreibbetrieb eine eigene Gerätenummer besitzt. Die Nummer zum Lesen liegt in Speicherstelle 119. Zu ihr wird 32 addiert. Die Nummer für Schreiben liegt bei der Adresse 120 und wird mit 64 addiert. Tippen Sie jetzt

```
LOAD"$",8
```

wird sich ihr Computer dafür mit einem '?DEVICE NOT PRESENT ERROR' bedanken.

Die neue Version muß lauten:

```
LOAD"$",9
```

An dieser Stelle sei wieder einmal auf die Programme im Anhang dieses Buches hingewiesen. Mit Ihren Kenntnissen durfte es jetzt kaum noch ein Problem sein, die Funktionsweise dieser Programme zu durchschauen.

### 4.2.9 Der MEMORY-EXECUTE-(M-E)-Befehl

Mit dem Lesen dieser Überschrift haben Sie sich unbemerkt in das Reich der professionellen Programmierung begeben. Ab jetzt werden wir lernen, wie man einem Gerät die allerletzten Geheimnisse entlockt und sich diese zu Nutze macht. Bevor wir in diese Programmiertechniken einsteigen, müssen wir noch die Befehle besprechen, die uns das ermöglichen. Wir fangen mit dem M-E-Befehl an.

Der M-E-Befehl gestattet uns, Maschinenprogramme, die in der Floppy gespeichert sind, dort auszuführen. Dieser Befehl entspricht in BASIC dem SYS-Befehl und erlaubt uns, sowohl eigene Programme, als auch Subroutinen des DOS aufzurufen. Die einzige Bedingung, die an diese Routinen gestellt wird, ist, dass sie mit RTS (Return from Subroutine, $60) abgeschlossen sind. Die Syntax des Befehls lautet wie folgt:

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

Dabei bedeuten:

```
ADL         - das niederwertige Byte der Startadresse
ADH         - das höherwertige Byte der Startadresse
```

Wie Sie wissen, hat die Floppystation 1541 eine Fehlerroutine im DOS, die einen auf getretenen Fehler mit einer blinkenden Leuchtdiode am Laufwerk quittiert. So einen 'Defekt' wollen wir mit unserem neuen Befehl simulieren, indem wir im DOS die Fehlerroutine aufrufen. Sie steht ab Adresse 49452 ($C12C):

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

Geben Sie diese Zeilen ein, beginnt die Leuchtdiode am Laufwerk Ihrer Floppystation sofort munter drauflos zu blinken.

Wie wir noch andere Routinen im DOS für unsere Zwecke benutzen können, erfahren Sie im weiteren Verlauf dieses Buches, das zu diesem Zweck auch mit einem voll kommentierten ROM-Listing der 1541 ausgestattet ist und daher für fortgeschrittene Maschinenspracheprogrammierer die ideale Grundlage darstellt.

## 4.2.10 Der BLOCK-EXECUTE-(B-E)-Befehl

Dieser Befehl ist gewissermaßen eine Erweiterung des M-E-Befehls. Er erfüllt jedoch den gleichen Zweck. Es geht auch hier darum, ein Maschinenprogramm im Puffer der Floppystation zu starten. Der einzige Unterschied besteht darin, daß dieser Befehl die Kombination des B-R und des M-E-Befehls darstellt. Das heißt, es wird ein Block von der Diskette in den reservierten Puffer der Floppy geladen und anschließend dort automatisch ausgeführt. Die Syntax dieses Befehls lautet:

```
B-E Kanal# Laufwerk# Track# Sektor#
```

Dabei bedeuten:

```
Kanal#      - die Nummer des Direktzugriffskanals (2-14)
Laufwerk#   - Bei der 1541 immer 0
Track#      - die gewünschte Tracknummer (1-35)
Sektor#     - der gewünschte Sektor (0-16, 17, 18, 20)
```

Da Maschinenprogramme nur selten vollständig relokativ (verschiebbar) sind, wird man beim Eröffnen des Direktzugriffskanals einen festen Puffer reservieren, um das Programm ausführen zu können.

Auf die Technik, wie man Programme in den Speicher der Floppy schreibt und dort ausführt, werden wir im Verlauf dieses Buches noch sehr ausführlich zu sprechen kommen.

## 4.3 Die USER-(U)-Befehle

Die U-Befehle stellen eine weitere Einrichtung der Floppy dar, Maschinenprogramme im Speicher laufen zu lassen. Tabelle 4.1 zeigt eine Aufstellung aller USER-Befehle mit deren Bedeutung.

| USER-Befehl | Startadresse und Bedeutung |
|---|---|
| U1 UA | $CD5F Ersatz für BLOCK-READ |
| U2 UB | $DC97 Ersatz für BLOCK-WRITE |
| U3 UC | $0500 Start in Benutzerpuffer |
| U4 UD | $0503 Start in Benutzerpuffer |
| U5 UE | $0506 Start in Benutzerpuffer |
| U6 UF | $0509 Start in Benutzerpuffer |
| U7 UG | $050C Start in Benutzerpuffer |
| U8 UH | $050F Start in Benutzerpuffer |
| U9 UI | $FF01 NMI-Vektor; Betriebsmodus umschalten |
| U: UJ | $EAA0 RESET-Vektor |

Tabelle 4.1 Startadressen der USER-Befehle

Wie Sie sehen, kann ein U-Befehl sowohl mit Zahlen und Satzzeichen, als auch mit Buchstaben zur Kennung aufgerufen werden. Wir werden uns im weiteren Verlauf des Buches auf die Zahlen und Satzzeichen beschränken.

Die Befehle U1 und U2 sind Ihnen ja schon bestens bekannt. Neu ist der Rest des Befehlssatzes, der sich hauptsächlich auf den sogenannten Benutzerpuffer der Floppy bezieht. Die Befehle U3 bis U8 starten ein Programm an den angegebenen Adressen und ermöglichen somit das Einrichten einer Sprungtabelle, da die Abstände der Einsprungadressen immer genau 3 Bytes auseinanderliegen, was der Länge eines Sprungbefehls in Maschinensprache entspricht.

Mit dem Befehl U9 hat es eine besondere Bewandtnis. Da die beiden Computer Commodore 64 und VC 20 unterschiedlich aufgebaut sind, unterscheiden sich auch deren Busroutinen in der Geschwindigkeit voneinander. Der VC 20 bei der Datenübertragung schneller als der Commodore 64. Deshalb hat die 1541 eine Einrichtung, die das Umschalten zwischen VC 20 und C 64 Betrieb ermöglicht. Zu diesem Zweck dient der U9-Befehl.

Schicken Sie den Befehl "U9+" zur Floppy, erfolgt die Umschaltung auf C 64 Betrieb; bei dem Befehl 'U9-' arbeitet die Floppy im VC 20 Modus und hat dann eine etwas schnellere Busbehandlung.

Der U: oder UA-Befehl schließlich bewirkt bei der Floppy einen Sprung zum RESET-Vektor und bringt diese dadurch wieder in den Einschaltzustand.

## *4.4 Ein Sonderling: der &-Befehl*

Von diesem Befehl haben Sie bisher sicher noch nichts gehört, da er ebenso wie die relative Datenspeicherung nirgends erwähnt wird und deshalb praktisch unbekannt ist.

Wenn man mit diesem Befehl arbeiten will, sind einige Regeln zu beachten, die seine Anwendung ein wenig kompliziert erscheinen lassen.

Der &-Befehl gleicht in seiner Arbeitsweise dem BLOCK-EXECÜTE Befehl. Auch hier wird ein Programm von der Diskette in den Pufferspeicher der Floppy gelesen und dann dort ausgeführt. Der Unterschied zum B-E Befehl besteht darin, daß der &-Befehl sich nicht nur auf einen bestimmten Block einer Diskette beschränkt sondern eine Datei behandelt, die anhand ihres Filenamens aufgerufen wird und aus mehreren Blöcken bestehen kann.

Das Kennzeichen einer Datei, auch als &-File bezeichnet, die vom &-Befehl aufgerufen wird, ist das erste Zeichen des Dateinamens. Es handelt sich hier immer um das '&'-Zeichen, so daß das DOS sofort weiß, daß man es hier mit einem &-File zu tun hat. Ein Dateiname wäre also beispielsweise: "&TEST".

Wollen Sie Ihr File mit dem &-Befehl aufrufen, genügt die Angabe des Filenamens über den Kommandokanal, in unserem Falle zum Beispiel:

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

Die Floppy sucht jetzt nach diesem Namen im Directory und lädt das Programm, wenn es gefunden wird, in ihren Pufferspeicher, wo es schließlich abläuft.

Jetzt wollen wir uns noch mit dem Aufbau eines solchen Files beschäftigen, der wie schon erwähnt, ein wenig kompliziert ist. Tabelle 4.2 zeigt den Aufbau eines Blocks, der ein &-File enthält.

| Byte | Bedeutung |
|---|---|
| 0 | Tracknummer des nächsten Blocks im File |
| 1 | Sektornummer des nächsten Blocks im File |
| 2 | Startadresse Low des Programms im Speicher |
| 3 | Startadresse High des Programms im Speicher |
| 4 | Anzahl der Bytes im Programm |
| 5.. | Programmbytes |
| ..X | letztes Programmbyte |
| X+1 | Prüfsumme über den Programmteil |
| X+2 | Startadresse Low des Programms im Speicher |
| X+3.. | Startadresse High.. |

Tabelle 4.2 zeigt den Aufbau eines &-Files

Wie Sie aus der Tabelle ersehen können, sind Sie nicht an eine einmal angegebene Startadresse gebunden. Sie können in einem einzigen &-File theoretisch beliebig viele Programmteile unterbringen, wobei jedoch ein Programmteil nicht länger als 255 Zeichen sein darf, da die Anzahl der Zeichen in einem einzigen Byte abgespeichert ist.

Die Berechnung der Prüfsumme am Ende jedes Programmteils ist etwas schwierig; man gewohnt sich jedoch schnell an deren Handhabung. Zur Berechnung der Prüfsumme werden alle Programmbytes, die Anzahl der Programmbytes und die beiden Bytes der Startadresse aufaddiert. Wichtig ist hierbei, daß nach jeder weiteren Addition eines Bytes sofort der Übertrag (Carry-Flag) hinzugezählt wird. An einem Beispiel will ich dieses Schema verdeutlichen:

Sie haben die Adresse Lo = $90. Dazu soll jetzt Adresse Hi addiert werden; diese beträgt $AA. $90+$AA=$3A + Übertrag 1. Nach der Berechnungsmethode der Prüfsumme müssen Sie den Übertrag noch hinzuaddieren, damit Sie das richtige Ergebnis erhalten: $3A+$01=$3B. So verfahren Sie bei jeder weiteren Addition eines Bytes bis hin zum letzten Programmbyte und speichern das Ergebnis als Prüfsumme hinter dem letzten Programmbyte ab.

Achtung! Die Prüfsumme ist nicht in der Gesamtanzahl der Programmbytes enthalten.

Wollen Sie die Prüfsumme in Maschinensprache errechnen, ist dies ganz einfach:

```
LDA letztes Ergebnis ;
CLC                  ; Addition vorbereiten
ADC neuer Wert       ; nächstes Byte addieren
ADC #$00             ; Carry addieren
STA neues Ergebnis   ; als neues Ergebnis abspeichern
RTS                  ; Ende
```

Diese Routine finden Sie fast identisch im DOS der 1541 vor, da damit Ihre Prüfsumme nachgerechnet und auf Richtigkeit kontrolliert wird (Adresse $E84B).

Ich möchte an dieser Stelle noch auf ein paar seltsame Fehlermeldungen der Floppy hinweisen, die erscheinen, falls Sie beim Aufbau eines &-Files einen Fehler gemacht haben:

- RECORD NOT PRESENT erscheint, wenn Sie die Prüfsumme nicht richtig ausgerechnet haben
- OVERFLOW IN RECORD erscheint, wenn die Anzahl Ihrer Programmbytes nicht mit der angegebenen übereinstimmt

Das DOS ist ziemlich pingelig, wenn es um die korrekte Syntax eines &-Files geht. Es eröffnet Ihnen damit aber andererseits vielfältige Möglichkeiten der DOS-Manipulation mit Hilfe nachgeladener Programmteilen, so daß Sie sich ein komplett neues Betriebssystem schreiben und auf einer Systemdiskette ablegen können. Die benötigten Routinen werden dann jeweils nachgeladen.

# 5
# Die Speicherorganisation der 1541

# 5 Die Speicherorganisation der 1541

Dieses Kapitel ist für all jene Programmierer bestimmt, die schon Erfahrung in der Programmierung gemacht haben und zwar betrifft dies sowohl den Computer, als auch die 1541. Sie sollten sich schon mit der Maschinenspracheprogrammierung auseinandergesetzt haben und grundlegende Kenntnisse über Prozessorsysteme besitzen. Ist dies bei Ihnen nicht der Fall, sollten Sie sich nicht scheuen, den Einstieg in die Maschinensprache jetzt zu wagen, sofern Sie die Programmierung in BASIC schon beherrschen. Wenn Sie weiterhin intensiv mit Ihrem Computer arbeiten wollen, werden Sie um diesen Schritt sowieso nicht herumkommen. Es ist aber von Vorteil, die Maschinensprache möglichst früh zu lernen, da sie einen Einblick in die Arbeitsweise des Computers erfordert und Sie somit anhält, mehr über dessen Funktionsweise zu erfahren. Außerdem läßt sich der Befehlssatz der Maschinensprache zwar sehr schnell erlernen; bis zur vollendeten Programmierung ist es dann aber noch ein weiter Weg.

Nun aber zur Organisation der 1541.

Die Floppystation ist nach dem gleichen Prinzip wie Ihr Computer aufgebaut. Sie besitzt einen 6502 Mikroprozessor, der einen Speicherbereich von 64 KByte verwalten kann. Innerhalb dieses adressierbaren Bereichs befinden sich RAM und ROM der Floppy und außerdem zwei Schnittstellenbausteine des Typs VIA 6522.

## *5.1 Der RAM-Bereich der 1541*

Eine der interessantesten Eigenschaften der Floppystation ist mit Sicherheit ihr relativ großer RAM-Bereich, der dem Programmierer, wie schon erwähnt, vielseitige Möglichkeiten bietet.

Der RAM-Bereich bei der 1541 hat eine Kapazität von 2 KByte, deren Einteilung Bild 5.1 zeigt;

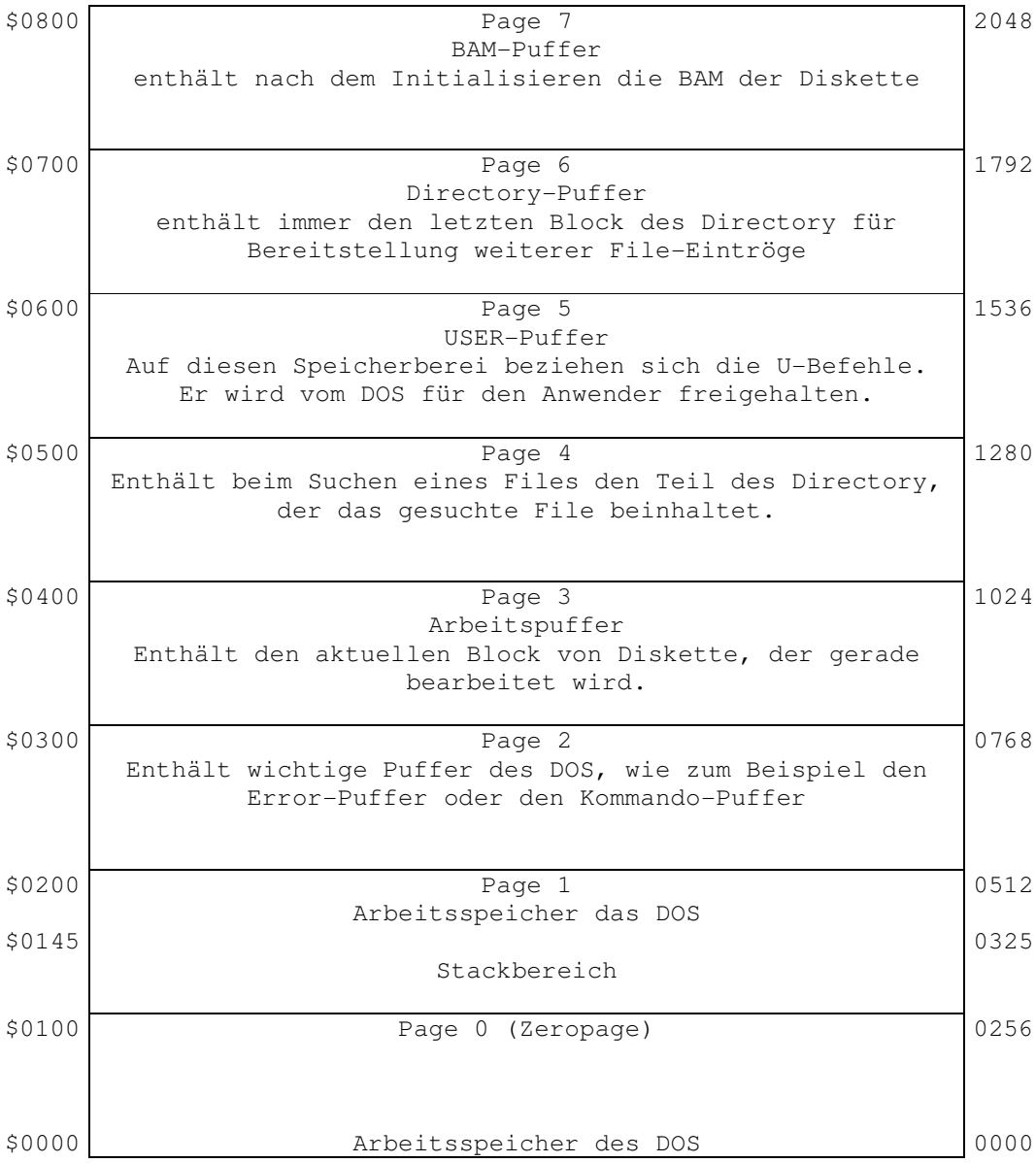



Bild 5.1 Die RAM-Aufteilung der 1541

### 5.1.1 Die Zeropage

Der Speicherbereich ist in 'Seiten' zu je 256 Bytes aufgeteilt. Die Angabe der jeweiligen Seitennummer übernimmt das höherwertige Byte einer Adresse. Die Angabe des eigentlichen Bytes übernimmt das niederwertige Byte der Adresse. Insgesamt haben wir bei einer Kapazität von 64 KBytes Speicher eine Einteilung in 256 Seiten zu je 256 Bytes. Auch bei der Floppy hat dieses System Gültigkeit.

In einem 6502-Prozessorsystem hat die Seite Null, das heißt, die ersten 256 Bytes, immer eine sehr wichtige Funktion als Zwischenspeicher; so auch in der Floppy 1541. Diese Zeropage enthält hier alle wichtigen Zeiger und Register, die das DOS zum Ausführen seiner Arbeit benötigt. Ein Beispiel hatten wir schon bei dem M-W-Befehl, als wir die Gerätenummer der Floppy veränderten. Durch Eingriffe in die Zeropage kann der Programmierer dafür sorgen, daß die gesamte 1541 'aussteigt'. Diese Gefahr besteht beim Computer ebenfalls, so daß man hierbei äußerste Vorsicht walten lassen sollte. Bei der Arbeit in Maschinensprache ist es daher besonders wichtig, daß alle Programme vor dem Ausführen abgespeichert werden, da sie bei einem 'Aussteigen' der Floppy und anschließendem RESET vollständig gelöscht werden.

### 5.1.2 Die Page 1

Die Seite 1 im Speicher der 1541 wird, ebenso wie die Zeropage, vom DOS für das Ablegen wichtiger Werte benutzt. Außerdem befindet sich hier der Hardware-Stack, der vom Mikroprozessor verwaltet wird. Ein Eingriff in diesen Teil des Speichers kann die gleichen Folgen, wie vorhin beschrieben, nach sich ziehen und sollte nur bei guter Kenntnis des DOS gewagt werden.

### 5.1.3 Die Seite 2

Die Speicherseite 2 wird als Arbeitspuffer der Floppy bezeichnet, da sie die Befehlsstrings vom Computer und die Fehlermeldungen des DOS aufnimmt, bevor diese weiterverarbeitet oder ausgelesen werden.

Werden diese Zwischenspeicher während der Ausführung eines Maschinenprogramms nicht benötigt, können sie von einem Anwender mit Erfahrung in der Arbeit mit DOS mitbenutzt werden, ohne die Floppy 'aussteigen' zu lassen.

### 5.1.4 Die Pufferspeicher der 1541

Ab der Adresse $0300, also ab Seite 3, befinden sich bei der Floppy die schon früher angesprochenen Pufferspeicher. In diesem Teil des Speichers werden alle Daten zwischengespeichert, die auf der Diskette geschrieben oder von der Diskette gelesen werden sollen. Tabelle 5.2 zeigt den Bereich der einzelnen Puffer und deren üblichen Inhalt:

| Puffer | Adresse | Funktion |
|---|---|---|
| 0 | $0300-03FF | Hauptarbeitspuffer |
| 1 | $0400-04FF | Enthält aktuellen Directoryblock |
| 2 | $0500-05FF | Benutzerpuffer; normalerweise frei |
| 3 | $0600-06FF | letzter Block des Directory |
| 4 | $0700-07FF | enthält Block 18,0 (BAM) |

Tabelle 5.2 Belegung der Pufferspeicher

Wenn Sie Programme in die Floppy einspeichern wollen, kommt für Sie der Bereich zwischen $0300 und $06FF in Frage. Als Benutzerpuffer eingerichtet ist nur der Puffer 2, von $0500 bis $05FF. Für diesen Puffer ist die Bedienung mit Hilfe der USER-Befehle stark erleichtert.

## *5.2 Die beiden Schnittstellenbausteine der 1541*

Alle für die interne Funktion des DOS wichtigen Speicherabschnitte wurden schon genannt. Jetzt fehlen nur noch die Bausteine im Computersystem, die der Floppy erstens die Bedienung der Diskette und zweitens den Dialog mit dem Computer gestatten. Zu diesem Zweck setzt man bei der 1541 zwei VIA 6522 Schnittstellenbausteine ein.

### 5.2.1 Der Buscontroller (BC) VIA 6522

Der VIA 6522 ist zwar ein vielseitiger Baustein, er wird jedoch in keinem Gerät von Commodore gemäß seiner Eigenschaften eingesetzt. Vielmehr hat es Commodore schon immer recht gut verstanden, diesen Baustein für Steuerzwecke zu mißbrauchen, für die er eigentlich gar nicht konzipiert ist; so auch in der 1541.

Der BC in der 1541 liegt größtenteils brach, wenn man einmal von seinen vielseitigen Funktionen ausgeht. Verwendet wird hauptsächlich der Port B.

Bei einem RESET wird dieser Port B aufgespalten, wobei jedes der acht Pins eine bestimmte Funktion zur Bedienung des seriellen Bus übernimmt. In Tabelle 5.3 können Sie die Verwendung der Pins ablesen:

| Port B Bit | Funktion |
|---|---|
| 0 | Leitung für DATA IN |
| 1 | Leitung für DATA OUT |
| 2 | Leitung für CLOCK IN Signal |
| 3 | Leitung für CLOCK OUT Signal |
| 4 | Leitung für ATN OUT |
| 5 | Hardware Gerätenummer |
| 6 | Hardware Gerätenummer |
| 7/CB2 | Leitung für ATN IN vom seriellen Bus |

Tabelle 5.3 Belegung des Port B des BC

Wie Sie sehen, hat fast jeder Pin des Port B eine Funktion beim Betreiben der seriellen Schnittstelle zwischen Floppy und Computer.

Auch der VC 20-Computer ist mit zwei VIAs 6522 ausgerüstet. Beim Commodore 64 sind die Konstrukteure hingegen einen anderen Weg gegangen. Er enthält zwei völlig neu entwickelte Bausteine: die ClAs 6526.

Für die Programmierung der 1541 genügt es in der Regel, wenn Sie die Belegung des Port B des BC kennen. Der Vollständigkeit halber sei erwähnt, daß der VIA 6522 noch zwei sogenannte Intervalltimer besitzt, die den zeitlichen Ablauf des Busbetriebs steuern.

Der Buscontroller hat die Grundadresse $1800.

## 5.2.2 Der Diskcontroller (DC) VIA 6522

Nun soll uns der zweite Schnittstellenbaustein der Floppy interessieren, dessen Grundadresse bei $1C00 liegt.

Das DOS muß bei seiner Arbeit zwei verschiedene Kontakte mit der Außenwelt herstellen. Es muß einerseits der serielle Bus bedient werden, und andererseits soll noch der Vorgang des Schreibens und Lesens auf die und von der Diskette gesteuert werden.

Zu letzterem Zweck wird der zweite VIA-6522-Baustein verwendet. Auch hier spielt wieder der Port B die Hauptrolle, wobei jeder Pin des Ports eine bestimmte Aufgabe übernimmt. Tabelle 5.4 zeigt die Belegung von Port B:

| Port B Bit | Funktion |
|---|---|
| 0 | Steppermotor für Laufwerk 1 (n.v.)[1] |
| 1 | Steppermotor; Bit=1: Motor ein |
| 2 | Laufwerksmotor; Bit=1; Motor ein |
| 3 | LED am Laufwerk; Bit=1: LED an |
| 4 | Schreibschutz; Bit=1: Lichtschranke frei |
| 5 | Timersteuerung der 4 Diskettenabschnitte |
| 6 | Timersteuerung der 4 Diskettenabschnitte |
| 7 | SYNC Signal beim Lesen von Diskette |

Tabelle 5.4 Belegung der Pins von Port B des DC

Wie Sie sehen, könnten Sie im Prinzip ganz einfach in die Laufwerkssteuerung eingreifen, wenn Sie die entsprechenden Bits setzen oder löschen. Im Kapitel über das Ablegen von Programmen in Pufferspeicher der Floppy bringe ich ein Beispiel, das Ihnen anhand dieser Bits ganz einfach anzeigt, ob sich eine Diskette mit Schreibschutzplakette im Laufwerk befindet, oder ob die Lichtschranke, die diese Plakette überprüft, nicht unterbrochen ist.

[1] @ST: Bit 0 und 1 zusammen bilden die Steppermotorsteuerung für Laufwerk 0. Aufwärtszählen (00, 01, 10, 11, 00) bewegt den Kopf nach innen, Rückwärtszählen (00, 11, 10, 01, 00) nach außen. Vergleiche $FA69.

## *5.3 Der ROM-Bereich der 1541*

Nachdem Sie nun über die Hardware der Floppy informiert wurden, die die notwendigen Steuervorgänge zuläßt, soll uns nun die Software interessieren, die diese gesamten Steuerungen übernimmt.

Ab dem Bereich $C000 (49152) beginnt bei der 1541 der ROM-Bereich. Sie enthält 16 KBytes ROM, in denen das gesamte DOS untergebracht ist. Bis zur Adresse $C100 ist der ROM Bereich ungenutzt. Einen Großteil dieses Buches nimmt das kommentierte DOS-Listing ein, das Sie im Anhang finden und das Ihnen bei der weiteren Arbeit eine unverzichtbare Unterstützung sein wird.

### 5.3.1 Die Aufgaben des DOS

Im bisherigen Verlauf dieses Buches haben Sie schon ein paarmal mit dem DOS zu tun gehabt. Sie erfuhren, daß das DOS Track 18 einer jeden Diskette für den 'eigenen Bedarf' benötigt, und haben die vielfältigen Befehle des DOS kennengelernt. Jetzt soll uns endlich interessieren, was in diesem DOS denn so alles passiert. Was für Vorgänge laufen ab, wenn wir einen Befehl an die Floppy schicken? Was geschieht, wenn wir einen Block in den reservierten Pufferspeicher laden wollen?

Die Aufgaben des DOS sind sehr vielfältig. Sie reichen von der Busbedienung und dem Einschalten der LED am Laufwerk bis hin zur Auswertung von Befehlen und deren Ausführung. Zu diesem Zweck müssen eigentlich mehrere Programme gleichzeitig ausgeführt werden.

## 5.3.2 Die Arbeitsweise des 6502 Mikroprozessors

Wie im vorigen Abschnitt erwähnt, ist es für eine sinnvolle Arbeit mit der 1541 eigentlich notwendig, daß mehrere Programme gleichzeitig ablaufen. Es müssen nämlich folgende Aufgaben schnell bewältigt werden:

- Reagieren auf das ATN-Signal des Computers (Siehe 10.1).
- Entgegennehmen und Auswerten der Befehle.
- Senden oder Empfangen von Daten (LOAD oder SAVE).

gleichzeitig:

- Einschalten des Drivemotors.
- Positionieren des Schreib-/Lesekopfes.
- Lesen oder Schreiben von Daten in die Puffer.

Diese Aufgaben quasi gleichzeitig zu bewältigen ist für einen einzigen 6502 Mikroprozessor, wie er in der 1541 eingebaut ist, ein ziemliches Problem. Die 'großen' CBM-Floppies haben in dieser Hinsicht einen Vorteil: sie besitzen nämlich einen zweiten Mikroprozessor vom Typ 6504, der hier als Diskcontroller arbeitet. Das heißt, dieser Koprozessor erledigt alle Aufgaben, die mit der Steuerung des Laufwerks zu tun haben.

Bei der 1541 ging man deshalb einen anderen Weg. Man benutzt hier die Interrupt-Technik. Ein 6502 Mikroprozessor hat für diese Möglichkeit zwei Interruptarten anzubieten; den IRQ und den NMI.

Das Kürzel IRQ steht für 'Interrupt request', was mit 'Interruptanforderung' zu übersetzen ist. Geht diese Leitung am Pin des Mikroprozessors auf logisch Null, so wird der gerade bearbeitete Befehl zu Ende ausgeführt. Danach wird der Inhalt des Prozessorstatusregisters und des Programmzählers auf den Stack gerettet und der Prozessor holt sich aus $FFFE/FFFF eine Adresse, zu der er verzweigt und dort ein Programm ausführt. Dies tut er solange, bis er auf den Befehl RTI (Return from Interrupt) stößt. Er holt jetzt den Programmzähler und das Prozessorstatusregister wieder vom Stack und arbeitet dort weiter, wo er vorher unterbrochen wurde.

Die spezielle Eigenschaft des IRQ ist, daß er auch unterbunden werden kann. Es gibt zu diesem Zweck im Prozessorstatusregister ein Bit (Interrupt Disable Bit), das ein IRQ-Signal nur dann zur Wirkung kommen läßt, wenn es auf Low steht. Die Befehle, um dieses Bit zu setzen oder zu löschen, sind SEI und CLI.

Auch der NMI ist ein Interrupt, der dafür sorgt, daß ein spezielles Interruptprogramm abgearbeitet wird, wenn der entsprechende Pin auf Low geht. Der Unterschied zum IRQ besteht darin, daß er nicht unterbunden werden kann. Aus diesem Grund wird er als NMI (Non Maskable Interrupt = 'nicht maskierbarer Interrupt") bezeichnet. Ein weiterer Unterschied ist der Sprungvektor für den NMI. Dieser steht nicht in $FFFE/FFFF, sondern in den Speicherstellen $FFFA/FFFB (L/H).

Der NMI erfüllt bei der 1541 keine Interruptfunktion. Sein Sprungvektor wird für den UI-Befehl verwendet.

Ein weiterer nicht maskierbarer Interrupt wäre noch das RESET-Signal des Prozessors, das grundsätzlich beim Einschalten von Computer oder Floppy auf Low geht und damit das RESET-Prograннn ab der in $FFFC/FFFD angegebenen Adresse ausführt. Dieses Signal läßt sich bei der 1541 durch den U:-Befehl nachvollziehen, der den gleichen Sprungvektor verwendet.

Jetzt aber zur wichtigsten Interruptart: dem IRQ. Die Floppy besitzt insgesamt zwei verschiedene Programme:

- das Hauptprogramm
- das Diskcontrollerprogramm

Das zweite Programm ist hierbei ein Interruptprogramm, das über das IRQ-Signal abgerufen wird.

Um diese Technik der Programmierung besser zu verstehen, werden wir uns als nächstes diese zwei Programme, jedes für sich, genauer ansehen.

### 5.3.3 Die Arbeit im Hauptprogramm

Das Hauptprogramm ist als das eigentliche Programm zu verstehen, das immer ausgeführt wird, sofern kein IRQ anliegt. Es analysiert die Befehle, die vom BC entgegengenommen werden und führt diese gegebenenfalls aus, sofern kein Syntaxfehler erkannt wurde.

Wenn ein Befehl ausgeführt werden soll, der den Laufwerksbetrieb erfordert, schickt das Hauptprogramm ein Kommando an den Diskcontroller und wartet, bis dieser die vollständige Ausführung zurückmeldet. Danach holt es die erhaltenen Daten ab und verarbeitet diese weiter. Wie diese Technik funktioniert, erfahren Sie in Kapitel 5.3.5.

Erfordert ein Befehl den Busbetrieb, so wird das über ein ATN-Flag angezeigt.

Das Hauptprogramm bedient nun solange zusätzlich auch noch den Bus, bis das ATN-Signal vom Computer wieder zurückgesetzt wird.

Wie das im einzelnen funktioniert, erfahren Sie in Kapitel 10, das sich mit der Funktionsweise des seriellen Busses beschäftigt.

### 5.3.4 Die Arbeit im Diskcontroller-Modus

Der Diskcontroller (DC) ist, wie schon erwähnt, der Programmteil und die Hardware der Floppy, der die Steuerung des Laufwerkes übernimmt. Hierzu zählen:

- Steuerung des Laufwerksmotors
- Steuerung des Steppermotors
- Steuerung der Lese- und Schreibvorgänge

Da der DC ein IRQ-Programm ist, wird er nur dann aufgerufen, wenn das entsprechende Signal am Mikroprozessor ausgelöst wird. Bei den vielfältigen Aufgaben dieses Programms ist es aber notwendig, daß dieser Aufruf regelmäßig erfolgt, damit die Steuerung effektiv sein kann.

Zum Zweck des regelmäßigen Aufrufs von Interruptprogrannnen sind in den vorher beschriebenen VIA 6522 Bausteinen Intervalltimer eingebaut. Diese Timer können auf bestimmte Zeiten eingestellt werden und laufen rückwärts auf Null. Sind sie dort angelangt, lösen Sie einen IRQ aus. Schaltet man diese Timer jetzt noch in den sogenannten 'Continuos Mode', erfolgt nach jedem Nulldurchlauf automatisch das erneute Starten mit dem vorher eingestellten Wert. Die Folge ist ein dauerndes Auslösen von IRQ-Signalen in einem bestimmten Rhythmus.

In der 1541 wird zu diesem Zweck der Timer 1 des DC VIA 6522 benutzt. Er löst ungefähr alle 14 ms einen IRQ aus und startet damit den DC.

### 5.3.5 Die Technik der Jobschleife

Dieses Kapitel wird sich jetzt mit dem wohl kompliziertesten Teil des DOS auseinandersetzen. Wenn Sie effektiv in das DOS eingreifen wollen, ist das Verständnis dieses Abschnitts unbedingt nötig.

Es handelt sich hier um das Problem der Verarbeitung von Befehlen innerhalb des DOS. Ich habe ja schon erwähnt, daß das eigentliche DOS aus drei verschiedenen Aufgabenbereichen besteht, die jeder von einem eigenen Programm abgedeckt werden. Zwei dieser Programme werden dabei interruptgesteuert aufgerufen.

Es stellt sich nun die Frage, wie diese Programme zusammenarbeiten. Wie können sie sich überhaupt synchronisieren? Immerhin wird das Hauptprogramm bei einem Interrupt praktisch abgeschaltet. Wie kann es seine Befehle und Informationen weitergeben, wenn es auf das IRQ-Programm keinen Einfluß hat?

Uns soll jetzt nur die Beziehung zwischen Hauptprogramm und DC interessieren; der serielle Bus wird in Kapitel 10 genauer unter die Lupe genommen.

Die Lösung des oben genannten Problems heißt: Jobschleife. Sie beruht auf folgendem Prinzip:

Alle Programme der Floppy befinden sich in einer Art "Ruhezustand", wenn kein Befehl anliegt. Dies gilt sowohl für das Hauptprogramm, das sich in einer Endlosschleife befindet, als auch für den DC.

Eine Schlüsselposition nimmt jetzt die Zeropage ein. Kommt ein Befehl vom Computer, wird als erstes der BC aktiv. Er wird durch ein ATN-Signal vom Computer gestartet und setzt eine bestimmte Speicherstelle auf einen bestimmten Wert. Diese Speicherstelle dient als Flag und wird laufend vom Hauptprogramm abgefragt. Ist jetzt der BC-Interrupt beendet, fährt das Hauptprogramm in seiner Endlosschleife fort und fragt besagte Speicherstelle ab. Findet es jetzt den Code, den der BC hineingeschrieben hat, "weiß" das Hauptprogramm, daß Daten anliegen. Es verläßt die Endlosschleife und holt die Daten vom Bus.

Erkennt das Programm einen Befehl, der zum Beispiel das Laden eines Blocks von der Diskette erfordert, speichert es ebenfalls einen Code in eine bestimmte Speicherstelle, die jetzt wiederum vom DC als Flag für einen anliegenden Befehl verwendet wird. Danach geht das Hauptprogramm wieder in eine Warteschleife und überprüft laufend eine Speicherstelle, die ihm mitteilt, ob der Block geladen wurde und zur Verfügung steht.

Wird jetzt ein IRQ von dem vorhin angesprochenen Timer ausgelöst, fragt der DC wie üblich die Speicherstelle ab, die ihm mitteilt, ob ein Befehl anliegt. Er wird jetzt den Befehl des Hauptprogramms finden, ausführen und danach wiederum dem Hauptprogramm mitteilen, daß der Befehl ausgeführt wurde

Diese Vorgänge laufen in der Floppy einige 100mal in der Sekunde ab.

Wie Sie sehen ist die Arbeitsweise eigentlich ganz einfach: jedes Programm wartet, bis ein Auftrag anliegt, indem es bestimmte Speicherstellen bei jedem Durchgang abfragt.

Jetzt ist Ihnen der Name Jobschleife vielleicht auch verständlich; Jedes Programm im DOS wartet in einer Schleife, bis ein Auftrag (Job) anliegt, der dann als Unterprogramm ausgeführt wird. Danach wird wieder in die Schleife zurückgekehrt.

Dieses Prinzip ist im Grunde leicht zu verstehen. Es erfordert jedoch enormen programmiertechnischen Aufwand bei der Realisierung, da mit einer Unmenge von Zeigern, Flags, Zwischenspeichern, usw. gearbeitet werden muß. Sie verstehen jetzt vielleicht auch, warum man keinen Eingriff in die Zeropage der Floppy wagen sollte, solange man sich nicht genau auskennt.

Wenn Sie sich das DOS-Listing im Anhang dieses Buches ansehen, so erschrecken Sie nicht. Die Programme sind um einiges komplizierter als im Computer; ich habe mir die Muhe gemacht, alles so ausführlich wie möglich zu kommentieren.

So, jetzt aber genug der grauen Theorie; wir wollen unsere ersten Gehversuche in der Programmierung der 1541 machen.

# 6
# Das Ausführen von Programmen im Pufferspeicher

# 6 Das Ausführen von Programmen im Pufferspeicher

Nachdem Sie bereits wissen, daß es beim DOS Befehle gibt, die es uns gestatten, Maschinenprogramme in den Pufferspeichern der Floppy ablaufen zu lassen, wollen wir uns nun genauer mit diesem Gebiet beschäftigen. In der Tat eröffnet uns diese Art der Programmiertechnik eine Fülle neuer Möglichkeiten in der Anwendung der 1541. Wir können uns eigene Betriebssysteme schreiben oder den Befehlssatz der Floppy erweitern. Wir können die Geschwindigkeit beim Diskettenzugriff erhöhen oder einen Kopierschutz auf Diskette bringen.

## *6.1 Aufruf von Unterprogrammen aus dem ROM*

Wie Sie sicher vermuten, können wir beim Ablauf unserer Maschinenprogramme im Floppyspeicher natürlich auch auf die Systemroutinen des DOS zugreifen. Dies erleichtert die Arbeit um einiges, da wir die meisten Programme, die wir für unsere 'Systemerweiterungen' benötigen, schon im DOS implementiert vorfinden.

Es sei an dieser Stelle auf den Anhang verwiesen. Hier finden Sie ein dokumentiertes Listing des DOS, um dessen Studium Sie kaum herumkommen werden, wenn Sie in den Programmablauf der Floppy eingreifen wollen.

## *6.2 Nutzung des DOS-Hauptprogramms*

Die Überschrift dieses Kapitels ist vielleicht etwas irreführend, denn es bleibt uns hier nur zu bemerken, daß Sie das Hauptprogramm des DOS praktisch überhaupt nicht nutzen können.

Der Grund besteht in der Tatsache, daß das Hauptprogramm in einer Endlosschleife arbeitet und von dort aus die einzelnen Unterprogramme mit JMP-Befehlen aufruft. Man kann aus den Unterprogrammen mit RTS nicht zurückkehren, sondern muß ebenfalls mit absoluten Sprüngen arbeiten, was uns die Arbeit natürlich sehr erschwert. Wollen wir nämlich beispielsweise die Routinen zur Bedienung des seriellen Bus aufrufen, gelingt uns das immer nur einmal, da diese Programme nach Ausführung direkt ins Hauptprogramm des DOS zurückspringen.

Das Wissen um diese Tatsachen ist für Sie besonders wichtig, da Sie hieraus ersehen können, daß Sie sich bei bestimmten Anwendungen ein gesamtes eigenes Programm schreiben müssen, obwohl dieses im ROM schon vorhanden ist. Erstellen Sie dabei auch ein eigenes Hauptprogramm, so ist zu beachten, daß Sie alle wichtigen Parameter der Floppy kontrollieren müssen, wie das auch das 'echte' Hauptprogramm macht, da sich ein 'Absturz' der Floppy sonst kaum verhindern läßt.

Schreiben Sie Programme in den Pufferspeicher der Floppy und lassen diese dann dort mit dem M-E-Befehl ausführen, ist noch zu beachten, daß diese Programme mit einem RTS abgeschlossen sein müssen, damit das DOS ordnungsgemäß weiterarbeiten kann.

Zur direkten Programmierung der Floppy noch ein Tip: wenn Sie schon länger Besitzer einer 1541 sind, sei Ihnen an dieser Stelle empfohlen, die Floppy aufzuschrauben und den Deckel abzunehmen, während Sie arbeiten. So haben Sie die gesamte Laufwerksmechanik im Blickfeld und können anhand der Kopfbewegung ständig kontrollieren, ob sich Ihr Programm im Floppyspeicher richtig verhält. Es wird Ihnen nämlich sicherlich öfter passieren, daß der Schreib-Lese-Kopf der 1541 scheinbar ohne Grund 'eigene Wege' geht.

Achtung: Beim Aufschrauben des Gehäuses ist sicherzustellen, daß der Netzstecker gezogen ist. Auch muß auf eventuell freigelegte spannungsführende Teile geachtet werden. Zu vermeiden ist außerdem das Anfassen der elektrischen Bauteile auf der Platine der Floppystation, da die ICs äußerst empfindlich gegenüber statischer Aufladung sind!

Das offene Gehäuse hat aber auch durchaus noch andere Vorteile (mehr Vorteile als Nachteile). So werden Sie zum Beispiel erkennen, daß sich der Schreib-Lese-Kopf der Floppystation auf der Unterseite einer eingelegten Diskette befindet.

Wie, Sie staunen? Dann überzeugen Sie sich am besten selbst von der Tatsache, daß die Seite der Diskette, auf der sich das Etikett befindet, nicht gleich der Seite ist, die auch tatsächlich beschrieben wird. In Wirklichkeit wird jede Diskette nämlich auf der Rückseite beschrieben. Darauf sollten Sie achten, wenn Sie Ihre wertvollen Programmdisketten das nächstemal ohne Hülle irgendwo hinlegen, obwohl das besser ganz zu unterlassen ist.

Ein anderer Vorteil des offenen Gehäuses liegt sicherlich darin, daß Sie einen guten Einblick bekommen, wie die Mechanik der Floppystation bei Diskettenoperationen arbeitet. Auch die Wartung der Floppy, wofür Sie in Kapitel 11 einige Ratschläge finden, wird so stark vereinfacht.

## *6.3 Nutzung der Jobschleife*

So, jetzt aber wieder zum Thema. Bei der Programmierung der Floppy sollten Sie sich natürlich mit der Jobschleife auskennen, da sie das Lesen und Schreiben überhaupt erst möglich macht. Wir sollten uns darüber im klaren sein, daß wir es hier mit einem Interruptprogramm zu tun haben, dessen Bedienung um einiges komplizierter ist, als sich das im Augenblick erahnen läßt.

Wenn Sie die RAM-Belegung der der Floppy in Anhang I betrachten, sehen Sie auf der ersten Seite die Bedeutung der Speicherstellen $00 bis $11. Dies sind die Jobspeicher der Floppy, also jene Speicherzellen, mit deren Hilfe die Kommunikation zwischen Hauptprogramm und Disk-Controller erfolgt.

Wenn Sie zum Beispiel in Speicherstelle $00 einen der aufgeführten Befehlscodes, zum Beispiel $E0, schreiben, erkennt der Diskcontroller beim nächsten Interrupt, daß ein Befehl anliegt und leitet die entsprechenden Schritte zu dessen Ausführung ein. In unserem Beispiel müßten Sie noch jeweils eine Track- und eine Sektornummer in $06/07 angeben.

Der DC schaltet jetzt den Drivemotor ein und positioniert den Kopf auf die angegebene Tracknummer; danach springt er zur Speicherstelle $0300 (Anfang des Puffer 0), da wir den Befehl zur Ausführung eines Maschinenprogramms in Puffer 0 gegeben haben.

Sie können jetzt theoretisch ein Programm im Puffer 0 stehen haben, das nicht im Normalmodus, und das ist zu beachten, sondern im DC-Modus, also als Interruptprogrannn, ausgeführt wird. Sie können ein solches Programm folglich auch nicht mit RTS abschließen, sondern müssen mit einem JMP wieder ins DC-Programm im ROM zurückspringen.

Zur Kontrolle hinterläßt der DC dem Hauptprogramm nach Ausführung eines Jobs immer eines Rückmeldung, die in der gleichen Speicherstelle übergeben wird, in der vorher der Jobcode stand. Eine Aufstellung dieser Rückmeldungen finden Sie in Anhang I.

Unten aufgeführt sehen Sie das Standardprogramm zur Übergabe eines Befehls an den DC und des Abwartens auf dessen Ausführung.

Unser Beispiel wird mit M-E aufgerufen und steht irgendwo im Pufferspeicher der Floppy. Es sollte jedoch nie in dem Pufferspeicher stehen, für den der Job aufgerufen wird. Dorthin wird nämlich entweder ein gelesener Block geschrieben (Jobcode $80), oder es steht dort ein Maschinenprogramm, das ausgeführt werden soll (Jobcode $E0 oder $D0).

Das kleine Programm führt ein sogenanntes 'BUMP' aus, das heißt, der Kopf wird zur Justierung an den Anschlag zurückgefahren:

```
         LDA #$12    Positionierung auf Track 18
         STA $06     Tracknummer für den Job
         LDA #$C0    Jobcode für 'Bump'
         STA $00     als Code für Puffer 0 übergeben
WEITER   LDA $00     Job schon ausgeführt ?
         BMI WEITER  Nein
         CMP #$02    auf fehlerfreie Durchführung prüfen
         BCC OK      kein Fehler aufgetreten
         JMP ERROR   Sprung in Fehlerroutine
OK       RTS         Ende
```

Nach diesem Standardbeispiel werden alle Routinen des DC behandelt, was eine sehr komfortable Bedienung ermöglicht, da fast alles übrige 'vollautomatisch' erledigt wird.

### 6.3.1 Die Steuerung des Laufwerkmotors

Wenn Sie ein Programm in der Floppy ablaufen lassen wollen, haben Sie in der Regel die Absicht, auch auf die Diskette zuzugreifen. Hierfür müssen Sie die Bedienung der Laufwerksmechanik beherrschen. Dazu zählt unter anderem der Laufwerksmotor.

Über die Bedienung des Laufwerksmotors gibt es im Grunde nicht viel zu sagen. Sie ist das Einfachste an der ganzen Floppyprogrammierung. Genauer gesagt brauchen Sie sich über das An- und Abschalten des Motors keine Gedanken zu machen, da es in der Regel automatisch erfolgt, sobald Sie ein entsprechendes Programm in der Floppy aufrufen, das über die Jobschleife arbeitet. Sie werden aus diesem Grund auch meistens den Jobcode $E0 benotigen, da dieser Ihr Programm im Speicher erst dann zur Ausführung bringt, sobald das Laufwerk bereit zum Schreiben und Lesen ist.

### 6.3.2 Die Steuerung des Steppermotors

Die Steuerung des Schrittmotors zur Tonkopfbewegung ist schon um einiges komplizierter. Sie erfolgt in der Regel nicht vollautomatisch, sondern erfordert einige Parameter in der Zeropage. Bei dem Jobcode $E0 sollten Sie zum Beispiel im Interruptprogramm wenigstens einmal einen 'Abstecher' in das Jobschleifenprogramm machen, um dem DC Gelegenheit zu geben, den Tonkopf in die richtige Position zu bringen.

Der Grund, warum ein einmaliges Starten der Jobschleife nicht genügt, um den Kopf zu positionieren, ist der, daß mehrere Interruptaufrufe zum Einstellen des Tonkopfes nötig sind. Der Ablauf sieht so aus:

Wenn der Befehl zum Kopfpositionieren kommt, werden alle Parameter für die zu 'fahrende' Strecke gesetzt, da sich der Tonkopf immer relativ von einer Position zur nächsten bewegt. Danach wird der Steppermotor initialisiert und die Jobschleife verlassen. Bei jedem weiteren IRQ werden die Zähler erniedrigt, und wenn sie auf Null stehen, wird der Steppermotor wieder in den Ruhezustand gebracht. Die ruckartige Bewegung des Steppermotors kommt daher, daß der Motor bei jedem IRQ eingeschaltet und am Schluß wieder ausgeschaltet wird; und das viele Male pro Sekunde.

Haben Sie sich schon einmal überlegt, daß Ihnen der Steppermotor vielleicht zu langsam ist? Erhöhen Sie einmal, wie angegeben, die Frequenz der IRQs. Sie werden sich wundern, wie der Kopf auf einmal über die Diskette 'rauscht':

```
OPEN 1,8,15
PRINT#1, "M-W" CHR$(5) CHR$(28) CHR$(3) CHR$(25) CHR$(0) CHR$(25)
CLOSE 1
```

Die Programmiertechnik beim Stellen des Tonkopfes bekommen Sie nur in den Griff, wenn Sie Erfahrung sammeln. Betrachten Sie die Beispiele im Anhang dieses Buches und scheuen Sie sich nicht davor, bei anderen Programmen zu 'spicken'. Sie ersparen sich damit normalerweise viel Arger und Mühe und unter Umständen auch mehrmals einen verstellten Schreib-Lese-Kopf, wenn die Floppy zu oft einen 'Bump' ausgeführt hat.

An dieser Stelle sollte ich vielleicht darauf hinweisen, daß Sie bei der Floppy im Gegensatz zum Computer mechanische Teile durch Programmsteuerung bewegen können. Mechanische Teile sind einem Verschleiß unterworfen, so daß Sie bei der Programmierung der Floppy mit viel Sorgfalt vorgehen sollten, um eine Überstrapazierung der Laufwerksmechanik zu vermeiden und Beschädigungen möglichst auszuschließenl

### 6.3.3 Lesen eines Blocks in den Pufferspeicher

Das Lesen eines Blocks in einen bestimmten Puffer der Floppy ist mit dem Vorrat an Kommandos, die uns zur Verfugung stehen, kein Problem. Aus dem Anhang können Sie ersehen, daß ein Jobcode existiert, der das Lesen eines Blocks veranlaßt. Bei der Anwendung dieses Codes ist darauf zu achten, daß die Diskette schon initialisiert ist. Beim Initialisieren übernimmt die Floppy nämlich die ID im Vorspann (Header) eines Datenblocks auf Track 18. Dieser Wert gilt als Standard und wird zum Vergleich für alle weiteren Lesevorgänge verwendet. Stimmt die ID nun nicht mit der im gefundenen Datenblockheader überein, so deklariert der DC den Block als nicht gefunden und übergibt eine Fehlermeldung.

Bevor Sie also das nächste Programm über den Kommandokanal in einen Puffer schreiben (nicht Puffer 0!!!), schicken Sie zuerst den I-Befehl zur Floppy:

```
         LDA #$00    ; Sektor 0
         STA $06     ; als Sektor für Puffer 0
         LDA #$12    ; Track 18
         STA $07     ; als Tracknummer für Puffer 0
         LDA #$80    ; Jobcode für 'Block lesen'
         STA $00     ; als Auftrag für Puffer 0
WEITER   LDA $00     ; Ende ?
         BMI WEITER  ; nein
         RTS         ; zurück zum Hauptprogramm
```

Dies ist der normale Weg zum Lesen eines Datenblocks. Wie Sie wissen, kann es auch vorkommen, daß zum Beispiel ein Block auf der Diskette zerstört wurde (absichtlich oder unabsichtlich). Hier können Sie mit dieser Routine nicht mehr viel ausrichten. Wie man modifizierte Routinen verwendet, darauf kommen wir im weiteren Verlauf dieses Buches noch zu sprechen.

### 6.3.4 Schreiben eines Blocks auf Diskette

Das Schreiben eines Datenblocks auf die Diskette mit Hilfe der Jobschleife erfolgt analog zum Lesen; nur muß hier der zu schreibende Puffer natürlich vorher mit den entsprechenden Daten gefüllt werden. Für das Initialisieren gilt das gleiche wie beim Lesen: die Diskette muß vor dem Schreibzugriff initialisiert worden sein, da der DC sonst den Header des zu schreibenden Blocks nicht finden kann.

Aus der Notwendigkeit des Initialisierens konnten Sie natürlich schon folgern, daß dem Schreibvorgang immer ein Lesevorgang vorausgeht. Das ist natürlich notwendig, da die Daten ja in einen bestimmten Sektor geschrieben werden sollen und dieser muß erstgesucht werden. Durch einen Schreibzugriff können deshalb auch keine Fehler im Header eines Datenblocks 'ausgemerzt' werden, da dieser nur einmal, nämlich beim Formatieren, geschrieben wird und danach nie mehr. Darüber werden wir später noch mehr erfahren.

# 7
# Die Aufzeichnung von Daten auf der Diskette

# 7 Die Aufzeichnung von Daten auf der Diskette

Wir haben bisher mit den Vorgängen des Schreibens und Lesens von Daten auf der Diskette hantiert und diese dabei als selbstverständlich hingenommen, ohne uns jemals darüber Gedanken gemacht zu haben, wie eigentlich Daten auf eine Diskette geschrieben werden. Was laufen in der Floppy für Vorgänge ab, wenn ein Block gelesen oder geschrieben wird?

Da diese Grundlagen für die Beherrschung der Floppyprogrammierung unbedingt notig sind, werden wir uns in diesem Kapitel ausschließlich damit befassen.

## *7.1 Aufbau eines Sektors auf der Diskette*

In Kapitel 3 haben wir uns bereits eine Diskette angesehen. Sie erfuhren, daß jede Diskette durch das Formatieren in Tracks und Sektoren unterteilt wird, und daß jeder Sektor einen Datenblock enthält. Jetzt wollen wir so einen Sektor genauer unter die Lupe nehmen.

Jeder Sektor der Diskette besteht aus dem Blockheader und dem eigentlichen Datenblock, der die abgespeicherten Informationen enthält. Der Blockheader ist dem Datenblock vorangestellt und enthält alle wichtigen Informationen, die zum Finden eines Blocks notwendig sind.

## 7.1.1 Aufbau des Blockheaders

| 8 | CKS | S | T | ID2 | ID1 | 0F | 0F | Lücke (10 GCR Bytes) |
|---|---|---|---|---|---|---|---|---|

Es bedeuten:

8 - $08 zeigt Beginn eines Blockheaders an
CKS - Checksumme über den Blockheader
S - Sektornummer des Blocks
T - Tracknummner des Blocks
ID2 - zweites ASCII-Zeichen der ID
ID1 - erstes ASCII-Zeichen der ID
0F - Lückenzeichen; hat keine Funktion

Bild 7.1 Aufbau eines Blockheaders

In Bild 7.1 sehen Sie eine schematische Darstellung eines Blockheaders. Er besteht in der Hauptsache aus Kontrollwerten, die dazu dienen, daß der DC einen bestimmten Block so schnell wie möglich ausfindigmachen kann.

Das Blockheaderkennzeichen 08 zeigt dem DC an, daß die nächsten Bytes, die gelesen werden, zu einem Blockheader gehören. Das ist zur Unterscheidung eines Blockheaders von einem Datenblock unbedingt nötig, da beide mit der gleichen speziellen Markierung auf der Diskette beginnen. Man nennt sie SYNC-Markierung. Sie wird in Kapitel 7.2.1 noch ausführlich erläutert werden.

Die Sektornummer zeigt dem DC an, ob er gerade auf den Block zugreifen will, der gewünscht wird. Anhand dieser Angabe sucht die Floppy nach den richtigen Sektoren bei allen Lese- und Schreibvorgängen.

Die Tracknunnner dient ebenfalls der Kontrolle, ob der richtige Sektor gefunden wurde. Sie dient außerdem der Kopfpositionierung, da der DC mit ihrer Hilfe feststellen muß, in welcher relativen Entfernung sich der nächste zu lesende Track befindet.

Die beiden ID-Kennzeichen wurden schon besprochen. Sie wissen jetzt auch, warum bei einem direkten Lese- oder Schreibzugriff (Kapitel 6.3) die Diskette immer vorher initialisiert werden muß. Durch das Initialisieren holt die Floppy den Standardwert der ID in zwei Speicherstellen (Zeropage $12/13). Bei jedem weiteren Zugriff auf einen Block wird dieser Standardwert mit den Werten im Blockheader verglichen, und nur bei Übereinstimmung ein Leseoder Schreibbefehl ausgeführt. Bei einem Direktzugriff mittels des Ul-Befehls würden Sie bei Nichtübereinstimmung einen "Disk ID Mismatch Error' erhalten.

Die beiden $OF-Werte haben bei der 1541 keine weitere Funktion, als zusätzlich zu den nächsten Bytes eine Lücke zu bilden, bevor die SYNC-Markierung des Datenblocks folgt.

## 7.1.2 Aufbau eines Datenblocks

| | | | |
|---|---|---|---|
| 7 | Datenblock mit 256 Bytes | CKS | Lücke |

Es bedeuten:

8 - Kennzeichen $07 für Datenblockbeginn
CKS - Checksumme über den Datenblock

Bild 7.2 Aufbau eines Datenblocks

Bild 7.2 zeigt den Aufbau eines Datenblocks. Dieser Teil folgt auf der Diskette immer hinter dem dazugehörigen Blockheader und bildet mit diesem die Gesamtheit eines Sektors.

Auch der Datenblock hat ein Kennzeichen, anhand dessen der DC feststellen kann, 'mit wem er es zu tun hat". Es hat hier den Wert $07 und folgt direkt nach der SYNC-Markierung.

Nach diesem Kennzeichen folgen die eigentlichen Daten, die wir als Benutzer auf die Diskette geschrieben haben. Sie haben die gleiche Reihenfolge, wie beim Schreibvorgang und werden üblicherweise von den beiden Linkbytes, die auf den nächsten Sektor zeigen, angeführt.

Hinter den 256 Datenbytes folgt, wie schon beim Blockheader, eine Prüfsumme, die zur Kontrolle auf Lesefehler dient.

Nach der Prüfsumme folgt wieder eine Lücke, die allerdings in ihrer Länge von Track zu Track variiert, da sie für eine gleichmäßige Verteilung der Sektoren auf Diskette sorgen soll.

## *7.2 Die Technik beim Schreiben auf Diskette*

Wir haben eben erfahren, wie Daten auf der Diskette abgelegt werden und sind dabei schon sehr genau auf die Details eingegangen. Mit den Kenntnissen über den beschrieben Blockaufbau dürfte es Ihnen nicht mehr schwer fallen, die Schreib- und Leseroutinen im DOS zu durchschauen. Sie werden allerdings noch bei ein paar Feinheiten stutzig werden. Um diese letzten Mängel auch noch zu beseitigen, wollen wir jetzt auf die grundlegenden Voraussetzungen eingehen, die zum Schreiben auf die Diskette erforderlich sind.

### 7.2.1 Die SYNC-Markierungen

Die SYNC-Signale oder -Markierungen wurden namentlich ja schon ein paarmal erwähnt; jetzt wollen wir sehen, was es mit dieser Einrichtung auf sich hat.

Können Sie sich noch an Kapitel 3.1 erinnern? Wir erfuhren dort, was es mit dem Formatieren eigentlich auf sich hat. Bei dieser Gelegenheit habe ich auch den Unterschied zwischen soft- und hardsektorierten Disketten erklärt.

Zur Wiederholung: bei hardsektorierten Disketten hat die Diskette eine Menge Indexlocher eingestanzt, die zur Orientierung beim Lesen und Schreiben dienen. Bei softsektorierten Disketten erwähnte ich dazu nur, daß sich diese ihre eigenen Markierungen beim Formatieren auf die Diskette schreiben.

Diese Markierungen, die ansonsten aus Löchern bestehen, sind die SYNC- oder Synchronmarkierungen. Eine SYNC-Markierung besteht aus mehreren $FF-Bytes hintereinander auf Diskette.

Aber, werden Sie sagen, wenn Sie nun in einem Datenblock lauter $FF-Bytes haben, dann bringt dieser ja den DC durcheinander, weil dieser die $FF-Bytes für eine SYNC-Markierung hält.

Daß dem natürlich nicht so ist und auch nicht sein kann erfahren sie im nächsten Kapitel.

Jetzt wollen wir uns mit den SYNC-Signalen noch etwas genauer beschäftigen.

Beim Formatieren wird eine Diskette folgendermaßen eingeteilt:

Wenn ein Sektor geschrieben werden soll, wird zuerst eine SYNC-Markierung auf die Diskette gebracht. Danach folgt die 8 als Kennzeichen des Blockheaders und dessen übrige Bytes mit der Lücke. Als nächstes wird wieder eine SYNC-Marke auf die Diskette gebracht und der eigentliche Datenblock, angeführt vom Wert 7, dahinter geschrieben. Nach der Lücke hinter dem Datenblock folgt die erste SYNC-Markierung des nächsten Blockheaders und so weiter. ..

In Bild 7.3 können Sie diese Organisation noch einmal grafisch dargestellt sehen.

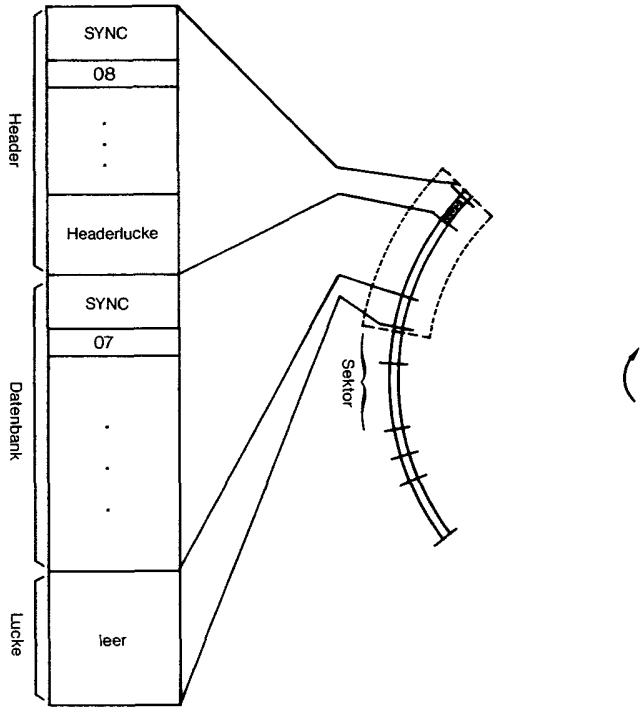



Bild 7.3 Schema einer Diskette mit SYNC-Marken

Das Besondere an den SYNC-Markierungen ist, daß sie nicht durch aktives Abfragen der Lese-/Schreibautomatik erkannt werden müssen, sondern daß sie anhand ihrer Bitfolge auf der Diskette automatisch ein Signal auslösen, das über CB2 des DC geleitet wird.

### 7.2.2 Die GCR-Codierung

Wenn Sie sich erinnern, haben wir im vorigen Kapitel erfahren, daß eine SYNC-Markierung durch aufeinanderfolgende $FF-Bytes auf Diskette geschrieben wird. Es stellte sich hier natürlich die Frage, was mit der gesamten Schreibautomatik passiert, wenn wir einen Block auf die Diskette schreiben wollen, der nur aus $FF-Bytes besteht. Nachdem wir die wichtige Funktion der SYNC-Markierungen kennen, müßte ein solcher Block ein Chaos auf der Diskette anrichten. Da ein solches Chaos aber nicht passiert, obwohl wir öfter mehrere $FF-Bytes hintereinander auf Diskette schreiben, scheint es hier eine Einrichtung zu geben, die die Verwechslung unterbindet.

Eine solche Einrichtung existiert in der Tat. Sie nennt sich GCR-Codierung (Group Code Recording).

Die Floppystation kennt also prinzipiell zwei Schreibarten auf der Diskette:

- das Schreiben von Markierungen
- das Schreiben von Daten

Wenn die Floppy beim Formatieren einen Track schreibt, werden die SYNC-Markierungen im 'Direktmodus' auf Diskette geschrieben, indem man einfach fünf $FF-Bytes zum Tonkopf schickt. Danach werden die Blockheader hergestellt. Diese werden in den GCR-Code umgewandelt und erst dann zum Tonkopf geschickt. Genauso verhält es sich mit den Datenblöcken, die ja alle einen bestimmten 'Leerinhalt' mitbekommen.

Wie die Umwandlung der Daten vom Binärcode in den GCR-Code erfolgt, wollen wir jetzt untersuchen.

Wie Sie wissen, besteht ein Hex-Byte aus acht Bits. Eine Byte im Hexadezimalsystem dargestellt besteht wiederum aus zwei Ziffern;

jede im Bereich von $0 bis $F. Wenn wir eine Hex-Zahl in die acht Bits aufspalten, so können wir erkennen, daß immer genau vier Bits für eine Ziffer zuständig sind. Diese Einheit von vier Bits nennt man Nibble, und zwar besteht eine Hex-Zahl aus einem nie-derwertigen (englisch: low) und einem hoherwertigen (englisch: high) Nibble.

An einem Beispiel sei dies verdeutlicht:

| Hex | (dezimal) | Binär | High-Nibble | Low-Nibble |
|---|---|---|---|---|
| $44 | (68) | 01000100 | 0100-—— | ——0100 |

Diese Zerlegung einer Zahl in ein Low- und ein High-Nibble ist für die GCR-Codierung unbedingt notwendig. Es ist nämlich beim Schreiben auf die Diskette erforderlich, daß nie mehr als zwei '0'-Bits oder neun '1'-Bits auf Diskette direkt hintereinander geschrieben werden. Warum das so ist, erfahren Sie später.

Da bei einer normalen Hex-Zahl eine solche unerlaubte Kombination ohne weiteres auftreten kann, müssen die Bytes codiert werden. Dazu wird die Zahl in zwei 4-Bit-Nibbles aufgespalten, und diese werden dann anhand der unten abgedruckten Tabelle in jeweils zwei 5-Bit-Nibbles umgewandelt.

| Hexadezimalwert | | Binärwert | 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 |

Bild 7.4 Tabelle für Binär-GCR-Konvertierung

Diese Umrechnung wollen wir mit einem Beispiel veranschaulichen:

Sie haben zwei Werte $45 und $E2. Diese Werte werden wir zunächst einmal in die entsprechenden Binärzahlen umwandeln:

```
$45  = %0100 0101   $E2  = %1110 0010
```

Wie Sie sehen, stehen die Nibbles jetzt schon da, um in die entsprechenden GCR-Äquivalente umgewandelt zu werden:

```
0100  0101  1110  0010   sehen im GCR-Code so aus:
01110 01111 11110 10010
```

Um aus diesen Bitkombinationen wieder Zahlen herzustellen, werden wieder jeweils 8 Bits zu einem Byte zusammengefaßt und weiterverarbeitet. In unserem Beispiel sähe das so aus:

| | | |
|---|---|---|
| 0111 0011 | 1111 1101 | 0010 ---- |
| $73 | $FD | $2- |

Wie Sie erkennen können, sind in unserem Fall aus vorher zwei Binärwerten jetzt zweieinhalb GCR-Bytes geworden. Diese Ungleichheit ergibt sich zwangsläufig aus der Tatsache, daß pro Byte immer zwei Bits hinzukommen, wenn in GCR umgewandelt wird. Bei der 1541 werden deshalb immer vier Werte zusammengenommen und codiert, da man dann die gerade Anzahl von fünf vollständigen Bytes erhält, wenn umgewandelt wird. Auch das wollen wir mit einem Beispiel verdeutlichen:

| | | | | |
|---|---|---|---|---|
| Hex: | $30 | $12 | $29 | $5A |
| Bin: | 0011 0000 | 0001 0010 | 0010 1001 | 0101 1010 |
| GCR: | 10011 01010 | 01011 10010 | 10010 11001 | 01111 11010 |
| | 1001 1010 | 1001 0111 | 0010 1001 | 0110 0101 | 1111 1010 |
| = | 9A | 97 | 29 | 65 | FA |

Dieses Beispiel zeigt die tatsächlichen Vorgänge bei der Codierung in der Floppy. Die Bytes werden immer in Viererblöcken codiert und anschließend auf die Diskette geschrieben,

Es wurde vorhin schon angesprochen, daß sich nie mehr als neun '1'-Bits und nie mehr als zwei '0'-Bits direkt hintereinander auf der Diskette befinden dürfen. Wenn Sie sich die codierten Werte ansehen, werden Sie feststellen, daß diese Voraussetzung hier erfüllt ist. Den Grund für die verbotenen zehn '1'-Bits können Sie sich höchstwahrscheinlich jetzt schon denken.

Es ist so, daß mehr als 9 gesetzte Bits ein SYNC-Signal darstellen und von der Schreib/Leseelektronik auch als ein solches erkannt würden, wenn eine derartige Kombination nicht ausgeschlossen werden würde.

## *7.3 Das Schreiben von Bits auf Diskette*

Im vorangegangenen Kapitel haben Sie erfahren, wie Daten in Wirklichkeit auf der Diskette abgelegt werden. Man stellt sich das ja immer recht einfach vor; Diskette rein, Daten zum Tonkopf und von dort auf die Diskette, und da haben sie normalerweise auch zu bleiben. Aber sehen Sie sich doch einmal eine Diskette an. Haben Sie sich nicht schon einmal darüber Gedanken gemacht, was man eigentlich alles auf kleinstem Raum unterbringt, wenn man eine Diskette beschreibt?

In diesem Kapitel soll die Funktionsweise der Lese/Schreibelektronik erklärt werden. Wie arbeitet ein Tonkopf überhaupt, und wie kann man Daten auf eine Magnetscheibe schreiben?

### 7.3.1 Funktionsweise eines Schreib/Lesekopfes

Wie Sie sicherlich wissen, beruhen die Schreib- und Lesevorgänge auf Magnetismus. Es gibt bestimmte Materialien, die wir als Magnete bezeichnen und die auf Eisen und Stahl eine 'anziehende' Wirkung haben.

Magnete haben einen Nord- und einen Südpol. Wenn Sie nun zum Beispiel einen Stabmagneten in der Mitte zerlegen würden, hätten Sie wieder zwei vollständige Magnete mit je einem Nord- und einem Südpol. Dieses Zerteilen ließe sich theoretisch bis in den molekularen Bereich fortsetzen. Irgendwann stoßen wir jedoch an eine Grenze. Wir haben die Größe der kleinsten Magnete erreicht, die es gibt. Diese 'Minimagnete' bezeichnet man als Elementarmagnete, da eine weitere Zerteilung nicht möglich ist, ohne deren Eigenschaften zu zerstören.

Jedes Metall, das es gibt, enthält nun solche Elementarmagnete, ohne daß es deswegen gleich magnetisch wirksam wäre. Der Grund liegt in der Tatsache, daß diese Elementarmagnete in einer ungeordneten Form vorliegen, also keine Kraft in eine bestimmte Richtung entfalten.

Könnte man jetzt diese (wir stellen sie uns als kleine Stabmagnete vor) Elementarmagnete in einer bestimmten Stellung ausrichten, würden alle Nordpole in die eine und alle Südpole in die andere Richtung zeigen und eine konzentrierte Wirkung hervorrufen. Es gibt Materialien, die haben diese Eigenschaft von Natur aus; eben das sind unsere Magnete.

Magnetismus hat aber unter anderem auch die Eigenschaft, **in** einer gewissen Wechselbeziehung zur Elektrizität zu stehen. So entwickelt eine Spule aus einem Metalldraht ein magnetisches Feld, sobald sie von Strom durchflossen wird. Wickelt man diese Spule um einen Eisenkern, richten sich in diesem, angeregt durch das erzeugte Magnetfeld, alle Elementarmagneten aus, und der Effekt verstärkt sich. Wir erhalten einen Elektromagneten.

Dieser Effekt funktioniert auch andersherum. Wenn wir einen Magneten zwischen den Windungen einer Spule bewegen, entsteht in ihr Strom; man sagt, es wird Strom induziert.

Diese eben beschriebenen Effekte nutzt man aus, wenn man Magnetdatenträger verwendet.

Disketten, Tonbänder und Magnetplatten sind mit speziellen Substanzen beschichtet, deren Elementarmagnete sich leicht ausrichten lassen. Man spricht bei diesem Vorgang vom Magnetisieren einer Substanz. Die Materialien haben hier den Vorteil, daß diese Magnetisierung lange aufrechterhalten bleibt und somit zur Datenspeicherung benutzt werden kann.

Um die Magnetpartikel auf der Diskette zu magnetisieren, verwendet man einen Tonkopf. Dieser Tonkopf ist im Prinzip nichts weiter als ein speziell geformter Elektromagnet.

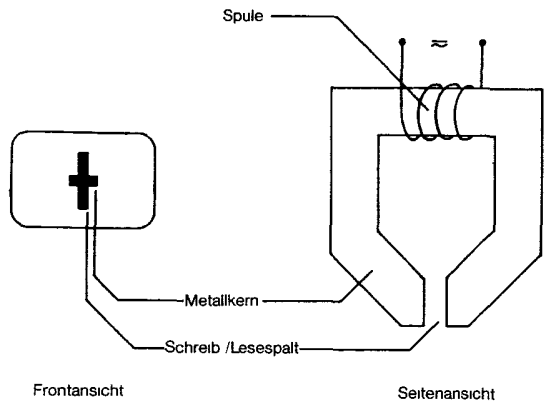



Bild 7.5 Schema eines Schreib/Lesekopfes

Wenn man durch die Spule des Tonkopfes Strom schickt, entsteht zwischen den beiden Enden im Tonkopfspalt ein Magnetfeld, das in seiner Richtung von der Stromrichtung in der Spule abhängt. Da der Tonkopf auf der Diskettenoberfläche aufliegt, wird das Magnetfeld in der Beschichtung der Diskette wirksam und magnetisiert so deren Oberfläche. Bild 7.6 zeigt, wie das dann im Schema aussieht.

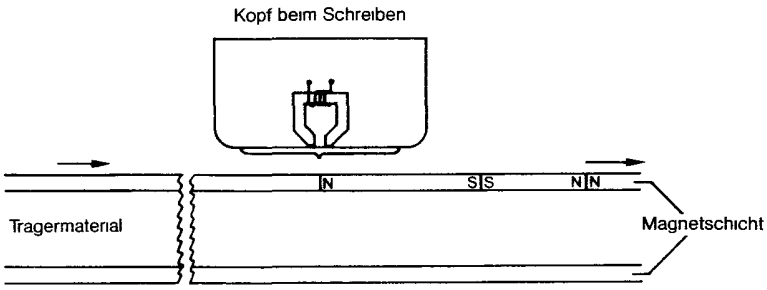



Bild 7.6 Aufzeichnung auf Diskette

## 7.3.2 Speichern von Daten auf Diskette

Die Voraussetzungen für das Schreiben auf Diskette hätten wir jetzt geschaffen. Jetzt bleibt uns nur noch, die Unterscheidung in '0'-Bits und in '1'-Bits vorzunehmen. Für die Datenspeicherung brauchen wir nur diese beiden Einheiten, aus denen sich die Bytes dann zusammensetzen.

Wie Sie aus Bild 7.6 entnehmen können, werden durch den Tonkopf eine Reihe von magnetisierten Zonen auf die Diskette geschrieben, die ihre Magnetrichtung mit der Richtung des Stromes in der Spule ändern. Die Definition von '0'- und '1'-Bits ist nun ganz einfach:

- jede Zone mit Magnetisierungswechsel ist '1'.
- jede Zone ohne Magnetisierungswechsel ist '0'.

Bild 7.7 zeigt noch einmal, was gemeint ist:

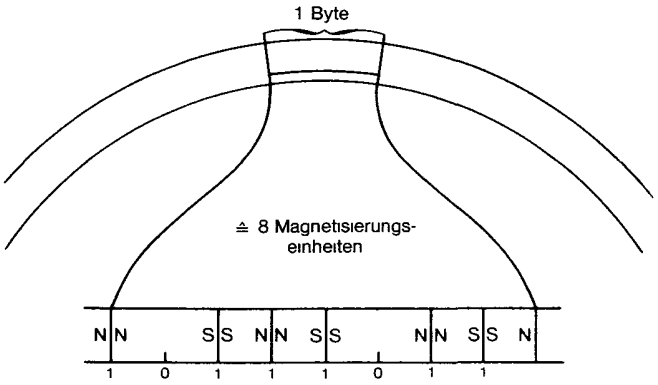



Bild 7.7 Daten auf Diskette

Schreibt die Floppy nun eine Reihe von Daten auf eine Diskette, fließt Strom durch den Tonkopf. Kommt ein '1'-Bit zum Schreiben an die Reihe, so wird die Stromrichtung umgekehrt; kommt ein '0'-Bit zum Schreiben, so wird die Stromrichtung beibehalten.

Es ist also für die Information nicht die Richtung der Magnetisierung maßgebend, sondern der Wechsel der Magnetisierung.

Die Schreibdichte der 1541 beträgt zwischen 4000 und 6000 Bits pro Zoll, was einer Information von bis zu einem 3/4 KByte entspricht. Der Unterschied ist auf die unterschiedliche Länge der inneren und äußeren Tracks zurückzuführen, was jeweils eine andere Winkelgeschwindigkeit des Diskettenabschnittes zur Folge hat.

## 7.3.3 Lesen der Daten von Magnetschichten

Das Schreiben von Daten auf eine Diskette reicht im Normalfall nicht aus. Man will die Informationen auch wieder von der Diskette lesen und weiterverarbeiten. Es wurde vorhin schon festgestellt, daß sich der Vorgang der Umwandlung von Strom in Magnetismus auch wieder umkehren läßt. In diesem Fall handelt es sich um den gleichen Vorgang wie beim Schreiben, nur in entgegengesetzter Reihenfolge.

Wenn die Magnetschicht der Diskette am Tonkopf vorbeibewegt wird, wird in der Spule des Tonkopfes ein Strom induziert. Dieser Strom wird verstärkt und in digitale Ausgangssignale zuruckverwandelt. Bleibt die Magnetisierung auf der Diskette gleich, ändert sich auch am Ausgangssignal nichts. Wir erhalten lauter '0'-Bits. Kehrt sich jetzt die Magnetisierung auf der Diskette um, kehrt sich auch der Stromfluß in der Spule des Tonkopfes um. Es wird ein Wechselspannungsimpuls empfangen, der ein '1'-Bit signalisiert.

Bild 7.8 zeigt eine Bitfolge, die anhand einer entsprechenden Magnetisierung auf einer Diskette entsteht.

Denken Sie immer daran: Nicht die Richtung des Magnetflusses bestimmt '1'-Bits sondern die Änderung der Flußrichtung!

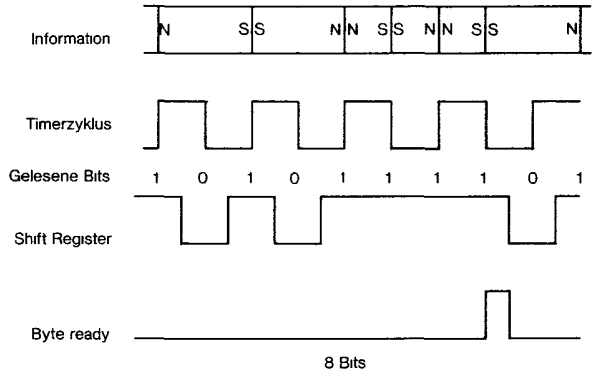



Bild 7.8 Lesen der Informationen von Diskette

## 7.3.4 Timing beim Lesen und Schreiben

Sicherlich werden Sie sich schon Ihre Gedanken gemacht haben: so eine komplizierte und störanfällige Angelegenheit; das kann ja nicht gutgehen?

Daß es dennoch gut geht, zeigt der tägliche und vielseitige Einsatz von Floppystationen. Naturlich ist die Schreibelektronik der 1541 noch gegen gewisse Störeinflüsse abgesichert; beispielsweise gegen Drehzahlschwankungen des Laufwerks. Wir werden gleich untersuchen, wie diese Sicherung funktioniert.

Da '0'-Bits durch 'gar nichts' auf Diskette gekennzeichnet werden, kann die Floppystation theoretisch gar nicht feststellen, ob jetzt nur eines oder mehrere dieser Bits gelesen wurden, da sie keinen Anhaltspunkt auf der Diskette hat, wann das vorherige Bit vorüber ist und wann das nächste beginnt.

Da so etwas in der Praxis aber trotzdem funktioniert, liegt an der speziellen Leseelektronik der 1541. Diese hat nämlich einen Timer (Zeitgeber) eingebaut, der durch die Magnetisierungsänderungen getriggert wird. Anders ausgedruckt: jedesmal, wenn ein '1'-Bit von der Diskette gelesen wird, wird der Timer neu gesetzt. Dem Timer entsprechend 'weiß' die Floppy, wann soviel Zeit verstrichen ist, daß das nächste Bit anliegt und gelesen werden kann.

Da dieser Timer aber eben durch jedes '1'-Bit neu gestellt wird, werden dadurch auch Laufwerksschwankungen ausgeglichen. Wie wir wissen, kommen ja nie mehr als zwei '0'-Bits hintereinander vor. Das wird durch die GCR-Codierung ausgeschlossen.

Jetzt haben wir also das Lesen der einzelnen Bits von der Diskette perfekt gemacht. Nun bleibt nur noch die Frage nach den Bytes. Der 6502 Prozessor arbeitet ja immer mit ganzen Bytes, die er vom Diskcontroller (VIA 6522) auch in der entsprechenden Form geliefert bekommt.

Das einzige Problem ist hier, daß der Prozessor wissen muß, wann ein Byte (acht Bits) fertig gelesen ist und zur Weiterverarbeitung anliegt.

Zu diesem Zweck gibt es eine spezielle Leitung, die direkt vom DC an ein Pin des 6502 gelegt ist. Wenn der Zähler vom Schieberegister des 6522 acht Bits durchgezählt hat, geht die eben erwähnte Leitung auf High. Als Folge davon wird im Prozessorstatusregister das Overflow-Flag gesetzt, woran die CPU erkennen kann, daß die Daten anliegen.

Die Signalleitung zum Prozessor bezeichnet man wegen ihrer Aufgabe als BYTE-READY-Leitung; das Signal heißt entsprechend BYTE READY und kann über einen bedingten Sprung (BVC, BVS) abgefragt werden.

Wichtig ist, beim Programmieren darauf zu achten, daß das gesetzte V-Flag nach jedem Gesetztwerden wieder 'von Hand' zurückgesetzt werden muß, da sonst ein 'dauerndes BYTE READY' die Folge ist.

Die Standardbefehlssfolge zum Empfangen eines Bytes vom Tonkopf der 1541 lautet also:

```
READY? BVC READY?  ; Warten auf BYTE RDY vom Diskcontroller
       CLV         ; BYTE RDY wieder löschen
       LDA $1C01   ; Byte von DC holen
       ...
```

Beim Schreiben auf die Diskette laufen die Vorgänge natürlich analog ab, da auch hier das Timing genau eingehalten werden muß:

```
       LDA Wert    ; Byte zum Schreiben auf Diskette
       STA $1C01   ; an DC übergeben
READY? BVC READY?  ; Schreibvorgang abwarten
       CLV         ; BYTE RDY wieder löschen
       ...
```

Wenn Sie dieses Kapitel gelesen haben, dürften eigentlich keine Schwierigkeiten mehr im Verständnis von Lese- und Schreibvorgängen auf magnetische Datenträger vorhanden sein.

Erstaunlich ist auch die Geschwindigkeit, mit der diese Vorgänge ablaufen. Wenn Sie auf niedrigster Ebene mit der Floppy arbeiten, bleibt Ihnen normalerweise zwischen dem Lesen des einen und des nächsten Bytes nur Zeit für einige wenige Maschinencodes, bevor der RDY-Pin schon wieder auf High geht.

# 8
# Wiederherstellen zerstörter Disketten

# 8 Wiederherstellen zerstörter Disketten

Wir haben uns lange mit dem Lernen der wichtigen Grundlagen der Floppyprogrammierung aufgehalten. Es ist jetzt an der Zeit, mit dem praktischen Teil dieses Buches zu beginnen. Ein sehr wichtiges Gebiet ist das Wiederherstellens zerstörter Disketten. Da es uns möglich ist, die Sicherheitseinrichtungen der Floppy zu umgehen und direkt auf die Diskette zuzugreifen, sind wir jetzt in der Lage, mit bestimmten Fehlern auf der Diskette fertigzuwerden und unter Umständen wichtige Daten zu retten.

## *8.1 Das Wiederherstellen gelöschter Files*

Diskette hineingelegt; ein S-Befehl zur Floppy geschickt und schon ist es womöglich passiert. Nein, das darf doch nicht wahr sein! Sie sehen sich noch einmal das Directory der anderen Diskette an, die Sie löschen wollten und ...

Genug der 'Vorgangsbeschreibung'. Wem wäre das noch nicht passiert. Eine kleine Unaufmerksamkeit und schon ist das verkehrte File gelöscht. Aber keine Angst, wenn Sie ein File mit dem SCRATCH-Befehl gelöscht haben, ist es eine Kleinigkeit, die Daten wieder zurückzuholen. Eine Schutzvorrichtung habe ich Ihnen ja schon im ersten Teil dieses Buches beschrieben. Genauso einfach wie das Aufbringen des SCRATCH-Schutzes ist auch das Wiederherstellen gelöschter Files.

Beim SCRATCH-Befehl löscht die Floppy das File nicht wirklich; sie stellt nur den Filetyp auf DELETED, beziehungsweise auf $00. Bei einem Programmfile steht er aber üblicherweise auf $82. Wir benutzen entweder einen Disk-Monitor oder schreiben uns ein Programm, das die Arbeit für uns übernimmt, den Filetyp des Eintrags im Directory wieder auf den ursprünglichen Wert zu stellen.

Ist dies geschehen, können Sie bei LOAD"$",8 erkennen, daß der Eintrag wieder an der ehemaligen Stelle im Directory erscheint.

Voraussetzung für das RESCRATCH ist allerdings, daß die Diskette inzwischen nicht neu beschrieben wurde!

Mit unserem RESCRATCH sind wir jedoch noch nicht fertig. Bei einem SCRATCH-Befehl wird nicht nur der Filetyp auf $00 gesetzt, es werden auch die Blöcke, die von diesem File belegt wurden, in der BAM wieder freigegeben. Auch dieses Problem läßt sich leicht lösen.

Sie wissen ja schon, daß der VALIDATE-Befehl bei jeder Ausführung wieder alle Blöcke auf der Diskette freigibt, die keinem File zugeordnet sind (Achtung bei B-W-Befehlen!). Da wir durch unseren neuen Eintrag im Directory die freigegebenen Blöcke des geSCRATCHten Files aber wieder einem Filenamen zugeordnet haben, reicht es, wenn wir nach der Wiederherstellung des Directory ein VALIDATE ausführen lassen. Dieser "fährt" dann unser File anhand des Eintrags im Directory ab und schreibt eine neue BAM, in der die Blöcke wieder als belegt gekennzeichnet werden.

## *8.2 Retten einer Diskette nach der Formatierung*

Eine Katastrophe größeren Ausmaßes können Sie durch das Neuformatieren einer Diskette auslösen. Hier ist das Wiederherstellen von Files nicht mehr so einfach wie beim SCRATCH-Befehl.

Wir müssen zwischen zwei verschiedenen Fällen unterscheiden:

- kurzes NEW
- langes NEW

### 8.2.1 Retten nach der kurzen Formatierung

Die 1541 kennt zwei unterschiedliche Arten der Formatierung; die kurze und die lange Formatierung. Jetzt wollen wir eine Diskette nach der kurzen Formatierung retten.

Eine Diskette kann nur dann kurz formatiert werden, wenn sie vorher bereits einmal lang formatiert worden ist. Das kurze Formatieren erreichen Sie dadurch, daß Sie beim NEW-Befehl kein Komma mit ID angeben.

In der Floppy wird jetzt die BAM neu erstellt (auf 664 Blocks free) und das gesamte Directory gelöscht, das heißt, mit Nullen ($00) aufgefüllt.

Die Schwierigkeiten, die wir beim Wiederherstellen einer solchen Diskette haben, bestehen in der Tatsache, daß im Directory keine Anhaltspunkte mehr zu finden sind, die angeben, wo wieviele Files mit welcher Länge auf Diskette geschrieben wurden. Die einzige Hilfe, die wir noch haben, ist der generelle Aufbau eines Files (bei dem noch nicht "herumgemurkst" wurde).

Wie Sie wissen, sind alle Files mit einem Endekennzeichen versehen, was besagt, daß der letzte Block jeder Datei ein $00 als Tracknummer des folgenden Blocks besitzt. Dieses Endekennzeichen ist unsere letzte Rettung.

Unsere Aufgabe besteht jetzt im Schreiben eines Programms, das die Diskette systematisch auf Blöcke, die mit diesem Endekennzeichen versehen sind, untersucht. Hat das Programm einen solchen Block gefunden, wird danach derjenige Block gesucht, der einen Zeiger auf den eben gefundenen enthält, also eigentlich der vorletzte Block im File war. Das geht solange, bis kein Block mehr gefunden wird, der in der Datei vor dem zuletzt gefundenen gestanden haben kann.

Ist man so mit einer Datei fertig, wird noch ein Eintrag im Directory hergestellt, der wieder den Zeiger auf den ersten Block der Datei (also den zuletzt gefundenen) enthält. Diese Dateieinträge im Directory sind natürlich jetzt nur sogenannte 'Dummies', die später von Hand wieder in 'richtige' Filenamen umgewandelt werden müssen.

Da ein solches Zurückverfolgen einer Datei ziemlich aufwendig ist, kann es ohne weiteres - je nach Programm und gelöschter Diskette - über eine Stunde dauern, bis das Directory wieder hergestellt ist. Auch hier dürfen Sie natürlich das VALIDATE am Ende nicht vergessen, sonst kann es passieren, daß Ihre Arbeit umsonst war.

### 8.2.2 Retten nach der langen Formatierung

Suchen Sie in diesem Artikel nach der Lösung für das Unmögliche? Haben Sie soeben eine Diskette lang formatiert, was auch akustisch durch das 'Rattern' am Anfang vernehmbar war?

Um Sie nicht lange auf die Folter zu spannen: es gibt keine Rettung Ihrer verlorenen Daten mehr!

Beim langen Formatieren wird nämlich die gesamte Diskette neu in Sektoren und Tracks unterteilt und damit komplett überschrieben.

Haben Sie also gerade eine Diskette im Laufwerk, bei der Sie nervös zum 'Nagelkauer' werden, weil Sie nicht genau wissen, ob es auch die richtige ist, die Sie soeben formatieren, dann sollten Sie keine Sekunde zögern. Denken Sie nicht mehr daran, was Sie in Büchern über das Herausnehmen von Disketten aus dem Laufwerk während des Betriebs gelesen haben. Mit jedem Bruchteil einer Sekunde, die Sie noch überlegen, gehen Ihnen vielleicht wertvolle Daten verloren. Jedes 'Klicken' des Laufwerks, das Sie hören, bedeutet, daß schon wieder eine ganze Spur (4 bis 5 1/4 KBytes) an Daten verloren ist. Ist die Diskette erst einmal fertig formatiert, gibt es keine Rettung mehr!!!

## *8.3 Lesen von fehlerhaften Files*

Wem wäre das noch nicht passiert: Sie laden ein Programm wie jedesmal, und auf einmal ist es geschehen. Die rote LED der Floppy blinkt und das Laufwerk hält mit einer Fehlermeldung an. Ihr bestes Programm scheint rettungslos verloren zu sein; zumal Softwarehersteller heutzutage keine Sicherungskopien mehr zulassen (aus Kopierschutzgründen versteht sich).

Wenn wir vom Fehler in einem Programmfile sprechen, müssen wir die zwei verschiedenen Arten von Fehlern unterscheiden, die uns das Leben sauer machen.

Tabelle 8.1 zeigt die Unterscheidung der Fehler in 'Soft-Errors' und 'Hard-Errors'.

| Fehlertyp | Nummer | Art des Fehlers |
|---|---|---|
| hard | 20 | READ ERROR |
| hard | 21 | READ ERROR |
| soft | 22 | READ ERROR |
| soft | 23 | READ ERROR |
| hard | 24 | READ ERROR |
| soft | 25 | WRITE ERROR |
| hard | 27 | READ ERROR |
| hard | 28 | WRITE ERROR |
| hard | 29 | DISK ID MISMATCH |

Tabelle 8.1 Liste aller Lesefehler von Diskette

### 8.3.1 Rettung bei Soft-Errors

Wie schon der Name vermuten läßt, scheint es sich bei den Soft-Errors um die weniger schlimmen zu handeln. In der Tat kann man bei diesen Fehlern die Daten meistens zum größten Teil retten.

Das etwas 'Dumme' an diesen Fehlern ist nur, daß sie um einiges seltener auftreten als die Hard-Errors.

Die Soft-Errors haben alle etwas gemeinsam: Wenn der Fehler auftritt, befinden sich die Daten normalerweise schon im DOS-Puffer der 1541, da die Soft-Errors in der Regel auf einen Verify-Error zurückzuführen sind, der nur durch den Vergleich des Speicherinhalts mit dem Disketteninhalt erkannt werden kann. Meistens handelt es sich um einen Prüfsummenfehler, der durch Abnutzung der Diskette entstanden sein kann.

Um sich bei Soft-Errors zu helfen, müssen Sie jeden Block des beschädigten Files einzeln anhand der Linker in den Floppyspeicher lesen und den Speicherinhalt dann in den Computer holen. Diese Technik funktioniert in der Regel perfekt, und Sie können die gelesenen Daten auf eine neue Diskette retten.

### 8.3.2 Rettung bei Hard-Errors

Die Hard-Errors machen im allgemeinen größere Probleme, wenn es um die Wiederherstellung von Files geht. Wenn Sie die Tabelle der Fehlermeldungen im Anhang dieses Buches durchgehen, werden Sie auch erkennen, warum.

Die Soft-Errors haben alle etwas mit dem Datenblock zu tun. Das heißt, der Blockheader ist in der Regel noch in Ordnung, so daß der entsprechende Sektor eindeutig identifiziert werden kann.

Bei den Hard-Errors ist es dem DC unmöglich, die, schon bei der Formatierung hergestellten, Blockheader zu erkennen oder einwandfrei zu lesen. In diesem Fall wird mit einem Fehler abgebrochen, weil der gesuchte Sektor nicht gefunden werden kann.

Die Ursachen von Hard-Errors sind entweder Defekte am Laufwerk oder an der Diskette. Es kann zum Beispiel passieren, daß der Tonkopf nicht richtig justiert ist. In diesem Fall kann die eingelegte Diskette nicht einwandfrei gelesen werden.

Besteht der Verdacht auf einen solchen Defekt, sollten Sie Ihre 1541 zum Kundendienst bringen. Der Verdacht ist dann gegeben, wenn die Lesefehler bei mehreren Disketten auftreten.

Die andere Möglichkeit ist die, daß die Diskette auf irgendeine Art beschädigt wurde. Das kann durch Eindringen von Fremdkörpern in die Umhüllung oder durch starke Magnetfelder geschehen.

Haben Sie einen DISK ID MISMATCH Fehler bekommen, ist dessen Beseitigung in der Regel sehr einfach. In diesem Fall müssen Sie die Diskette systematisch auf defekte Sektoren durchsuchen, nachdem Sie vorher eine vollständige Sicherungskopie angefertigt haben.

Es kommt jetzt nur darauf an, die Informationen noch soweit als möglich in den DOS-Speicher zu lesen. Wenn Sie sich im DOS-Listing mit den Leseroutinen des DC beschäftigen ($F510 - Blockheader lesen), werden Sie erkennen, daß der DC die ID in den Speicherstellen $12/13 mit der ID des Blockheaders vergleicht. Ihnen bleibt jetzt nur noch, eine eigene Blockleseroutine zu schreiben, die mit der des DOS bis auf die ID-Abfrage identisch ist. Die ID-Abfrage lassen Sie einfach heraus und lesen anschließend den gesamten Sektorinhalt in den Speicher (Sie können bei den Routinen des DOS 'spicken'). Dieser muß nach Beendigung nur noch vom Computer ausgelesen werden.

Bei den anderen Fehlern sieht die Sache für uns nicht so günstig aus. Der 21 READ ERROR ist zum Beispiel nicht der Fehler eines einzigen Datenblocks, sondern bezieht sich normalerweise auf eine ganze Spur der Diskette. Tritt diese Fehlermeldung auf, ist in der Regel nicht mehr viel zu retten.

Der Fehler mit der Nummer 20 kann sich auf einen einzelnen Block beziehen oder auf einen ganzen Track. Im ersten Fall kann man noch die übrigen Blöcke der Spur von Diskette retten; im zweiten Fall verhält es sich ähnlich wie bei Error 21; hier ist normalerweise nicht mehr viel zu holen.

Insgesamt sind allerdings solche Hard-Errors, die durch eine zerstörte Diskette hervorgerufen werden, äußerst selten. Der Fehler ist in der Regel beim Laufwerk zu suchen, oder die Diskette wies von Anfang an einen Herstellungsfehler auf.

## *8.4 Retten von physikalisch zerstörten Disketten*

Von den Defekten beim Diskettenbetrieb sind nicht nur die Fehler zu nennen, die durch die Abnutzung von Disketten entstehen. Es kann auch ohne weiteres passieren, daß Disketten auf irgendeine andere Art beschädigt werden, zum Beispiel, weil sie geknickt werden oder weil ein scharfer Gegenstand die Oberfläche zerstört hat.

Gehen wir davon aus, daß eine Diskette nicht mehr in der Ummantelung lauffähig ist, sei es, daß sie klemmt, oder daß die Hülle geknickt wurde. In diesem Fall gibt es nur eine Möglichkeit, wenn Sie die Daten noch retten wollen. Diese Methode eignet sich nur für den NOTFALL, und Sie müssen dabei mit äußerster Sorgfalt vorgehen, damit nichts beschädigt wird.

Schieben Sie die Diskette in der Hülle möglichst weit auf eine Seite. Danach nehmen Sie ein scharfes Messer und schneiden die Hülle am entgegengesetzten Ende am Rand auf. Jetzt können Sie die Magnetscheibe vorsichtig aus der Hülle ziehen. Achten Sie dabei darauf, daß Sie mit den Fingern nicht auf die Magnetoberfläche kommen, und merken Sie sich die Seiten der Diskette (die beschriebene Seite ist dem Etikett abgewandt).

Jetzt kommt der schwierige Teil des Unternehmens: Sie müssen die Diskette in das Laufwerk einlegen. Dazu muß das Gehäuse der Flop-py abgenommen sein, damit Sie von oben den Diskettenauswerfer zurückschieben können (er würde sonst die Scheibe zerstören). Legen Sie die Scheibe jetzt ein, wobei Sie darauf achten, daß die Ränder der Diskette in den Führungsrillen des Laufwerks liegen. Beim Betrieb der Floppy müßte die Diskette einwandfrei zu lesen sein, wenn nicht noch andere Defekte vorliegen.

Vorsicht; Beim Öffnen der Klappe am Laufwerk schnellt im Inneren der Diskettenauswerfer nach vorne, um die Diskette herauszuschieben. Er muß festgehalten werden, da er die dünne Magnetscheibe zerstören würde!

Ist es nicht die Hülle, die den Diskettenbetrieb stört, sondern ist die Oberfläche der Magnetscheibe beschädigt, sieht die Sache schon um einiges kritischer aus. Hier hilft nur das systematische Absuchen der Diskette auf Fehler, das Retten der 'gesunden' und das 'Behandeln' der fehlerhaften Sektoren (siehe 8.3).

# 9
# Funktionsweise von Softwareschutz auf Disketten

# 9 Funktionsweise des Softwareschutzes auf Disketten

Das wohl interessanteste Gebiet der Nutzung der DOS-Manipulation ist sicherlich der Softwareschutz. Hier geht es darum, eine Diskette möglichst sicher gegen das Kopieren mit 'Spezialprogrammen' zu machen. Die Vielfalt der Möglichkeiten, die das DOS der 1541 dafür bietet, ist fast nicht abzusehen. Wir wollen uns in diesem Kapitel mit den bekanntesten Methoden auseinandersetzen.

## *9.1 Schreiben von definierten Fehlern auf Diskette*

Die wohl älteste Methode des Softwareschutzes auf einer ganzen Diskette ist das Aufbringen von Fehlern. Es wird hierbei zum Beispiel ein Track gezielt von SYNC-Markierungen befreit. Versucht der DC nun diese Spur zu lesen, findet er natürlich keine SYNC-Markierung und steigt mit einer Fehlermeldung aus. In diesem Fall bekämen wir die Meldung 21 READ ERROR, die das Fehlen der SYNC-Signale anzeigt. Im Schutzprogramm, das ebenfalls auf dieser Diskette steht, wird diese Fehlermeldung gezielt abgefragt. Stimmt sie nicht mit der vorgegebenen überein, tritt der Schutz in Kraft.

Die Wirkung dieses Schutzes war anfänglich recht beträchtlich, da alle Kopierprogramme bei einem Fehler mit dem Kopieren aufgehört haben und den entsprechenden Block übergingen. Die Folge war, daß die Kopie keinen solchen Fehler enthielt.

Es dauerte aber gar nicht lange, da waren die 'Freaks' solchen Methoden gewachsen. Es tauchten die ersten Kopierprogramme auf, die in der Lage waren, viele der Fehler einfach mitzukopieren. Damit war der Schutz auf einfache Weise unwirksam geworden.

In dieser Situation schieden sich die Geister der Autoren von Programmschutzmethoden. Es kamen jetzt viele neue Ideen auf, die es den Raubkopierern so schwer wie möglich machen sollten.

Eine dieser Möglichkeiten beruht auf der Tatsache, daß die Kopierprogramme, die Fehler 'mitkopieren', diese in der Regel nur simulieren; also nicht wirklich kopieren.

Das nutzten die Softwarehersteller aus und komplizierten die Fehlerbehandlung, indem sie den fehlerhaften Blöcken Inhalte mitgaben. Es genügt also jetzt nicht mehr, die Fehler auf die Kopie zu übertragen. Entscheidend sind jetzt die Inhalte dieser Blöcke. Ein paar Programme, die Fehler auf Diskette erzeugen, finden Sie im Anhang dieses Buchs.

## *9.2 Verändern der Reihenfolge der Sektoren*

Die 'Fehlermethode' bei den Kopierschutzprogrammen hat ein paar Nachteile. Erstens schont sie die Floppystation nicht gerade, da die 1541 bei fast jedem Fehler zu 'Rattern' anfängt, um den Kopf neu zu positionieren, und zweitens wird anhand der Reaktion der Floppystation sehr deutlich, wo der Schutzmechanismus zu suchen ist.

Eine viel raffiniertere Methode des Programmschutzes besteht in der Möglichkeit, die Sektoren auf einer Diskette in ihrer Reihenfolge zu vertauschen. Normalerweise stehen diese in chronologischer Reihenfolge von 0 bis beispielsweise 20 auf einer Spur. Bei der Vertauschung wird die Reihenfolge der Sektoren willkürlich durcheinandergebracht und im Schutzprogramm abgefragt. Dieser Mechanismus wird während des Formatierens aufgebracht und ist normalerweise nicht 'spürbar'. Im Anhang finden Sie unter anderem ein Programm, das vielfältige Diskmanipulationen zuläßt und ein anderes, mit dem Sie die Reihenfolge von Sektoren auf der Diskette abtasten können.

## 9.3 Verändern der Abstände zwischen den Sektoren

Noch eine Schutzmethode. Diesmal eine der raffiniertesten, die derzeit existiert.

Wie Sie wissen, werden beim Formatieren alle Sektoren auf die Diskette geschrieben. Zwischen jedem Blockheader und dem dazugehörigen Datenblock existiert dabei eine Lücke, die eine festdefinierte Länge hat und somit nicht manipulierbar ist.

Eine andere Lücke ist jedoch ohne weiteres manipulierbar. Es handelt sich um die Abstände zwischen den einzelnen Sektoren. Für diese gilt nur ein Maßstab: die Lücke zwischen zwei Sektoren muß mindestens vier Bytes lang sein. Wenn auf einem Track jetzt zum Beispiel alle Lücken im Durchschnitt 10 Bytes lang sind, ist es ein leichtes, die eine Lücke auf 15 Bytes auszuweiten und die nächste dafür auf 5 Bytes schrumpfen zu lassen.

Im Schutzprogramm wird nun die Länge einer bestimmten Lücke gemessen und anhand derer auf das Vorliegen der Originaldiskette geprüft. Kopierprogramme sind nämlich normalerweise nicht in der Lage, auch noch die Verschiebung von Sektoren zu berücksichtigen.

Auch diese Schutzmethode hat den Vorteil, daß sie praktisch nicht spürbar ist und somit bei entsprechend verstecktem Schutzprogramm der Schutzmechanismus nur schwer zu erkennen ist. Liegt er erst einmal offen, ist es leicht, ein Programm zu schreiben, das Disketten in der benötigten Weise präpariert.

## 9.4 Das Arbeiten mit ungültigen Spuren

Wie Sie wissen, existieren auf einer Diskette, die mit der 1541 formatiert wurde, 35 Spuren mit der entsprechenden Numerierung von 1 bis 35. Die Einhaltung dieser Grenzen wird vom DOS vorgenommen, das spezielle Sicherheitseinrichtungen besitzt, die automatisch eine Fehlermeldung erzeugen, sobald wir versuchen, beispielsweise mit BLOCK-WRITE, eine Spur außerhalb dieses Bereichs zu beschreiben (ILLEGAL TKACK OR SEKTOR).

Auf der Ebene der DOS-Manipulation, auf der wir arbeiten, existieren allerdings keine derartigen Schutzeinrichtungen mehr. Es wurde schon darauf hingewiesen, daß man deshalb sehr sorgfältig beim Programmieren sein sollte. Wenn wir dem DC auf dieser Ebene den Befehl geben, den Kopf auf Track 50 zu positionieren, fuhrt er ihn auch aus, beziehungsweise er versucht es zumindest.

Nun ergibt es sich aber, daß die 1541 in der Lage ist, ein wenig mehr als nur 35 Spuren zu beschreiben. Auch die meisten Disketten vertragen noch eine gewisse Erweiterung dieses Bereichs bei voller Datensicherheit.

Was liegt also näher, als den Bereich über Spur 35 hinaus noch zu beschreiben und auf diese Weise einen Schutz zu konstruieren, der von den 'normalen' Kopierprogrammen nicht mehr erreicht wird.

In der Tat ist das ohne weiteres möglich. Die 1541 läßt eine Kopfpositionierung bis über Track 40 zu, bevor sich der Kopf am oberen Anschlagpunkt befindet.

Schutzeinrichtungen dieser Art enthalten entweder einen Code oder einen wichtigen Programmteil auf irgendeiner Spur oberhalb von Spur 35. Für uns ist es natürlich mittlerweise eine Kleinigkeit, auf einen Track oberhalb 35 zuzugreifen und einen derartigen Schutz selbst zu konstruieren. Versuchen Sie es doch einmall

# 10
# Der serielle Bus der 1541

# 10 Der serielle Bus der 1541

Nachdem wir uns mit den internen Vorgängen der Floppy schon hinreichend vertraut gemacht haben, soll nun auch ein weiteres wichtiges Glied im Floppybetrieb nicht unerwähnt bleiben. Es handelt sich um die Verbindung zwischen Floppy und Computer, mit deren Hilfe der Datenaustausch überhaupt erst erfolgen kann.

In der 'Computersprache' wird die Gesamtheit von Datenübertragungsleitungen immer als BUS bezeichnet. Hierbei kann man zwei Busarten unterscheiden: den seriellen und den parallelen Bus.

Innerhalb eines Computers existieren in der Regel nur der sogenannte Adreßbus und der Datenbus. Bei einem 6502-Prozessorsystem ist der Datenbus eine 8-Bit-parallele und der Adreßbus eine 16-Bit-parallele Übertragungsleitung. Bei diesen Bussytemen werden die Informationen also immer zu zwei oder mehreren Bits gleichzeitig (parallel) übertragen.

Anders der serielle Bus. Ein 8-Bit-Parallelbus benötigt nur einen einzigen Buszyklus, um ein ganzes Byte (8 Bits) zu übertragen. Da der serielle Bus jedoch alle Bits nacheinander (seriell) überträgt, benötigt dieser für ein Byte ganze 8 Buszyklen, da jedes Bit für sich über den Bus gesendet wird.

Jetzt werden Sie sich fragen, warum denn dann überhaupt der serielle Bus zum Einsatz kommt. Er ist ja in jedem Fall langsamer als der parallele Bus.

Die Antwort ist ganz einfach: der serielle Bus benötigt weniger Hardwareaufwand zu seinem Betrieb. Sehen Sie sich doch einmal das Kabel an, das die 1541 und den Computer miteinander verbindet. Es handelt sich um ein billiges Kabel mit DIN-Steckern, das für ein paar Mark in jedem Elektrogeschäft zu haben ist. Anders die Kabel für den parallelen Bus. Diese sind meistens mit Goldkontakten bestückt und müssen in der Regel über eine bessere Abschirmung verfügen, da die Übertragungsrate sehr viel hoher liegt. Ein solches Kabel ist deshalb nur in Spezialläden zu haben und kostet meistens über 100 Mark.

## *10.1 Die Arbeitsweise des seriellen Bus*

Auf die hardwaremäßige Funktionsweise des seriellen Bus wollen wir nicht weiter eingehen. Uns soll jetzt nur die Arbeitsweise interessieren. Wie wird der Busbetrieb 'abgewickelt'?

Dazu erst einmal ein paar Grundvoraussetzungen für die einwandfreie Funktion des seriellen Bus.

Wenn wir uns bisher mit logischen Zuständen beschäftigten, hatten wir uns an eine Eigenschaft gewohnt. Ein Bit ist dann gesetzt, wenn es auf logisch '1' steht und nicht gesetzt, also inaktiv, wenn es auf logisch '0', also auf Low steht.

Beim Busbetrieb ist diese Regelung nicht mehr vorhanden. Hier arbeitet man genau umgekehrt (Low-aktiver Betrieb). Eine Leitung ist dann aktiv, wenn ihr Zustand dem logischen '0' entspricht; und sie ist inaktiv, wenn sie auf High (logisch '1') steht. Diese Regelung ist für den einwandfreien Betrieb unbedingt notwendig; auf die Gründe soll jedoch jetzt nicht weiter eingegangen werden.

Eine zweite Voraussetzung für einwandfreien Busbetrieb ist die Anwesenheit eines 'Controllers'. Das heißt nichts anderes, als daß ein Gerät die Priorität am Bus besitzt und die Abläufe bei der Datenübertragung regelt. In unserem Fall ist das der Computer. Wird ein weiterer Controller am Bus angeschlossen, kommt der Betrieb durcheinander und es gibt ein Chaos.

Jetzt noch ein paar Worte zur Funktion der drei Signalleitungen, die beim seriellen Bus von Commodore verwendet werden.

Die ATN-(Attention)-Leitung kann nur vom Computer beinflußt werden. Wird sie nach Masse (auf Low) gezogen, so bereiten sich sämtliche angeschlossenen Peripheriegeräte auf den Empfang vor.

Die CLK-(Clock)-Leitung dient dem Timing des Busbetriebs. Sie signalisiert dem Empfänger, daß das nächste Bit gesendet wird.

Die DATA-Leitung schließlich ist die eigentliche Übertragungsleitung. Über sie werden alle Informationen gesendet.

Nun zum Ablauf beim Betrieb des seriellen Bus. Es gibt hier zwei verschiedene Fälle. Entweder der Controller (Computer) möchte Daten über den Bus senden oder er möchte Daten empfangen.

Nehmen wir einmal an, wir wollen ein Programm von der Diskette laden:

Der Computer zieht die ATN-Leitung auf Low, und sofort gehen alle Peripheriegeräte in den empfangsbereiten Zustand. Als nächstes gibt der Computer nun die Sekundäradresse oder Gerätenummer auf den Bus. In unserem Fall ist dies die Nummer 8. Das ist die Kennziffer für die Floppystation. Bis auf das angesprochene ziehen sich jetzt alle Geräte wieder vom Busbetrieb zurück, um die Übertragung nicht zu stören. Jetzt kommen der Filename und alle wichtigen Fileparameter auf den Bus. Die Floppy beginnt zu arbeiten und öffnet das File anhand des LOAD-Befehls für den Lesebetrieb. Jetzt kann der Computer in beliebigen Zeitabständen Daten von der Floppy empfangen. Er sendet dazu, wie oben, das ATN-Kommando, gibt danach die Gerätenummer an und sendet jetzt den Befehl, der der Floppy anzeigen soll, ob Daten gelesen oder geschrieben werden sollen. Wir wollen Daten lesen und schicken deshalb das TALK-Kommando über den Bus. Jetzt stellt die Floppy so lange Daten bereit, bis vom Computer das UNTALK-Signal kommt.

Da der Computer wissen muß, wann er das vollständige Programm geladen hat, muß ihm die Floppy mitteilen, wenn das letzte Byte des Files gelesen wurde. In diesem Fall geht ein EOI (End Of Information) zum Computer und dieser sendet als Reaktion darauf wiederum ein UNTALK, schließt das File auf dem Bus und meldet sich mit READY zurück.

Wichtig bei der Übertragung ist die Tatsache, daß vor jedem Übertragungsbeginn neu festgelegt werden muß, ob gelesen (TALK/UNTALK) oder geschrieben (LISTEN/UNLISTEN) werden soll. Auch ist es erforderlich, daß jedesmal die Gerätenummer wieder neu mitangegeben wird. Diese zwingenden Maßnahmen erlauben es nämlich andererseits dem Computer, gleichzeitig mehrere Files auf dem Bus offenzuhalten, um völlig 'gemischt' auf die einzelnen Geräte zugreifen zu können.

Wichtig ist es außerdem zu wissen, daß bestimmte Geräte nur Daten senden beziehungsweise empfangen (Drucker, Plotter, Bildschirm), und andere wiederum sowohl senden als auch empfangen können (Floppystationen, Festplattenspeicher, Kassettenrekorder).

Es ist kaum sinnvoll, an einen Drucker ein TALK-Kommando zu schicken, da er ein reines Ausgabegerät darstellt.

## *10.2 Spooling von Diskette*

Nach soviel Theorie über den Busbetrieb wollen wir uns wieder der Praxis zuwenden. Wir haben eben etwas über die Arbeitsweise des seriellen Busses erfahren. Am Schluß dieser Beschreibung wurde auch etwas über die Eigenschaften bestimmter Peripheriegeräte gesagt; nämlich, daß eine Floppystation Daten senden und auch empfangen, ein Drucker hingegen nur Daten empfangen kann.

Haben Sie sich schon überlegt, was eigentlich passiert, wenn Sie ein Programmlisting auf dem Drucker ausgeben?

Sie laden das Programm erst von der Floppy in den Computer, und anschließend schicken Sie es vom Computer an den Drucker weiter. Davon einmal abgesehen, daß dieses System 'eigentlich' sehr umständlich ist, kostet es auch noch Ihre wertvolle Zeit, da der Computer für die Dauer des Ausdrucks blockiert ist.

Wir haben vorhin etwas vom Controller gehört und daß der Busbetrieb nur funktioniert, wenn der Controller die Prioritäten regelt. Nun, in unserem Fall wollen wir einmal eine Ausnahme dieser Regelung kennenlernen: das Spooling.

Leider funktioniert das im folgenden Gesagte nicht mit allen Druckern. Sie müssen es also von Fall zu Fall ausprobieren.

Das Prinzip des Spooling ist folgendes; Wir holen uns das zu druckende Programm in den Computer. Anschließend eröffnen wir in der Floppy ein List-File und leiten den LIST-Befehl auf Diskette (siehe Kapitel 2.4). Dies funktioniert genauso wie beim Drucker, nur daß jetzt ein Filename angegeben werden muß:

```
OPEN 1,8,2,"filename,U,W"
CMD 1
LIST
CLOSE 1
```

Das Programmlisting steht jetzt so auf der Diskette wie es später auf dem Papier aussehen wird.

Jetzt senden wir einfach ein LISTEN zum Drucker, worauf sich dieser auf den Empfang von Daten und deren Ausdruck vorbereitet. Danach erfolgt ein TALK an die Floppy, nachdem wir zuvor das List-File wieder zum Lesen geöffnet haben.

Da der Computer nichts mehr zu tun hat, meldet er sich mit READY. Jetzt passiert etwas Seltsames: die Floppy beginnt zu laufen und der Drucker listet das gesamte Programm aus, obwohl der Computer nachweislich keinen Einfluß mehr auf den Betrieb der beiden Geräte hat.

Warum das so ist, ist klar. Der Drucker wurde auf Empfang geschaltet. Woher die Daten kommen, die er empfängt, ist ihm 'egal'. In diesem Fall erhält er sie von der 1541, die wir zuvor mit dem TALK-Kommando zum Senden aufgefordert hatten. Da also alle Regeln des Busbetriebs eingehalten wurden, kann die Übertragung störungsfrei erfolgen. Die Bedingung ist natürlich, daß sich der Computer in dieser Zeit 'still' verhält.

Das kurze Listing des Spooling-Programms für den Commodore 64 ist in Bild 10.1 dargestellt.

```
033C   JSR $0079    letztes Zeichen holen
033F   BEQ $0384    verzweige, wenn kein Zeichen
0341   JSR $FFE7    CLALL; alle Kanäle schließen
0344   LDY #$00
0346   LDA $03A5,Y  'SPOOLING' ausgeben
0349   BEQ $0351
034B   JSR $FFD2
034E   INY
034F   BNE $0346
0351   JSR $E254    Filenamen holen
0354   JSR $F5C1    Filenamen hinter Text auf Bildschirm
0357   LDX $B7      Länge des Filenamens
```

```
0359   BEQ $03C1    verzweige, wenn kein Filename
035B   LDA #$01
035D   LDX #$08
035F   LDY #$02
0361   JSR $FFBA    Fileparameter für Floppy setzen
0364   JSR $FFC0    File öffnen
0367   LDA #$04
0369   JSR $FFB1    LISTEN zum Drucker senden
036C   JSR $EDBE    Sekundäradresse zum Drucker senden
036F   LDX #$01
0371   JSR $FFC6    Eingabegerät setzen
0374   JSR $EDBE    Sekundäradresse zur Floppy senden
0377   JSR $EE85    Busbetrieb initialisieren
037A   JSR $EE97
037D   LDA #$00     Geräte zurücksetzen
037F   STA $99
0381   STA $98
0383   RTS          Ende; Spooling startet
0384   LDA #$01
0386   STA $98
0388   JSR $FFAE    UNLISTEN zum Drucker senden
038B   JSR $FFAB    UNTALK zur Floppy senden
038E   LDA #$01
0390   JSR $FFC3    File auf Floppy schließen
0393   LDY #$00
0395   LDA $03AF,Y  'END OF SPOOLING' ausgeben
0398   BEQ $03A0
039A   JSR $FFD2
039D   INY
039E   BNE $0395
03A0   LDA #$00     mit READY zurück ins BASIC
03A2   JMP $A474
03A5   53 50 4F 4F 4C 49 4E 47  'SPOOLING'
03AD   20 00 45 4E 44 20 4F 46  ' END OF'
03B5   20 53 50 4F 4F 4C 49 4E  ' SPOOLIN'
03BD   47 8D 00 00              'G   '
03C1   JMP $AF08                'SYNTAX ERROR' ausgeben
```

Bild 10. 1. Listing für Spooling von Diskette

Die Bedienung dieses Programmes ist denkbar einfach. Nachdem Sie Ihr Listing auf der Diskette abgelegt haben, tippen Sie:

```
SYS 828,"filename"
```

Danach erhalten Sie vom Computer die Rückmeldung

```
SPOOLING filename
READY.
```

und der Druckvorgang beginnt.

Ist der Ausdruck abgeschlossen, befinden sich Drucker und Floppy noch beide in Betriebsbereitschaft. Um sie wieder in den 'Normalzustand' zu versetzen tippen Sie:

```
SYS 828
```

(diesmal ohne Filename), und die LED an der Floppy erlischt. Gleichzeitig erhalten Sie die Meldung

```
END OF SPOOLING
READY.
```

## *10.3 Dem seriellen Bus Beine gemacht*

Wir haben bestimmt alle schon einmal über die Geschwindigkeit der 1541 geschimpft, wenn wir ein Programm schnell laden wollten und eine schier ewige Zeit darauf warten mußten, daß sich der Computer mit READY zurückmeldet.

Sie werden inzwischen sicherlich schon erfahren haben, daß die Floppy an dieser 'Zeitverschwendung' eigentlich keine Schuld hat. Die Ursache liegt in der Tatsache, daß der serielle Bus keine übermäßig schnelle Datenübertragung erlaubt.

Insgesamt ist der serielle Bus von Commodore im Vergleich zu seinem 'großen Bruder', dem parallelen IEEE-488-Bus, etwa um den Faktor 10 langsamer, was die schon bekannte Verzögerung verursacht.

Haben Sie schon einmal etwas von HYPRA-LOAD gehort? Es erschien im 64'er Magazin (Ausgabe 10/1984). Dieses Programm erlaubt 5-bis 6mal schnelleres Laden von der Diskette, weil es modifizierte Busroutinen verwendet.

Die Entwicklung von HYPRA-LOAD wurde durch die schnellen Kopierprogramme, die schon früher auf dem Markt waren, angeregt. Alle diese Programme arbeiten mit veränderten Busroutinen, was die Übertragungsrate mindestens auf das Niveau des parallelen Bus von Commodore hebt.

Wir wollen uns diese Busroutinen am Beispiel von HYPRA-LOAD einmal genauer betrachten, da sie für eine Fülle von Anwendungen geeignet sind und einen enormen Geschwindigkeitszuwachs bringen.

Die aufgelisteten Routinen sind alle für den Commodore 64 bestimmt und lassen sich auf den VC 20 aufgrund dessen anderer Struktur der I/O-Bausteine nicht übertragen. Beim Betrieb der schnellen Busroutinen muß der Bildschirm des C 64 grundsätzlich abgeschaltet werden, da er den Taktzyklus des Prozessors stört und eine reguläre Übertragung sonst nicht zuläßt. Wird der Bildschirm in Maschinensprache abgeschaltet, ist außerdem darauf zu achten, daß eine Warteschleife vor der Datenübertragung eingebaut wird, da der VIC eine bestimmte Zeit für den Abschaltvorgang benotigt.

Bild 10.2/3 zeigen die Sende- und Empfangsroutinen von HYPRA-LOAD.

```
      SEI          ; Interrupts verhindern
      LDA #$OB     ; ATN-Signal (Bit 3) setzen
      STA $DDOO    ; Bit 0 und 1 müssen gesetzt bleiben
WAIT  BIT $DDOO    ; Rückmeldung der Floppy abwarten
      BPL WAIT     ; (Synchronisation)
      LDA #$03     ; ATN-Signal (Bit 3) wieder löschen
      STA $DDOO    ; Bus in Übertragungsbereitschaft
      LDX #$05     ;
HOLD  DEX          ; Schleife, um der Floppy genügend Zeit
      NOP          ; zur Bereitstellung der Daten zu geben
      BNE HOLD     ;
      LDX #$04     ; 4 mal 2 Bits müssen übertragen werden
```

```
LOOP LDA $DD00   ; zwei Bits vom Bus holen
     ROL         ; Bits an die richtige Stelle bringen
     ROL         ;
     ROR $FF     ; $FF erhält das Datenbyte
     ROR         ;
     ROR $FF     ;
     NOP         ; Verzögerung für Timing
     DEX         ;
     BNE LOOP    ; nächste Datenbits
     LDA $FF     ; Datenbyte nach A holen
     EOR #$FF    ; Bits invertieren
     CLI         ; Interrupts wieder zulassen
     RTS         ; Ende -->
```

Bild 10.2 Empfangsroutine des Computers

```
     SEI         ;
     STA $77     ; Byte abspeichern
WAIT BIT $1800   ; ATN testen
     BPL WAIT    ; (Synchronisation)
     LDA #$10    ;
     STA $1800   ; Rückmeldung an Computer
BFRE BIT $1800   ; warten, bis Bus freigegeben
     BMI BFRE    ;
     LDX #$04    ; 4 mal 2 Bits müssen gesendet werden
LOOP LDA #$00    ; alle Bits in A löschen
     ROR $77     ; Datenbits an richtige Position
     ROL         ; In Akku schieben
     ROL         ;
     ROR $77     ;
     ROL         ;
     ROL         ;
     STA $1800   ; Daten zum Computer
     DEX         ;
     BNE LOOP    ; nächste Datenbits
     NOP         ; warten, um dem Computer Zeit für
     NOP         ; die Datenübernahme zu lassen
     NOP         ;
     NOP         ;
     NOP         ;
     NOP         ;
```

```
LDA #$0F    ;
STA $1800   ; Bus wieder freimachen
CLI         ; Interrupts wieder zulassen
RTS         ;
```

Bild 10.3 Senderoutine der Floppystation

Wenn Sie diese Routinen betrachten, werden Sie erkennen, daß nur eine einmalige Synchronisation am Anfang der Übertragung stattfindet. Während des Busbetriebs sind die Taktzyklen der einzelnen Befehle genau ausgezählt, was diesen Routinen eine erstaunliche Zuverlässigkeit verleiht.

Wenn Sie sich die Routinen ansehen, durfte es Ihnen keine Probleine bereiten, Programme zu schreiben, die eine Übertragung vom Computer zur Floppy möglich machen.

# 11
# Die Hardware der 1541

# 11 Die Hardware der 1541

Nachdem wir uns nun die ganze Zeit mit der Software beschäftigt haben, wollen wir uns jetzt der Hardware zuwenden.

Ich habe Ihnen schon in den vorherigen Kapiteln den Tip gegeben, Ihre 1541 aufzuschrauben und das Oberteil abzunehmen, um der Floppystation bei der Arbeit zuzusehen, und um die korrekte Ausführung bei eigenen Programmen zu überwachen.

## *11.1 Das Laufwerk der 1541*

Wie wir schon erfahren haben, arbeitet die 1541 softsektoriert. Wenn Sie sich die Laufwerksmechanik betrachten, werden Sie erkennen, daß die Floppy noch nicht einmal das Indexloch, das bei allen softsektorierten Disketten vorhanden ist, benutzt. Dieses Indexloch zeigt normalerweise immer den Beginn eines Tracks an, damit ein schneller sequentieller Datenzugriff möglich ist. Commodore geht hier einen anderen Weg: da jeder Sektor einen langen Header besitzt, in dem alle Informationen zur Orientierung des DC vorhanden sind, benötigt die 1541 keinen Hardwarezusatz zum Erkennen bestimmter Sektoren.

Diese Eigenschaft erlaubt es uns, alle Disketten doppelseitig zu verwenden. Wenn Sie diese Methode noch nicht angewendet haben, sollten Sie es einmal versuchen. Fast alle Disketten besitzen auf beiden Seiten eine Magnetschicht. Bei einseitig verkauften Disketten wird zwar die zweite Seite nicht getestet, sie ist in den meisten Fällen aber voll nutzbar. Schneiden Sie mit einer Schere oder einem Messer eine zweite Schreibschutzkerbe auf der gleichen Höhe in die entgegengesetzte Seite der Diskettenhülle. Zur Sicherheit sollten Sie die Diskettenscheibe bei diesem 'Manöver' in die entgegengesetzte Ecke der Hülle schieben, um eine Beschädigung bei Verschnitt zu vermeiden.

Denken Sie auch beim doppelseitigen Benutzen der Disketten daran, daß die beschriebene Seite der Magnetscheibe grundsätzlich der obenliegenden Seite gegenüber liegt. Das heißt, die zweite Seite einer Diskette befindet sich in der Regel dort, wo das Etikett klebt.

Schieben Sie die präparierte Diskette in das Laufwerk (andere Seite nach oben) und starten Sie das Formatieren. Sie werden sich wundern, wieviel Geld für Disketten man auf diese Weise einsparen kann.

Aber jetzt wieder zum Laufwerk. Der Tonkopf mit dem gegenüberliegenden Andruckfilz bewegt sich auf Laufschienen. Angetrieben wird er über ein Stahlband, das direkt über ein Rad läuft, das auf die Achse des Steppermotors aufgesteckt ist. Auf diesem Rad befinden sich unter anderem zwei Anschläge. Den einen bekommen Sie ziemlich oft zu hören, wenn der Tonkopf zurückfährt, um neu positioniert zu werden. Der zweite Anschlag dient als Schutz beim Positionieren des Tonkopfes auf Spuren, die höhere Nummern als 40 haben.

Im Gegensatz zu manchen anderen Floppystationen arbeitet die 1541, was die Abfrage der Schreibschutzkerbe angeht, nicht mechanisch mit einem Taster, sondern mit einer Infrarotlichtschranke. Das ist besonders dann zu beachten, wenn Tesafilm als Löschschutz verwendet werden soll. Er ist nämlich untauglich, da das Infrarotlicht durch die durchsichtige Folie hindurchgeht.

## *11.2 Eingriffe in die Platine bei der 1541*

Die 1541 gibt es inzwischen in mehreren Ausführungen. Bei der alten Version mit der großen Platine muß diese übrigens abgeschraubt werden, um Einblick in das Laufwerk zu nehmen.

Wichtig ist für uns die dauerhafte Änderung der Gerätenummer. Um die richtigen Stellen ausfindig zu machen, betrachten Sie bitte Bild 11.1. Die Pfeile deuten auf die beiden Lötbrücken.

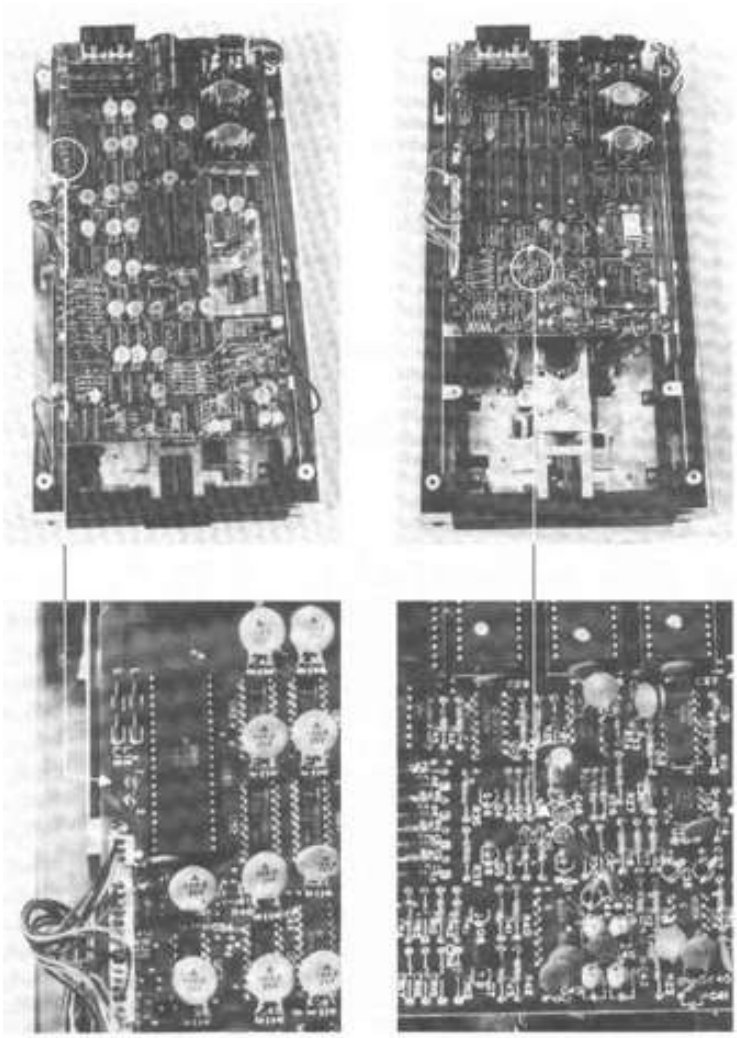


Bild 11.1 Die Pfeile kennzeichnen die Trennstellen

Soll die Gerätenummer geändert werden, ist die Floppy auszuschalten! Danach nehmen Sie ein Messer oder einen anderen spitzen Gegenstand und durchtrennen eine oder beide Verbindungen. Der hintere der beiden Kontakte erhöht die Nummer jeweils um eins, der vordere um zwei. Bild 11.2 zeigt die Aufstellung der erreichbaren Gerätenummern.

| zu durchtrennen | Nummer ist jetzt |
|---|---|
| --- | 8 |
| hinten | 9 |
| vorne | 10 |
| hinten+vorne | 11 |

Bild 11.2 Einstellen der Gerätenummer

An dieser Stelle ein Hinweis:

Die Hardware der Floppy ist sehr empfindlich gegenüber externen Einflüssen. Wenn Sie also nicht wissen, an welchem Ende ein Lotkolben heiß wird oder generell beim Basteln zwei linke Hände haben, dann sollten Sie die beschriebenen Eingriffe besser bleiben lassen. Ich empfehle Ihnen in diesem Fall einen Fachmann aufzusuchen. Außerdem sollten Sie wissen, daß Sie bei einem unautorisierten Eingriff in die Floppy auf jeden Fall die Werksgarantie verlieren.

Ein weiterer Eingriff dürfte für all jene von Interesse sein, die ihre Disketten, wie vorhin empfohlen, beidseitig verwenden. Hier ist es in der Regel notwendig, eine zusätzliche Schreibschutzkerbe in die Diskettenhülle zu schneiden. Diese Maßnahme kann unterbleiben, wenn wir die Abfrage der Lichtschranke beeinflussen. Wenn Sie sich die Stecker auf der Platine ansehen, entdecken Sie einen (den größten), der die Nummer P6 besitzt. Dieser Stecker enthält unter anderem zwei Kabel, die lila und orange Farbkennungen haben (die zwei äußersten Kabel rechts am Stecker). Wenn Sie diese zwei Leitungen miteinander verbinden, ist die Floppy nicht mehr in der Lage, den Wechsel einer Diskette und eine eventuell aufgeklebte Schreibschutzplakette zu erkennen. Es ist also nicht mehr nötig, die Schreibschutzkerben anzubringen.

Sie sollten sich jedoch der Tatsache bewußt sein, daß Sie mit dieser Maßnahme eine wichtige Schutzvorrichtung der Floppy 'lahmlegen', was Sie bei der geringsten Unaufmerksamkeit wichtige gespeicherte Daten kosten kann, wenn Sie aus Versehen den SCRATCH-Befehl benutzen. Auch das automatische Initialisieren der Disketten funktioniert nicht mehr, da ein Diskettenwechsel nicht erkannt wird.

## *11.3 Tips zur Behandlung der Floppy*

Daß Sie mit der 1541 ein Präzisionsgerät vor sich stehen haben, bedarf eigentlich keiner weiteren Erläuterung. Wie man die Lebensdauer und Zuverlässigkeit der Floppystation erhöht, das sei in diesem Kapitel verraten.

Generell gilt natürlich das Fernhalten von Staub und Schmutzpartikelchen aller Art, sowohl von der Diskette als auch von der Floppy. Allein die Rauchteilchen im Zigarettenqualm haben schon eine Größe, die für den Verlust einiger Bits verantwortlich sein kann.

Rattattattat... wieder einmal! Ein Fehler auf der Diskette, und schon fährt der Tonkopf der Floppy mit viel Lärm an den Anschlag zurück um dann neu positioniert zu werden. Diese Eigenart der 1541 läßt sich ohne Eingriff in deren Betriebssystem leider nicht ändern und sollte uns eigentlich auch gar nicht sonderlich stören. Der Nachteil ist nur, daß das Rad, das für die Übertragung der Steppermotorbewegung auf den Tonkopf verantwortlich ist, lediglich auf die Motorachse aufgesetzt ist. Deshalb kann es passieren, daß sich der Tonkopf bei häufigem Anschlagen dejustiert und vom Kundendienst wieder eingestellt werden muß.

Eine Möglichkeit, die Geräuschentwicklung der Floppy einzuschränken, ist das Einlegen von Filzstückchen in die Schraubverbindungen zwischen Gehäuseboden und Metallrahmen.

Diese Methode erfordert nicht viel Mühe, aber sie macht die Arbeit mit der Floppy angenehmer.

# 12
# Fehler im DOS 2.6 der 1541

# 12 Fehler im DOS 2.6 der 1541

Leider kann man dem DOS der 1541 nicht ganz vertrauen, da es in einigen Punkten ganz erhebliche Mängel und Fehler aufweist, die einen unter Umständen böse Überraschungen erleben lassen. Hier ist eine Aufzählung der festgestellten Mängel.

1) Der Befehl BLOCK-READ:

Dieser Befehl ist eigentlich nicht verwendbar, da das erste Byte eines Blocks grundsätzlich nicht mitgelesen wird. Außerdem hat er noch eine weitere 'nette' Eigenschaft, die einen ziemlich verblüfft. Wenn Sie mit B-R einen Block in den DOS-Puffer lesen, wird die Anzahl der verfügbaren Bytes dieses Blocks immer mit der Tracknummer des gelesenen Blocks gleichgesetzt, das heißt, wenn Sie einen Block von Spur 17 lesen, und dann mit GET# dessen Inhalte untersuchen wollen, schickt die Floppy nach 17 Bytes ein EOI zum Computer und fängt danach wieder mit der Übertragung des ersten Bytes an. Sie erhalten immer die ersten 17 Bytes Ihres Blocks und niemals die übrigen 239.

2) Der Befehl BLOCK-WRITE:

Auch diesen Befehl haben wir in der Anwendung durch einen USER-Befehl ersetzt. Dieser Befehl hat nämlich den Nachteil, daß er beim Schreiben eines Blocks grundsätzlich das erste Byte eines Blocks zerstört, indem er den Pufferzeiger dort hineinschreibt und der vorherige Wert dadurch verlorengeht.

3) Der Befehl BLOCK-ALLOCATE:

Dieser Befehl arbeitet einwandfrei, solange der Block, der belegt werden soll, auch frei ist. Ist er es nicht, sucht der Befehl automatisch den nächsten freien Block und zeigt ihn uns an. Der Machteil ist nur, daß es hier egal ist, ob der nächste freie Block auf Track 18 liegt oder nicht. Außerdem belegt das DOS automatisch alle Blöcke auf der Spur, auf der der nächste freie Block entdeckt wurde, was sicherlich nicht der Sinn der Sache ist.

4) Der Befehl REPLACE (@):

Bei diesem Befehl muß immer darauf geachtet werden, daß noch soviel Platz auf der Diskette vorhanden ist, wie das abzuspeichernde Programm benötigt. Andernfalls erscheint die Fehlermeldung DISK FULL. Es wird nämlich immer erst das neue Programm abgespeichert und anschließend das alte gelöscht.[2]

5) Falscher Leerinhalt von Blöcken:

Dieser Fehler hat keine Auswirkungen auf die korrekte Arbeit der 1541. Es handelt sich hier um einen 'Schönheitsfehler'. Normalerweise sollen die Blöcke beim Formatieren mit 256 x $00 vorbesetzt werden. Dies ist bei der 1541 nicht der Fall. Hier steht am Anfang $4B gefolgt von 255 x $01. Wie schon gesagt, kein ernsthafter Fehler, der es jedoch immerhin erlaubt, Disketten der 1541 zu identifizieren.

Am Schluß muß noch erwähnt werden, daß es noch ein paar mysteriöse Fehler gibt, die in den neueren ROMs der 1541 bereits ausgemerzt sind, sofern sie je bestanden haben. Besonders Besitzer der alten Floppies mit der großen Platine können infolge der noch implementierten Autostart-Routine manchmal Probleme mit dem Initialisieren der Floppy nach einem Reset bekommen.

[2] @ST: Der Befehl hat noch einen schwerwiegenden Fehler, der zum Datenverlust führen kann. Am besten ist er nicht zu benutzen, sondern stattdessen die Datei unter neuem Namen abzuspeichern und die alte zu löschen. Siehe auch die Diskussion dieses Problems im Buch „Die Floppy 1570/1571“.

# 13
# Die 1541 im Vergleich zu den anderen CBM-Floppies

# 13 Die 1541 im Vergleich zu den anderen CBM-Floppies

Hier soll die 1541 einmal mit den anderen CBM-Floppies verglichen werden, wobei an den Anfang gleich einmal die sogenannte 'Vollkompatibilität' zwischen 4040 und 1541 gestellt werden soll. Diese beiden Floppystationen sind keinesfalls vollkompatibel; ja, es kann sogar unter Umständen zu schwerwiegenden Problemen kommen, wenn es um den Austausch von Disketten beider Formate geht.

Beim Lesen der Disketten haben die beiden Floppystationen keine Probleme. Die Probleme ergeben sich erst beim Schreiben. Es ist nämlich so, daß die Lücke nach dem Header eines Datenblocks bei der 4040 genau 100 Bits umfaßt. Bei der 1541 sind dies jedoch nur 92 Bits. Die Folgen werden gleich deutlich, wenn wir uns mit dem Vorgang des Lesens und Schreibens noch einmal beschäftigen. Beim Lesen von Daten macht der Unterschied nichts. Hier wird der Blockheader gelesen und danach auf die nächste SYNC-Markierung gewartet.

Beim Schreiben sieht die Sache etwas anders aus. Hier wird ebenfalls der Blockheader gelesen. Danach werden die Bytes, die die Lücke darstellen, einfach abgezählt, da der Floppystation die Anzahl der Bytes des eigenen Formates ja bekannt ist. Anschließend wird auf den Schreibmodus umgeschaltet.

Schreibt man nun mit einer 1541 auf Disketten, die auf der 4040 formatiert wurden, ist das verhältnismäßig harmlos. Die Floppy wartet eben die Lücke des 4040 Formats nicht ganz ab und beginnt schon ein Byte früher zu schreiben, so daß die Lücke anschließend auf die 8 Bytes des DOS 2.6 Formats geschrumpft ist.

Beim Schreiben mit der 4040 auf Disketten der 1541 sieht die Sache hingegen etwas anders aus. Hier passiert folgendes:

Die Floppy wartet 9 Bytes als Lücke ab, wobei sie sich dann bereits in der SYNC-Markierung befindet, da diese bei der 1541 schon nach 8 Bytes beginnt. Die Folge ist, daß sich der Anfang der ursprünglichen SYNC-Markierung (gefolgt vom nicht definierten Bereich beim Umschalten des DC auf Schreiben) noch auf der Diskette befindet. Wird dieser nicht definierte Bereich jetzt beim späteren Lesen als "0"-Bit erkannt und beträgt die Länge des 'SYNC-Rests' der 1541 zufällig 8 Bits oder mehr, erkennt der DC

diesen Rest als vollständige SYNC-Markierung und fängt an, die Daten zu lesen, obwohl jetzt erst die eigentliche SYNC-Markierung folgt. Die Folge ist ein "22, READ ERROR", weil der DC nach der Datenblock-SYNC-Marke immer den Wert $07 erwartet.

Die Fehlerwahrscheinlichkeit ist zum Glück ziemlich gering (ca. 2 Blöcke/Diskette). Sie sollten sich diese Möglichkeit einer Fehlerquelle aber immer vor Augen halten.

Das größere Problem liegt in der unterschiedlichen Justierung der Tonkopfe, auch bei neu eingestellten Laufwerken. Hier passiert es relativ oft, daß 'alte' Informationen der einen Floppy von der Floppy des ändern Typs nicht genau überschrieben werden. Die Folge ist eine Art 'Mischinformation' beim Lesen, die für eine Fülle von Fehlern sorgen kann und dies auch tut.

Für die Freaks unter Ihnen, die es gerne genau wissen wollen, hier noch eine Tabelle der technischen Daten einiger CBM-Flop-pies.

| Modell | 1541/2031 | 4040 | 8050 | 1001/8250 |
|---|---|---|---|---|
| Laufwerke | 1/1 | 2 | 2 | 1/2 |
| Kopfe/Laufwerk | 1/1 | 1 | 1 | 2/2 |
| Speicher (KBytes) | 2/2 | 4 | 4 | 4/4 |
| Anzahl Tracks | 35/35 | 35 | 77 | 154/154 |
| Blöcke/Drive | 664/664 | 664 | 2052 | 4133/4133 |
| max. Kbytes | 170/170 | 170 | 500 | 1000/1000 |
| Directorytrack | 18/18 | 18 | 38+39 | 38+39/38+39 |
| Prozessor(en) | 1/1 | 2 | 2 | 2/2 |
| U/min | 300/300 | 300 | 300 | 300/300 |
| Übertragungsrate | | | | |
| intern (Kbytes/s) | 38/38 | 38 | 38 | 38/38 |
| extern (Kbytes/s) | 0.4/1.8 | 1.8 | 1.8 | 1.8/1.8 |

# Anhang I
## RAM-Belegung der 1541

| Adresse | Bedeutung der Speicherstelle(n) |
|---|---|
| $0000 -$0005 | Diese Speicherstellen bilden die Schnittstelle zwischen dem Hauptprogramm und dem Disk—Controller. Das Hauptprogramm schreibt die Kommandos für den DC in eine dieser Speicherstellen, wobei sie dann beim nächsten IRQ—Aufruf ausgeführt werden. Jede Jobausführung hinterläßt unter anderem auch eine Rückmeldung, die die korrekte Ausführung eines Jobs oder einen auf getretenen Fehler signalisiert. |

| Jobcodes | | Statusmeldungen | |
|---|---|---|---|
| $80 | Lesen eines Sektors | $01 | fehlerfreie Durchführung |
| $90 | Schreiben eines Sektors | $02 | Blockheader nicht gefunden |
| $A0 | Verify eines Sektors | $03 | SYNC nicht gefunden |
| $B0 | Suchen eines Sektors | $04 | Datenblock nicht gefunden |
| $C0 | Anschlagen des Kopfes (BUMP) | $05 | Datenprüfsumme falsch |
| $D0 | Programm im Puffer ausführen | $07 | Fehler bei Verify |
| $E0 | Jobprogramm im Puffer ausführen, nachdem das Laufwerk hochgefahren wurde | $08 | Diskette schreibgeschützt |
| | | $09 | Headerprüfsumme falsch |
| | | $0A | Datenblock zu lang |
| | | $0B | falsche ID im Blockheader |
| | | $0F | keine Diskette im Laufwerk |
| | | $10 | Fehler bei Dekodierung |

| Adresse | Bedeutung der Speicherstelle(n) |
|---|---|
| $0000 | Jobspeicher für Puffer 0 ($0300-$03FF) |
| $0001 | Jobspeicher für Puffer 1 ($0400-$04FF) |
| $0002 | Jobspeicher für Puffer 2 ($0500-$05FF) |
| $0003 | Jobspeicher für Puffer 3 ($0600-$06FF) |
| $0004 | Jobspeicher für Puffer 4 ($0700-$07FF) |
| $0005 | Jobspeicher für Puffer 5 (im RAM nicht vorhanden) |
| $0006/7 | Track und Sektor für Puffer 0 |
| $0008/9 | Track und Sektor für Puffer 1 |
| $000A/B | Track und Sektor für Puffer 2 |
| $000C/D | Track und Sektor für Puffer 3 |
| $000E/F | Track und Sektor für Puffer 4 |
| $0010/1 | Track und Sektor für Puffer 5 (n.v.) |

| Adresse | Bedeutung der Speicherstelle(n) |
|---|---|
| $0012/3 | ID der Diskette im ASCII-Code; die beiden Zeichen der aktuellen ID werden bei jedem Blocksuchbefehl gelesen und hier aktualisiert abgespeichert. Auch das Initialisierkommando benutzt diesen Befehl und bringt die ID dadurch auf den neuesten Stand |
| $0014/5 | ID für Drive 1; bei der 1541 nicht implementiert |
| $0016 – $001A | Hier sind die Bytes für den aktuellen Blockheader gespeichert, und zwar sind das:<br>$0016 erstes Zeichen der ID (ID 1)<br>$0017 zweites Zeichen der ID (ID 2)<br>$0018 Tracknummer des Blocks<br>$0019 Sektornummer des Blocks<br>$001A Prüf summe über den Blockheader<br>Auf der Diskette stehen diese Werte genau in der umgekehrten Reihenfolge! |
| $001B | Unbenutzt |
| $001C | Flag für Änderung beim Schreibschutz; Diskettenwechsel |
| $001D | Funktion von $001C für Drive 1; unbenutzt |
| $001E | zeigt Zustand der Schreibschutzvorrichtung an |
| $001F | Funktion von $001E für Drive 1; unbenutzt |
| $0020 | Flags für Drivestatus:<br>Bit 4: Drivemotor im Ausschaltmodus? 1=ja; 0=nein<br>Bit 5: Drivemotor 1=an; 0=aus<br>Bit 6: soll Steppermotor aktiviert werden? 1=ja; 0=nein<br>Bit 7: Laufwerk bereit für Zugriff 1=nein; 0=ja |
| $0021 | Funktion von $0020 für Drive 1; unbenutzt |
| $0022 | Nummer des aktuellen Tracks für Disk-Controller |
| $0023 | Funktion von $0022 für Drive 1; unbenutzt[3] |
| $0024 – $002D | Zwischenspeicher für den Blockheader, nachdem dieser in den GCR-Code umgewandelt wurde. |
| $002E/F | Zwischenspeicher für Pufferadresse bei GCR-Umwandlung |
| $0030/1 | aktuelle Pufferadresse |
| $0032/3 | Zeiger auf aktuellen Blockheader beim Schreiben |
| $0034 | Zeiger in Puffer für GCR-Codierung |
| $0035 | Unbenutzt |
| $0036 | Zähler für GCR/Binär-Umwandlung |
| $0037 | unbenutzt |
| $0038 | Kennzeichen ($07) für Beginn eines Datenblocks |
| $0039 | Kennzeichen ($08) für Beginn eines Blockheaders |
| $003A | Zwischenspeicher für Prüfsummen |
| $003B | unbenutzt |
| $003C | Unbenutzt |

[3] @ST: $23 wird benutzt, um zwischen 1540 und 1541 Timing zu unterscheiden.

| Adresse | Bedeutung der Speicherstelle(n) |
|---|---|
| $003D | aktuelle Drivenummer für DC; bei der 1541 immer B |
| $003E | gerade aktives Laufwerk; $FF = keine Laufwerk |
| $003F | Puffernummer für gerade aktuellen Jobcode |
| $0040 | Zähler für SCR/Binär-Umwandlung |
| $0041 | Puffernummer des nächsten Jobs |
| $0042 | Tracknummer, auf die der Kopf positioniert werden soll |
| $0043 | enthält die maximale Zahl der Sektoren für einen Track |
| $0044 | Zwischenspeicher |
| $0045 | Zwischenspeicher für aktuellen Jobcode |
| $0046 | Unbenutzt |
| $0047 | enthält aktuelles Kennzeichen für Beginn eines Datenblocks. Wird nur bei RESET einmal auf $07 gesetzt und kann vom Benutzer verändert werden, wobei das Hi—Nibble des wertes immer auf 0 ($0—) stehen sollte, da der DC sonst Probleme bei der Endeerkennung der SYNC—Markierung bekommen kann.<br>Diese Speicherstelle kann zur Erzeugung eines '22, READ ERROR' mit Inhalten verwendet werden, wenn sie vor dem Schreiben eines Blocks abgeändert wird. |
| $0048 | Verzögerungszähler für Drivemotor.<br>Nach Einschalten wird dieser Wert auf 60 gesetzt, um eine Verzögerung von 1.5 Sekunden zu erhalten, in der der Motor hochlaufen kann.<br>Bei Abschalten des Motors wird hier der Wert 255 gespeichert; Ausschalten erst nach einer Verzögerung von 6.4 Sekunden. |
| $0049 | Zwischenspeicher für den Stackpointer |
| $004A | Zähler für den Kopftransport; Werte >127 bewegen den Kopf nach innen; Werte <128 bewegen ihn nach außen. |
| $004B | Zwischenspeicher |
| $004C | Nummer des zuletzt gelesenen Sektors |
| $004D | Nummer des nächsten zu lesenden Sektors |
| $004E | Hi-Byte eines Zeigers in den nächsten Puffer, der von GCR—Code in den Binärcode umgewandelt werden soll.<br>Werte im Ausweichpuffer werden zuerst umgewandelt; deshalb steht hier der Zeiger auf den anderen Teil des zu codierenden Blocks. |
| $004F | Lo—Byte des Zeigers in den Puffer zur Umwandlung |
| $0050 | zeigt an, ob der Inhalt des aktuellen Fuffers in Binärcode (0) oder in GCR-Code (1) vorliegt. |
| $0051 | aktuelle Tracknummer bei der Formatierung; enthalt $FF, wenn keine Formatierung aktiviert ist |

| Adresse | Bedeutung der Speicherstelle(n) |
|---|---|
| $0052 - $0055 | Zwischenspeicher bei der Umwandlung von 4 Bytes in den GCR-Code |
| $0056 - $005D | Zwischenspeicher bei der Umwandlung von 5 GCR—Bytes in den Binärcode |
| $005E | steht normalerweise auf 4 und enthält damit die Anzahl der Schritte, die zum Anfahren und Abbremsen des Steppermotors benötigt werden. |
| $005F | normalerweise 4; Faktor zum Anfahren und Abbremsen des Steppermotors |
| $0060 | Anzahl der noch anzufahrenden/abzubremsenden Schritte |
| $0061 | Anzahl der noch zu fahrenden Schritte beim schnellen Steppermodus (Laufmodus) |
| $0062/3 | Adresse der aktuellen Steppermotor—Routine; steht am Anfang auf $FA05 (keine Kopfbewegung) |
| $0064 | minimale Anzahl von Schritten für den schnellen Steppermodus; normalerweise $C8 (200) |
| $0065/6 | Zeiger auf NMI-Routine; steht nach RESET auf $EB22 |
| $0067 | Flag für aufgetretenen NMI |
| $0068 | Flag zum Ermöglichen (0) oder Sperren (1) der automatischen Initialisierung bei einem -29, DISK ID MISMATCH' Error. |
| $0069 | Abstand der Sektoren bei der Zuteilung; normalerweise 10 |
| $006A | Anzahl der Leseversuch eines Sektors; Bit 6 zeigt an, ob der Kopf auf (1) oder neben dem Track (0) positioniert ist; Bit 7 verhindert einen BUMP bei Lesefehlern, wenn es auf 1 steht. |
| $006B/C | Zeiger auf Sprungtabelle der USER-Vektoren. Steht normalerweise auf $FFEA. |
| $006D/E | Zeiger auf Bitmuster für einen Track in der BAM |
| $006F | Anzahl der Jobs für Drive 0 |
| $0070 | Funktion von $006F für Drive 1; unbenutzt |
| $0071 | Zwischenspeicher |
| $0072 | Zwischenspeicher; steht nach RESET auf $FF |
| $0073 | Zwischenspeicher |
| $0074 | Zwischenspeicher |
| $0075/6 | Zeiger auf $0100; wird nach RESET gestellt |
| $0077 | Gerätenummer + $20 für das LISTEN—Kommando |
| $0078 | Gerätenummer + $40 für das TALK—Kommando |
| $0079 | Flag für LISTEN (1/0) |
| $007A | Flag für TALK (1/0) |
| $007B | Flag für Adressierung |
| $007C | Flag für ATN-Signal vom seriellen Bus |

| Adresse | Bedeutung der Speicherstelle(n) |
|---|---|
| $007D | Flag für Prozessor im ATN—Modus |
| $007E | Flag für letzten Programmzugriff |
| $007F | aktuelle Drivenummer; enthält immer $00 |
| $0080 | aktuelle Tracknummer |
| $0081 | aktuelle Sektornummer |
| $0082 | aktuelle Kanalnummer |
| $0083 | Sekundäradresse für Befehlsausführung |
| $0084 | interne Sekundäradresse |
| $0085 | aktuelles Datenbyte für Ein-/Ausgabe |
| $0086 | Zwischenspeicher |
| $0087 | Zwischenspeicher |
| $0088 | Zwischenspeicher |
| $0089 | Zwischenspeicher |
| $008A | Zwischenspeicher |
| $008B | Speicher für Rechenergebnisse |
| - $008E | |
| $008F | Akkumulator für Berechnungen |
| - $0093 | |
| $0094/5 | Zeiger auf Directorypuffer; enthält $0205 |
| $0096 | Kommando vom lEEE—Bus; unbenutzt |
| $0097 | MY PA Flag; steht immer auf $00 |
| $0098 | Bitzähler für seriellen 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 |
| | Alle diese Adressen werden als Zeiger vom B—P—Befehl verwendet! |
| $00A3/4 | Adresse des INPUT-Puffers; steht auf $0200 |
| $00A5/6 | Zeiger auf ERROR-Puffer; steht auf $02D6 |
| $00A7 | Pufferbelegungstabelle; enthält für jeden Kanal die |
| - $00AD | entsprachende Puffernummer; Puffernummer = $FF, wenn Puffer |
| | nicht belegt. |
| $00AE | Pufferstatustabelle; enthält für jeden Kanal die entsprechende |
| - $00B4 | Puffernummer; Puffernummer = $FF, wenn der Puffer inaktiv ist. |
| $00B5 | Tabelle der Lo—Bytes der Recordnummern für jeden Puffer |
| - $00BA | |
| $00BB | Tabelle der Hi-Bytes der Recordnummern für jeden Puffer |
| - $00C0 | |

| Adresse | Bedeutung der Speicherstelle(n) |
|---|---|
| $00C1 | Tabelle der Zeiger auf den jeweils nächsten Record |
| - $00C6 | |
| $00C7 | Tabelle der Recordlängen für jeden Puffer |
| - $00CC | |
| $00CD | Tabelle der Side-Sektoren für jeden Puffer |
| - $00D2 | |
| $00D3 | Zeiger auf ersten Filenamen |
| $00D4 | Zeiger auf Beginn des Record |
| $00D5 | Nummer des aktuellen Side—Sektors |
| $00D6 | Zeiger in Side—Sektor |
| $00D7 | Zeiger in Record |
| $00D8 | Sektornummer des Fileeintrags im Directory |
| - $00DC | |
| $00DD | Tabelle der Zeiger in Directory-Eintragungen |
| - $00E1 | |
| $00E2 | Standardwerte für Drivenummern; hier alle 0 |
| - $00E6 | |
| $00E7 | Flags für spezielle Zeichen in der Eingabezeile; 'wild cards' |
| - $00EB | ('*', '?') |
| $00EC | Kanal—Filetyp |
| - $00F1 | |
| $00F2 | Kanal Status |
| - $00F7 | |
| $00F8 | Zwischenspeicher für EOI |
| $00F9 | Aktuelle Puffernummer für Befehlscode |
| $00FA | Zwischendurch verwendete Tabelle |
| - $00FE | |
| $00FF | Flag für Laufwerk aktiv |
| $0100 | Funktion von $00FF für Drive 1; unbenutzt |
| $0101 | Formatkennzeichen aus Block 18,0 nach Initialisierung |
| $0102 | Funktion von $0101 für Drive 1; unbenutzt |
| $0103 | unbenutzt |
| $0104 | Bereich des Hardware-Stack; nicht benutzbar |
| - $0145 | |
| $01BB | Ausweichpuffer bei der Umwandlung von Werten in den GCR—Code, da |
| - $01FF | hierbei deren Länge um den Faktor 5 zu 4 zunimmt. |
| $0200 | INPUT-Puffer; enthält Befehlsstring vom Computer |
| - $0229 | |
| $022A | Codenummer des auszuführenden Befehls |

| Adresse | Bedeutung der Speicherstelle(n) |
|---|---|
| $022B | Kanaltabelle; enthält für jeden Kanal den Statuswert; |
| - $023D | $FF nicht benutzt $81 zum Schreiben geöffnet |
| | $41 Schreiben/Lesen $01 zum Lesen geöffnet |
| $023E | aktuelles Datenbyte für jeden Kanal |
| - $0243 | |
| $0244 | Tabelle der Zeiger auf das letzte aktuelle Zeichen in jedem für |
| - $0249 | den Kanal zuständigen Pufferspeicher |
| $024A | gerade behandelter Filetyp |
| $024B | Länge des Befehlsstrings |
| $024C | Zwischenspeicher für Sekundäradresse |
| $024D | Zwischenspeicher für Jobcode |
| $024E | Arbeitsspeicher beim Suchen des nächsten Sektors |
| $024F/0 | Pufferbelegungsspeicher; 1 = Puffer belegt |
| $0251 | Flag für 'BAM dirty', d.h. BAM wurde im Speicher abgeändert und |
| | stimmt nicht mehr mit der Version auf Diskette überein. |
| $0252 | Funktion von $0251 für Drive 1; unbenutzt |
| $0253 | Flag für Directory—Eintrag gefunden |
| $0254 | Flag für '*'; Ausgabe des Directory |
| $0255 | Flag für Befehlsausführung; <>$00, wenn Befehl anliegt |
| $0256 | Flag für Belegung der Kanalnummern |
| $0257 | Nummer des letzten benutzten Puffers |
| $0258 | Recordlänge |
| $0259 | Side—Sektor—Tracknummer |
| $025A | Side-Sektor-Sektornummer |
| $025B | Tabelle; enthält letzten Befehlscode für Puffer |
| - $025F | |
| $0260 | Sektornummern der Directoryeinträge in den Puffern |
| - $0265 | |
| $0266 | Zeiger auf die Directoryeinträge in den Puffern |
| - $026B | |
| $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 | aktueller Schreibkanal |
| $0271 | aktueller Lesekanal |
| $0272/3 | Speicher für Anzahl der Blöcke |
| $0274 | Länge des Befehlsstrings im INPUT—Puffer |
| $0275 | Zeichen zum Suchen im String |
| $0276 | letztes Zeichen +1 im INPUT-Puffer |
| $0277 | Länge von Filename 1 |

| Adresse | Bedeutung der Speicherstelle(n) |
|---|---|
| $0278 | Anzahl der Kommas; Länge von Filename 2 |
| $0279 | Zeiger auf 2. Filenamen |
| $027A | Zeiger auf Filetabelle |
| - $027F | |
| $0280 | Tracknummern der Files für den aktuellen Puffer |
| - $0284 | |
| $0285 | Sektornummern der Files für den aktuellen Puffer |
| - $0289 | |
| $028A | Joker-(*)-Flag |
| $028B | Flags für Befehlssyntax |
| $028C | Anzahl der Lesezugriffe |
| $028D | Flag für Diskettenzugriff |
| $028E | Nummer des zuletzt benutzten Laufwerks |
| $028F | Flag für Fileeintrag im Directory gefunden |
| $0290 | Sektornummer des aktuellen Directoryblocks |
| $0291 | Sektornummer des ersten Directoryeintrags |
| $0292 | Zeiger auf ersten gültigen Directoryeintrag |
| $0293 | zeigt letzten Block an; enthält dann *00 |
| $0294 | aktueller Pufferzeiger |
| $0295 | Zähler für Fileeinträge |
| $0296 | Filetyp |
| $0297 | Betriebsart des aktuellen Files (Lesen/Schreiben) |
| $0298 | zeigt Fehler bei Ausführung eines Jobs an |
| $0299 | Speicher für Zeiger |
| $029A | Byte für Kopf Positionierung .$029B/C Flag für BAM wiederhergestellt |
| $029B/C | |
| $029D/E | Tracknummer der BAM |
| $029F/0 | Funktion von $029D/E für Drive 1; unbenutzt |
| $02A1 | Zwischenspeicher für BAM Eintragungen |
| - $02B0 | |
| $02B1 | Puffer für Directory |
| - $02D4 | |
| $02D5 | ERROR-Puffer; enthält auszugebende Fehlermeldung |
| - $02F8 | |
| $02F9 | Flag für BAM neu auf Diskette schreiben (weil 'dirty') |
| $02FA | Lo-Byte der Anzahl der freien Blöcke auf Diskette |
| $02FB | Funktion von $02FA für Drive 1; unbenutzt |
| $02FC | Hi-Byte der Anzahl der freien Blöcke auf Diskette |
| $02FD | Funktion von *02FC für Drive 1; unbenutzt |
| $02FE/F | Parameter für Kopftransport |
| $0300 | Puffer 0 (Hauptarbeitspuffer) |
| - $03FF | |

| Adresse | Bedeutung der Speicherstelle(n) |
|---|---|
| $0300 - $03FF | Puffer 0 (Hauptarbeitspuffer) |
| $0400 - $04FF | Puffer 1 (enthält aktuellen Teil des Directory) |
| $0500 - $05FF | Puffer 2 (USER-Puffer; normalerweise frei) |
| $0600 - $06FF | Puffer 3 (enthält letzten Block des Directory) |
| $0700 - $07FF | Puffer 4 (enthält Block 18,0 nach Initialisierung) |
| $0800 - $BFFF | keine RAM-Belegung |
| $C100 - $FFFF | DOS 2.6 1541 |

# Anhang II
# DOS-Listing der 1541

```
------------------------------------------------------------
C100                        Diese Routine schaltet die LED am
                            aktuellen (0) Laufwerk an und die LED
                            an Laufwerk 1 aus.
                            Für den Zustand der LED ist das Bit 3
                            des Port B des Diskcontrollers
                            zuständig (Bit 1 = LED an). $7F enthalt
                            dabei die Laufwerksnummer.
C100 78       SEI
C101 A9 F7    LDA #$F7
C103 2D 00 1C AND $1C00
C106 48       PHA
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
C111 09 08    ORA #$08      LED-Bit setzen (Bit 3)
C113 8D 00 1C STA $1C00     LED an
C116 58       CLI
C117 60       RTS
------------------------------------------------------------
C118                        Routine schaltet LED an Laufwerk B
                            durch Setzen von Bit 3 ein.
C118 78       SEI
C119 A9 08    LDA #$08      LED-Bit für Laufwerk 0 setzen
C11B 0D 00 1C ORA $1C00
C11E 8D 00 1C STA $1C00     LED an
C121 58       CLI
C122 60       RTS
------------------------------------------------------------
C123                        Löschen der Flags des Fehlerstatus;
                            dadurch wird der Fehlerzustand der
                            Floppy gelöscht, die LED erlischt beim
                            nächsten IRQ-Durchlauf.
C123 A9 00    LDA #$00
C125 8D 6C 02 STA $026C     Fehlernummer = 0
C128 8D 6D 02 STA $026D     Flag für LED-Blinken = 0
```

```
C12B 60       RTS
-------------------------------------------------------------
C12C                        Initialisierung des LED-Blinkens bei
                            Auftreten eines Fehlers. Aktiviert
                            werden dabei die Fehlerflags des DC, um
                            der IRQ-Routine den Fehler mitzuteilen
C12C 78       SEI
C12D 8A       TXA           X-Registerinhalt retten
C12E 48       PHA
C12F A9 50    LDA #$50      Fehlernummer setzen
C131 8D 6C 02 STA $026C
C134 A2 00    LDX #$00
C136 BD CA FE LDA $FECA,X Defaultwert für LED-Bit (8)
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
C143 AA       TAX           X-Wert zurückholen
C144 58       CLI
C145 60       RTS
-------------------------------------------------------------
C146                        Wertet die Befehlsstrings 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
C148 8D F9 02 STA $02F9     Flag für 'BAM nicht auf Diskette
                            schreiben' löschen
C14B AD 8E 02 LDA $028E     Standard Laufwerksnummer (0)
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
C157 29 0F    AND #$0F      auf 15 begrenzen
C159 C9 0F    CMP #$0F      Kommandokanal (15)?
C15B F0 03    BEQ $C160
C15D 4C B4 D7 JMP $D7B4     OPEN-Routine, da Kanalnr. <>15
C160 20 B3 C2 JSR $C2B3     Flags 'für Befehlsübernahme' setzen
C163 B1 A3    LDA ($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
C173 10 F5     BPL $C16A
C175 A9 31     LDA #$31    Fehlernummer laden und
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 Kommandos mit Dateina-
C17F 90 03     BCC $C184   men 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 finden
C18A 85 6F     STA $6F     Adresse speichern
C18C BD A1 FE  LDA $FEA1,X
C18F 85 70     STA $70
C191 6C 6F 00  JMP ($006F) Sprung auf Befehl
-----------------------------------------------------------
C194                       Abschluß eines Befehls. Stellt die OK-
                           Meldung bereit, sofern die Fehlerflag
                           in S026C =0 ist. Ansonsten erfolgt
                           Aufruf der Fehlerroutine.
C194 A9 00     LDA #$00    Flag für 'BAM nicht auf Diskette
C196 8D F9 02  STA $02F9   schreiben' löschen
C199 AD 6C 02  LDA $026C   Fehlerflag prüfen und
C19C D0 2A     BNE $C1C8   ggf. zur Fehlerbehandlung
C19E A0 00     LDY #$00
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 A9 00     LDA #$00
C1B5 95 FF     STA $FF,X   löschen
C1B7 20 BD C1  JSR $C1BD   INPUT-Puffer löschen (*0200-0228)
C1BA 4C DA D4  JMP $D4DA   interne Kanäle löschen
-----------------------------------------------------------
C1BD                       Löscht den INPUT-BUFFER von CT200 bis
                           $0228 durch überschreiben mit fSHS.
```

```
C1BD A0 28     LDY #$28
C1BF A9 00     LDA #$00
C1C1 99 00 02  STA $0200,Y
C1C4 88        DEY
C1C5 10 FA     BPL $C1C1
C1C7 60        RTS
```

---

C1C8 — Fehlermeldung ausgeben, wobei Spur- und Sektornummer gleich $00 gesetzt werden. Fehlernummer muß bei Einsprung im Akku stehen.

```
C1C8 A0 00     LDY #$00
C1CA 84 80     STY $80     Spurnuminer
C1CC 84 81     STY $81     Sei« tornummer
C1CE 4C 45 E6  JMP $E645   Fehlerbehandlung
```

---

C1D1 — Untersucht die Eingabezelle auf einen Doppelpunkt ':'. Wird einer gefunden, so steht seine Position in Y und wird minus 2 nach $027A gespeichert. Anschließend wird die LED am Laufwerk eingeschaltet.

```
C1D1 A2 00     LDX #$00
C1D3 8E 7A 02  STX $027A   Zeiger auf Filetabelle löschen
C1D6 A9 3A     LDA #$3A    ASCII Wert für ':'
C1D8 20 68 C2  JSR $C268   Befehlsstring auf ':' durchsuchen
C1DB F0 05     BEQ $C1E2
C1DD 88        DEY         Zeiger auf ':' in Y
C1DE 88        DEY         Zeiger 2 Bytes vor ':' setzen
C1DF 8C 7A 02  STY $027A   zeigt dann auf Laufwerksnummer
C1E2 4C 68 C3  JMP $C368   Laufwerksnummer setzen; LED an
```

---

C1E5 — Sucht nach einem Doppelpunkt ': ' in der Eingabezeile.

```
C1E5 A0 00     LDY #$00    Stelle für Suchbeginn
C1E7 A2 00     LDX #$00    Anzahl der gefundenen Kommas
C1E9 A9 3A     LDA #$3A    ':'
C1EB 4C 68 C2  JMP $C268   Befehlsstring durchsuchen
```

---

C1EE — Prüft die Eingabezeile auf die Bestandteile mehrteiliger Befehle. Dazu zählt z.B. RENAME. Es wird dabei auf ':', '=' und ',' durchsucht.

```
C1EE 20 E5 C1  JSR $C1E5   Sucht nach ':' in Befehlszelle
C1F1 D0 05     BNE $C1F8   verzweige, wenn gefunden
C1F3 A9 34     LDA #$34
C1F5 4C C8 C1  JMP $C1C8   34 SYNTAX ERROR- ausgeben
C1F8 88        DEY
C1F9 88        DEY
C1FA 8C 7A 02  STY $027A   Position der Laufwerksnummer
C1FD 8A        TXA         Komma vor dem Doppelpunkt?
C1FE D0 F3     BNE $C1F3   Ja, dann 'SYNTAX ERROR'
C200 A9 3D     LDA #$3D    ASCII Wert 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
C210 8E 77 02  STX $0277   Lange von Filename 1 setzen
C213 8E 78 02  STX $0278   Lange 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
C225 8D 8A 02  STA $028A   Jokerflag 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   Lange 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
C260 8D 6C 02  STA $026C   Fehlerflag setzen
C263 A9 30     LDA #$30    Fehlernummer
C265 4C C8 C1  JMP $C1C8   '30 SYNTAX ERROR' ausgeben
-------------------------------------------------------------
C268                       Durchsucht die Eingabezelle auf ein
                           bestimmtes Zeichen (in Akku). Y
                           bestimmt die Startposition beim Suchen;
                           X enthält den Zeiger in die Tabelle der
                           Filenamenzeiger. Es wird auch auf die
                           Sonderzeichen im Befehlsstring
                           geachtet, z.B. '*', '?' und ','. Wenn
                           das gesuchte Zeichen nicht gefunden
                           wird, wird das Z—Flag gesetzt.
C268 8D 75 02  STA $0275   Zeichen zum Suchen
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
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
C27C C9 3F     CMP #$3F    vergleiche auf '?'
C27E D0 03     BNE $C283
C280 EE 8A 02  INC $028A   Jokerflag setzen
C283 C9 2C     CMP #$2C    vergleiche auf ','
C285 D0 E4     BNE $C26B
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
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
-------------------------------------------------------------
C2B3                      Initialisiert alle Kommandotabellen und
                          —zeiger und prüft 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 Null
C2B7 88       DEY
C2B8 F0 10    BEQ $C2CA   verzweige, wenn Eins
C2BA B9 00 02 LDA $0200,Y Zeichen aus Befehlsstring holen
C2BD C9 0D    CMP #$0D    Marke für Zeilenende?
C2BF F0 0A    BEQ $C2CB   verzweige, wenn ja
C2C1 88       DEY
C2C2 B9 00 02 LDA $0200,Y davorstehendes Zeichen holen
C2C5 C9 0D    CMP #$0D    Marke für Zeilenende?
C2C7 F0 02    BEQ $C2CB   verzweige, wenn ja
C2C9 C8       INY
C2CA C8       INY         Zeiger auf alten Wert setzen
C2CB 8C 74 02 STY $0274   und als Zeilenlänge speichern
C2CE C0 2A    CPY #$2A    mit max. Lange (42) vergleichen
C2D0 A0 FF    LDY #$FF
C2D2 90 08    BCC $C2DC   verzweige, wenn kleiner
C2D4 8C 2A 02 STY $022A   Befehlsnummer löschen
C2D7 A9 32    LDA #$32    Fehlernummer
C2D9 4C C8 C1 JMP $C1C8   '32 SYNTAX ERROR' ausgeben
```

```
C2DC A0 00     LDY #$00
C2DE 98        TYA
C2DF 85 A3     STA $A3     Zeiger auf 1NPUT-Puffer LO
C2E1 8D 58 02  STA $0258   Rcordlänge
C2E4 8D 4A 02  STA $024A   gerade behandelter Filetyp
C2E7 8D 96 02  STA $0296
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
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
C30F D0 EC     BNE $C2FD
C311 60        RTS
-----------------------------------------------------------------
C312                       Laufwerksnummern holen und setzen.
C312 AD 78 02  LDA $0278   Länge des zweiten Filenamens
C315 8D 77 02  STA $0277   Länge des ersten Filenamens
C318 A9 01     LDA #$01
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
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
C336 EC 78 02  CPX $0278   alle Nummern geholt?
C339 90 EA     BCC $C325   verzweige, wenn nein
C33B 60        RTS
-----------------------------------------------------------------
```

```
C33C                        Holt Laufwerksnummer aus Befehls-
                            string oder setzt ansonsten den
                            Standardwert (0). A enthält bei
                            Einsprung den Zeiger in den INPUT-
                            Buffer auf die Laufwerksnummer.
C33C AA       TAX           Zeiger merken
C33D A0 00    LDY #$00
C33F A9 3A    LDA #$3A      ':'
C341 DD 01 02 CMP $0201,X   Doppelpunkt 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
C34C 98       TYA           Laufwerksnummer
C34D 29 01    AND #$01      auf 0 und 1 beschränken
C34F A8       TAY
C350 8A       TXA           Position zurückholen
C351 60       RTS
C352 BD 00 02 LDA $0200,X   Laufwerksnummer holen
C355 E8       INX
C356 E8       INX
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     verzweige unbedingt
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 Laufwerksnummer und
                            schaltet LED am Laufwerk ein.
C368 A9 00    LDA #$00
C36A 8D 8B 02 STA $028B     Befehlssyntaxflag 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
```

```
C381 D0 ED     BNE $C370   Laufwerksnummer suchen
C383 CE 8B 02  DEC $028B   Flags für nicht gefunden setzen
C386 A9 00     LDA #$00
C388 29 01     AND #$01
C38A 85 7F     STA $7F     Standardlaufwerk setzen
C38C 4C 00 C1  JMP $C100   LED einschalten und RTS
-------------------------------------------------------------
C38F                       Schaltet Laufwerksnummer in $7F durch
                           Umkehrung von Bit 0 um.
C38F A5 7F     LDA $7F
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
-------------------------------------------------------------
C398                       Zeiger für Filenamen setzen und Filetyp
                           holen und setzen.
C398 A0 00     LDY #$00
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 $027A,Y Zeiger auf Fi lenainen holen
C3AB A8        TAY         als Index benutzen und
C3AC B1 A3     LDA ($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
-------------------------------------------------------------
C3BD                       Testet auf Laufwerksnummer. Wird weder
                           0 noch 1 gefunden, so wird in A das Bit
                           7 gesetzt ansonsten werden alle
                           ungültigen Bits gelöscht.
C3BD C9 30     CMP #$30    Drive 0 ?
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    Akku negieren (Bit 7=1)
C3C7 29 81     AND #$81    andere Werte ausschließen
C3C9 60        RTS
-------------------------------------------------------------
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 Drivenuaimern zum prüfen
C3D5 68        PLA         Akkuinhalt zurückholen
C3D6 05 6F     ORA $6F
C3D8 48        PHA         Schleife zum Prüfen aller vorhan-
C3D9 A9 01     LDA #$01    denen Drivenummer 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
C3E4 06 6F     ASL $6F
C3E6 06 6F     ASL $6F
C3E8 4A        LSR         Drivenummer 0 oder 1
C3E9 90 EA     BCC $C3D5   verzweige, bei Drivenummer 0
C3EB 06 6F     ASL $6F
C3ED D0 E6     BNE $C3D5
C3EF 68        PLA
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 unischalten
C411 A9 00     LDA #$00
```

```
C413 8D 8C 02  STA $028C   Anzahl der Lesezugriffe 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    Fehlernummer in Akku
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, initial isisren
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
C42F 8D 8C 02  STA $028C   Anzahl der Lesezugriffe 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
-----------------------------------------------------------------
C44F                       Bytes zum Suchen einer Datei
C44F 00 80 41 01 01 01 01 81
C44F 81 81 81 42 42 42 42
-----------------------------------------------------------------
C44F                       Sucht alle Files, die in der
                           Eingabezeile gefordert werden und
                           stellt danach die erforderlichen Zeiger
                           zur weiteren Bearbeitung her. Aussprung
                           mit RTS, wenn File gefunden worden ist.
C44F 20 CA C3  JSR $C3CA   Drive(s) auf Zugriff vorbereiten
C452 A9 00     LDA #$00
C454 8D 92 02  STA $0292   Suche nach gültigem File 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
C464 8D 8D 02  STA $028D   setze Flag für Lesezugriff
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
C494 8D 8D 02  STA $028D   Flag für Lesezugriff 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
C49F 8D 92 02  STA $0292   Suche nach gültigem File 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; Ende
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 $C4B5   weitersuchen, wenn ungleich
C4D7 60        RTS         File gefunden; Ende
-------------------------------------------------------------
C4D8                       Vergleichen aller Fileeinträge im
                           Directory mit denen im Befehlsstring;
                           ggf. erfolgt Setzen eines Zeigers.
C4D8 A2 FF     LDX #$FF
C4DA 8E 53 02  STX $0253   Flag für Eintrag gefunden 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         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         Ende der Eingabezeile suchen
C502 20 A6 C6  JSR $C6A6
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
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 Fileeintrags
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
C550 48        PHA         und 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 und 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
C572 B1 94     LDA ($94),Y Tracknummer des Files holen
C574 9D 80 02  STA $0280,X und ersten Track abspeichern
C577 C8        INY
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
C58B 8D 8F 02  STA $028F   Flag für Fileeinträge gefunden
C58E AD 78 02  LDA $0278   Anzahl der Filenamen
C591 8D 79 02  STA $0279
C594 CE 79 02  DEC $0279   vermindern
C597 10 01     BPL $C59A   verzweige, wenn größer 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
C5A8 8D 8F 02  STA $028F   Flag zurücksetzen und
C5AB 60        RTS         Ende, da alle Namen gefunden
-------------------------------------------------------------
C5AC                       Vorbereitung zum Suchen im Directory.
                           Setzen aller Standardwerte.
C5AC A0 00     LDY #$00
C5AE 8C 91 02  STY $0291   Sektor des ersten Directoryeintrags
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
C5CC 8D 95 02  STA $0295   Zähler für Fileeinträge 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 Directory lesen
C5E1 D0 18     BNE $C5FB   verzweige, wenn kein DEL File
C5E3 AD 91 02  LDA $0291   wurde schon ein DEL File gefunden 7
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
C5F5 8D 92 02  STA $0292   auf ersten Directoryeintrag 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   nein, es wird nach einem DEL File
                           gesucht; also weitersuchen
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   Zeiger auf aktuellen Fileeintrag
C614 20 C8 D4  JSR $D4C8   Zeiger auf Eintrag setzen
C617 A9 FF     LDA #$FF
C619 8D 53 02  STA $0253   Flag für Eintrag gefunden 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
-------------------------------------------------------------
C63D                       Testet auf Diskette in Laufwerk und
                           initialisiert, wenn Diskette gewechselt
                           wurde.
C63D A5 68     LDA $68     Flag zum Sperren des Initialisieren
```

```
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
C649 8D 98 02  STA $0298   Job Fehlerflag 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 ansprechbar ?
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 98        TYA         Fehlerflag in A
C662 95 FF     STA $FF,X   und abspeichern
C664 D0 03     BNE $C669   verzweige, wenn Fehler
C666 20 42 D0  JSR $D042   Drive initialisieren
C669 A6 7F     LDX $7F     aktuelle Drivenummer
C66B B5 FF     LDA $FF,X   Fehlercode holen
C66D 60        RTS
-----------------------------------------------------------------
C66E                       Schreibt Filenamen aus dem
                           Kommandopuffer in den Directorypuffer.
                           A enthält die Länge des Namens; X
                           enthält die Position im Befehlsstring;
                           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 wiederholen
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 $C687   nein; größer ? dann verzweigen
C67F A9 A0     LDA #$A0    kleiner, dann den Rest mit
C681 91 94     STA ($94),Y 'Shift Space' auffüllen
C683 C8        INY
C684 CA        DEX
C685 D0 FA     BNE $C681
C687 60        RTS
```

```
------------------------------------------------------------
C688                        Schreibt den Inhalt des INPUT-Buffer in
                            irgendeinen anderen Puffer. Y enthält
                            die Nummer des anderen Puffers; X
                            enthält die Position des ersten
                            Zeichens im INPUT—Butter.
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
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
C69D F0 06    BEQ $C6A5     Ende, wenn Y > 255
C69F E8       INX
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-Buffer sich in X befindet.
C6A6 A9 00    LDA #$00
C6A8 8D 4B 02 STA $024B     Länge des Namens setzen
C6AB 8A       TXA           Position des ersten Zeichens
C6AC 48       PHA           auf Stack retten
C6AD BD 00 02 LDA $0200,X   Zeichen aus INPUT-Puffer 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
C6CD 60       RTS
-------------------------------------------------------------
C6CE                        Holt einen Fileeintrag auis dem
                            Directory über den internen Lesekanal
                            (SA=$11/17)
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
C6DA 68       PLA           Sekundäradresse zurückholen
C6DB 85 83    STA $83
C6DD 60       RTS
-------------------------------------------------------------
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 offnen
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
C70B 4C 8F C3 JMP $C38F     auf anderes Drive umschalten; Ende

C70E A9 00    LDA #$00
C710 8D 73 02 STA $0273     Anzahl der Blocks Hi
C713 8D 8D 02 STA $028D     Flag für Diskettenzugriff löschen
C716 20 B7 C7 JSR $C7B7     Directoryüberschrift schreiben
C719 38       SEC
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 der Blocks 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 Blocks Lo
C729 B1 94     LDA ($94),Y Anzahl der Blocks 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
C737 C9 64     CMP #$64    vergleiche Blockzahl Lo mit 100
C739 90 01     BCC $C73C   verzweige, wenn kleiner
C73B CA        DEX
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    '<' 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
C755 B9 C0 FE  LDA $FEC0,Y 2. Buchstaben holen
C758 9D B1 02  STA $02B1,X und in Puffer schreiben
C75B CA        DEX
C75C B9 BB FE  LDA $FEBB,Y 1. Buchstaben holen
C75F 9D B1 02  STA $02B1,X und in Puffer schreiben
C762 CA        DEX
C763 CA        DEX
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 Blank'
C76D 9D B1 02  STA $02B1,X in den Puffer schreiben
C770 CA        DEX
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
C779 88       DEY
C77A C0 03    CPY #$03
C77C B0 F5    BCS $C773
C77E A9 22    LDA #$22    '"' Anführungszeichen
C780 9D B1 02 STA $02B1,X vor Filenamen in Puffer schreiben
C783 E8       INX
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
C798 E8       INX
C799 E0 20    CPX #$20    Ende des Namens schon erreicht ?
C79B B0 0A    BCS $C7A7   verzweige, wenn ja
C79D A9 7F    LDA #$7F
C79F 3D B1 02 AND $02B1,X Bit 7 in den restlichen Zeichen
C7A2 9D B1 02 STA $02B1,X löschen
C7A5 10 F1    BPL $C798
C7A7 20 B5 C4 JSR $C4B5   nächsten Directoryeintrag suchen
C7AA 38       SEC
C7AB 60       RTS
-----------------------------------------------------------------
C7AC                      Löschen des Puffers für den Namen im
                          Directory durch Füllen mit $20.
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
C7B4 D0 FA    BNE $C7B0   27 mal
C7B6 60       RTS
-----------------------------------------------------------------
C7B7                      Kopf des Directory für Anzeige erzeugen
                          und in Directory-Puffer $02B0-02D4
                          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
C7C2 85 6F     STA $6F
C7C4 A6 7F     LDX $7F     aktuelle Drivenummer
C7C6 8E 72 02  STX $0272   als Blockanzahl Lo in Speicher
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 Hl als aktuellen
C7D3 85 95     STA $95     Pufferzeiger Hl 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    '1'
C7E4 2C        .BYTE 2C
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
C7F1 10 F2     BPL $C7E5
C7F3 A9 12     LDA #$12    'RVS on'
C7F5 8D B1 02  STA $02B1   in den Puffer schreiben
C7F8 A9 22     LDA #$22    '"'
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
-------------------------------------------------------------
C806                       Zeile mit 'BLOCKS FREE' erzeugen und in
                           Directory-Puffer 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
C812 10 F7     BPL $C80B
C814 4C 4D EF  JMP $EF4D   Anzahl der freien Blocke holen
-------------------------------------------------------------
C817                       Bytes für 'BLOCKS FREE'
```

```
C817 42 4C 4F 43 4B 53 20 46
C81F 52 45 45 2E
-----------------------------------------------------------------
C823                        SCRATCH-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
C82E 85 86     STA $86      Anzahl der gelöschten Flies
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
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
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
C85A 35 E7     AND $E7,X    Bit 5 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 'Fehlermeldung'
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 (SA=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 Block7)
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 in 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 Bleck wieder schreiben
C8BE 4C 99 D5  JMP $D599    und Ende des Jobs abwarten
-----------------------------------------------------------------
C8C1                        BACKUP-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 Dis-
                            kette.
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 DC
C8E0 A5 03     LDA $03
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    Fehlernummer
C8EA A2 00     LDX #$00    für Drive 0
C8EC 4C 0A E6  JMP $E60A   '21 READ ERROR' ausgeben
C8EF 60        RTS         Job fehlerlos beendet
-------------------------------------------------------------
C8F0                       COPY-Befehl
C8F0 A9 E0     LDA #$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 Kanäle
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   '3l 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 File-
C91F C9 2A     CMP #$2A    namens mit '*' vergleichen
C921 D0 05     BNE $C928   verzweige, wenn kein '*'
C923 A9 30     LDA #$30    Fehlernummer
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   ordnungsgemäß kopieren
-------------------------------------------------------------
C932                       Parameter für das Kopieren einer ganzen
                           Diskette setzen (n.v.).
```

```
C932 A9 00     LDA #$00
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 Sektornununer
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   125 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    Fehlernummer
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   Kanal Status
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 Filas
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 im 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 SA setzen
C9C3 20 EB D0  JSR $D0EB   unbenutzten Lesekanal suchen
C9C6 20 25 D1  JSR $D125   Abfrage auf REL-Datei
C9C9 D0 03     BNE $C9CE   verzweige, wenn kein REL—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
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 REL—File
CA26 20 A0 D9  JSR $D9A0   internen Lesekanal öffnen
CA29 A0 01     LDY #$01
CA2B 20 25 D1  JSR $D125   Filetyp auf REL-Datei testen
CA2E F0 01     BEQ $CA31   verzweige, wenn keine REL—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
-----------------------------------------------------------------
```

```
CA35                        Byte über den internen Lesekanal holen
                            und Sekundäradresse für internen
                            Lesekanal setzen.
CA35 A9 11     LDA #$11     17 als SA des internen Lesekanals
CA37 85 83     STA $83      setzen
CA39 20 9B D3  JSR $D39B    ein Byte über 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 Kanalstatustabelle 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 REL-File testen
CA4B F0 05     BEQ $CA52    verzweige, wenn kein REL-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 (SA=18)
CA61 85 83     STA $83      als aktuellen Kanal speichern
CA63 20 07 D1  JSR $D107    freien Schreibkanal suchen
CA66 20 D3 D1  JSR $D1D3    Drivenummer setzen
CA69 20 CB E1  JSR $E1CB    letzten Side-Sektor setzen
CA6C 20 9C E2  JSR $E29C    Block 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 Side-Sektors
CA75 85 86     STA $86      zwischenspeihern
CA77 A9 00     LDA #$00
CA79 85 88     STA $88
CA7B 85 D4     STA $D4      Zeiger auf Beginn des Record
CA7D 85 D7     STA $D7      Zeiger in Record
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
CA88 20 20 C3  JSR $C320   Drivenummern aus Befehlszeile holen
CA8B A5 E3     LDA $E3     Standardlaufwerksnummer (0)
CA8D 29 01     AND #$01    Bit für Nummer isolieren
CA8F 85 E3     STA $E3     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 setzen, um Suche auf beiden
CA97 85 E2     STA $E2     Drives zu initialisieren
CA99 20 4F C4  JSR $C44F   File im Directory suchen
CA9C 20 E7 CA  JSR $CAE7   Namen schon vorhanden?
CA9F A5 E3     LDA $E3     Drivenummer (Standardwert)
CAA1 29 01     AND #$01    Nummer isolieren
CAA3 85 7F     STA $7F     als aktuelle Nummer übernehmen
CAA5 A5 D9     LDA $D9     Sektornummer des Eintrags
CAA7 85 81     STA $81     als aktuelle Sektor Hummer nehmen
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
CAB4 20 C8 D4  JSR $D4C8   Pufferzeiger auf Namen setzen
CAB7 20 93 DF  JSR $DF93   Nummer des aktiven Puffers holen
CABA A8        TAY         nach Y
CABB AE 7A 02  LDX $027A   Zeiger auf Filetabelle
CABE A9 10     LDA #$10    Anzahl 16, Länge des Filenamens
CAC0 20 6E C6  JSR $C66E   Namen in Puffer schreiben
CAC3 20 5E DE  JSR $DE5E   Sektor wieder auf Diskette
CAC6 20 99 D5  JSR $D599   schreiben und Ende abwarten
CAC9 4C 94 C1  JMP $C194   Ende, Fehlermeldung bereitstellen
-------------------------------------------------------------
CACC                       Prüft auf Vorhandensein des aktuellen
                           Files.
CACC A5 E8     LDA $E8     Masken für Fileidentifikation
CACE 29 07     AND #$07    Filetyp isolieren
CAD0 8D 4A 02  STA $024A   und als aktuellen Typ abspeichern
CAD3 AE 78 02  LDX $0278   Filenamenzeiger
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 ungleich Null
```

```
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 gegebenen! Namen vorhanden?
CAEA BD 80 02  LDA $0280,X zugehörige Tracknummer
CAED F0 05     BEQ $CAF4   verzweige, wenn gleich Null
CAEF A9 63     LDA #$63    Fehlernummer
CAF1 4C C8 C1  JMP $C1C8   '63, FILE EXISTS' ausgeben
CAF4 CA        DEX         Zeiger auf Tracknummern minus 1
CAF5 10 F3     BPL $CAEA   verzweige, wenn größer Null
CAF7 60        RTS         Ende, kein File mit gleichem Namen
-------------------------------------------------------------
CAF8                       MEMORY-Befehle
CAF8 AD 01 02  LDA $0201   zweites Zeichen des Befehlsstrings
CAFB C9 2D     CMP #$2D    mit '-' vergleichen
CAFD D0 4C     BNE $CB4B   verzweige, wenn ungleich
CAFF AD 03 02  LDA $0203   viertes Zeichen aus Befehlsstring
CB02 85 6F     STA $6F     als Adresse Lo speichern
CB04 AD 04 02  LDA $0204   fünftes Zeichen aus Befehlsstring
CB07 85 70     STA $70     als Adresse Hi speichern
CB09 A0 00     LDY #$00
CB0B AD 02 02  LDA $0202   drittes Zeichen aus String
CB0E C9 52     CMP #$52    mit 'R' vergleichen (M-R)
CB10 F0 0E     BEQ $CB20   verzweige, wenn gleich
CB12 20 58 F2  JSR $F258   (RTS)
CB15 C9 57     CMP #$57    mit 'W' vergleichen (M-W)
CB17 F0 37     BEQ $CB50   verzweige, wenn gleich
CB19 C9 45     CMP #$45    mit 'E' vergleichen (M-E)
CB1B D0 2E     BNE $CB4B   Fehler, wenn ungleich
CB1D 6C 6F 00  JMP ($006F) Sprung in Programm
-------------------------------------------------------------
CB20                       M-R-Befehl
CB20 B1 6F     LDA ($6F),Y Byte an Adresse holen
CB22 85 85     STA $85     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 kleiner
CB2B AE 05 02  LDX $0205   Anzahl der zu lesenden Bytes
CB2E CA        DEX         minus 1
CB2F F0 14     BEQ $CB45   verzweige, wenn nur ein Byte
CB31 8A        TXA         Anzahl der Bytes
```

```
CB32 18       CLC
CB33 65 6F    ADC $6F     plus Startadresse
CB35 E6 6F    INC $6F     Adresse auf zweites Byte
CB37 8D 49 02 STA $0249   als Endwert merken
CB3A A5 6F    LDA $6F     Adresse Lo
CB3C 85 A5    STA $A5     Zeiger für Fehlermeldung Lo
CB3E A5 70    LDA $70     Adresse Hi
CB40 85 A6    STA $A6     Zeiger für Fehlermeldung Hi
CB42 4C 43 D4 JMP $D443   Byte auf Bus ausgeben
CB45 20 EB D0 JSR $D0EB   unbenutzten Lesekanal suchen
CB48 4C 3A D4 JMP $D43A   Bytes auf Bus ausgeben
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 Werte aus INPUT-Puffer holen
CB53 91 6F    STA ($6F),Y und abspeichern
CB55 C8       INY         Zeiger erhöhen
CB56 CC 05 02 CPY $0205   mit Anzahl der Bytes vergleichen
CB59 90 F5    BCC $CB50   verzweige, wenn kleiner
CB5B 60       RTS         Ende
-------------------------------------------------------------
CB5C                      USER-Befehl
CB5C AC 01 02 LDY $0201   zweites Zeichen aus Befehlsstring
CB5F C0 30    CPY #$30    mit '0' vergleichen
CB61 D0 09    BNE $CB6C   verzweige, wenn ungleich
CB63 A9 EA    LDA #$EA    Adresse Lo für Sprungtabelle der
CB65 85 6B    STA $6B     USER-Befehle setzen
CB67 A9 FF    LDA #$FF    Adresse Hi für Sprungtabelle der
CB69 85 6C    STA $6C     USER-Befehle setzen
CB6B 60       RTS         Ende

CB6C 20 72 CB JSR $CB72   Adresse setzen und Befehl ausführen
CB6F 4C 94 C1 JMP $C194   Ende, Fehlermeldung bereitstellen

CB72 88       DEY         ASCII-Befehlsnummer minus 1
CB73 98       TYA
CB74 29 0F    AND #$0F    Zahlenwert isolieren
CB76 0A       ASL         mal 2 nehmen
CB77 A8       TAY         und als Zeiger benutzen
CB78 B1 6B    LDA ($6B),Y Adresse Lo des Befehls
CB7A 85 75    STA $75     setzen
CB7C C8       INY         Zeiger auf nächstes Byte
CB7D B1 6B    LDA ($6B),Y Adresse Hi des Befehls
CB7F 85 76    STA $76     ebenfalls setzen
CB81 6C 75 00 JMP ($0075) Sprung auf Befehl
```

```
CB84                        '#' 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 ggf. 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 bestimmter
CB96 D0 0D     BNE $CBA5    Puffer gewünscht
CB98 A9 01     LDA #$01     Nur #; also kein bestimmter Puffer
CB9A 20 E2 D1  JSR $D1E2    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
CBA5 A0 01     LDY #$01     Zeiger auf Puffernummer und
CBA7 20 7C CC  JSR $CC7C    diese aus Befehlsstring holen
CBAA AE 85 02  LDX $0285    Puffernummer nach X
CBAD E0 05     CPX #$05     mit Maximalwert vergleichen
CBAF B0 EF     BCS $CBA0    Fehler, wenn größer
CBB1 A9 00     LDA #$00
CBB3 85 6F     STA $6F      Masken für Puffernummer löschen
CBB5 85 70     STA $70
CBB7 38        SEC          Carry—Bit als Maskenbit setzen
CBB8 26 6F     ROL $6F      Setzen des entsprechenden Bit-
CBBA 26 70     ROL $70      musters für jeden Puffer, um
CBBC CA        DEX          Vergleichsmaske aufzubauen
CBBD 10 F9     BPL $CBB8
CBBF A5 6F     LDA $6F      Maske für Drive 0
CBC1 2D 4F 02  AND $024F    mit Pufferbelegung vergleichen
CBC4 D0 DA     BNE $CBA0    Fehler, wenn Puffer belegt
CBC6 A5 70     LDA $70      Maske für Drive 1
CBC8 2D 50 02  AND $0250    mit Pufferbelegung vergleichen
CBCB D0 D3     BNE $CBA0    Fehler, wenn Puffer belegt
CBCD A5 6F     LDA $6F      Maske für Pufferbelegung
CBCF 0D 4F 02  ORA $024F    Belegung in Tabelle eintragen
CBD2 8D 4F 02  STA $024F    und 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 abspeichern
CBDD A9 00     LDA #$00     Standardwert laden und
```

```
CBDF 20 E2 D1  JSR $D1E2   unbenutzten Kanal suchen
CBE2 A6 82     LDX $82     aktuelle Kanalnummer
CBE4 AD 85 02  LDA $0285   Sektornummer für Kanal
CBE7 95 A7     STA $A7,X   in Kanalbelegung eintragen
CBE9 AA        TAX
CBEA A5 7F     LDA $7F     aktuelle Drivenummer
CBEC 95 00     STA $00,X   für entsprechenden Puffer setzen
CBEE 9D 5B 02  STA $025B,X und in Jobtabelle eintragen
CBF1 A6 83     LDX $83     Sekundäradresse
CBF3 BD 2B 02  LDA $022B,X Kanal Status aus Tabelle holen
CBF6 09 40     ORA #$40    Bit für Schreiben/Lesen setzen
CBF8 9D 2B 02  STA $022B,X und neuen Status abspeichern
CBFB A4 82     LDY $82     Kanalnummer als Index
CBFD A9 FF     LDA #$FF    Zeiger auf letztes Zeichen im
CBFF 99 44 02  STA $0244,Y aktuellen Puffer in Tabelle
CC02 A9 89     LDA #$89    Flag für Schreiben/Lesen in
CC04 99 F2 00  STA $00F2,Y Kanalstatustabelle setzen
CC07 B9 A7 00  LDA $00A7,Y zugehörige Puffernummer holen und
CC0A 99 3E 02  STA $023E,Y in Kanaltabelle eintragen
CC0D 0A        ASL         Puffernummer mal 2
CC0E AA        TAX         als Index verwenden
CC0F A9 01     LDA #$01    Wert für Pufferzeiger auf zweites
CC11 95 99     STA $99,X   Byte im Puffer setzen
CC13 A9 0E     LDA #$0E    Code für Direktzugriff
CC15 99 EC 00  STA $00EC,Y als Filetypwert setzen
CC18 4C 94 C1  JMP $C194   Diskstatus bereitstellen
-------------------------------------------------------------
CC1B                       BLOCK-Befehle
CC1B A0 00     LDY #$00
CC1D A2 00     LDX #$00
CC1F A9 2D     LDA #$2D    ASCII-Code für '-'
CC21 20 68 C2  JSR $C268   in Befehlsstring suchen
CC24 D0 0A     BNE $CC30   verzweige, wenn gefunden
CC26 A9 31     LDA #$31    Nummer für Fehlermidung
CC28 4C C8 C1  JMP $C1C8   '31 SYNTAX ERROR' ausgeben
CC2B A9 30     LDA #$30    Nummer für Fehlermeldung
CC2D 4C C8 C1  JMP $C1C8   '30 SYNTAX ERROR' ausgeben
CC30 8A        TXA         Flag für Komma gefunden testen
CC31 D0 F8     BNE $CC2B   Fehler, wenn Komma gefunden
CC33 A2 05     LDX #$05    Zeiger in Befehlsstring
CC35 B9 00 02  LDA $0200,Y Zeichen aus String holen
CC38 DD 5D CC  CMP $CC5D,X mit 'P E W R F A' vergleichen
CC3B F0 05     BEQ $CC42   verzweige, wenn Übereinstimmung
CC3D CA        DEX         Zähler vermindern
```

```
CC3E 10 F8    BPL $CC38   verzweige, wenn größer gleich Null
CC40 30 E4    BMI $CC26   unbedingter Sprungs Fehl er ausgäbe
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 abspeichern
CC48 20 6F CC JSR $CC6F   Blockparameter holen und prüfen
CC4B AD 2A 02 LDA $022A   Befehlsnummer holen
CC4E 0A       ASL         mal 2
CC4F AA       TAX         und als Index in Tabelle
CC50 BD 64 CC LDA $CC64,X Befehlsadresse Hi aus Tabelle
CC53 85 70    STA $70     setzen
CC55 BD 63 CC LDA $CC63,X Befehlsadresse Lo aus Tabelle
CC58 85 6F    STA $6F     setzen
CC5A 6C 6F 00 JMP ($006F) Befehl ausführen
-------------------------------------------------------------
CC5D 41 46 52 57 45 50    Bytes der Namen der BLOCK-Befehle
-------------------------------------------------------------
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-W
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
CC71 A2 00    LDX #$00
CC73 A9 3A    LDA #$3A    ASCII-Code für ':'
CC75 20 68 C2 JSR $C268   in Befehlsstring suchen
CC78 D0 02    BNE $CC7C   verzweige, wenn gefunden
CC7A A0 03    LDY #$03    Zeiger auf viertes Zeichen
CC7C B9 00 02 LDA $0200,Y Zeichen aus Befehlsstring holen
CC7F C9 20    CMP #$20    mit ' ' vergleichen
CC81 F0 08    BEQ $CC8B   verzweige bei Übereinstimmung
CC83 C9 1D    CMP #$1D    'CURSOR LEFT' ?
CC85 F0 04    BEQ $CC8B   verzweige, wenn Ja
CC87 C9 2C    CMP #$2C    mit ',' vergleichen
CC89 D0 07    BNE $CC92   verzweige, wenn kein Komma
CC8B C8       INY         Zeiger erhöhen
CC8C CC 74 02 CPY $0274   Zeilenende erreicht?
CC8F 90 EB    BCC $CC7C   verzweige, wenn nein
CC91 60       RTS         Ende
```

```
CC92 20 A1 CC  JSR $CCA1   Parameter konvertieren
CC95 EE 77 02  INC $0277   Zeiger in Befehlsstring
CC98 AC 79 02  LDY $0279   Zeiger auf zweiten Teil des Strings
CC9B E0 04     CPX #$04    Maximalzahl der Werte erreicht?
CC9D 90 EC     BCC $CC8B   verzweige, wenn nein
CC9F B0 8A     BCS $CC2B   unbedingter Sprung; Fehler
-------------------------------------------------------------
CCA1                       Konvertiert die ASCII-werte aus dem
                           INPUT-Buffer in HEX-Werte und legt
                           diese in den Spur- und
                           Sektornummertabellen ab.
CCA1 A9 00     LDA #$00
CCA3 85 6F     STA $6F     Rechenbereich löschen
CCA5 85 70     STA $70
CCA7 85 72     STA $72
CCA9 A2 FF     LDX #$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    Steuerzeichen?
CCB4 90 14     BCC $CCCA   verzweige, wenn keine Ziffer
CCB6 29 0F     AND #$0F    Zahlenwert aus Byte isolieren
CCB8 48        PHA         und merken
CCB9 A5 70     LDA $70
CCBB 85 71     STA $71     Werte um eine Stelle weiterschieben
CCBD A5 6F     LDA $6F
CCBF 85 70     STA $70
CCC1 68        PLA         Wert wieder zurückholen
CCC2 85 6F     STA $6F     und abspeichern
CCC4 C8        INY         Zeiger in String erhöhen
CCC5 CC 74 02  CPY $0274   schon Stringende erreicht?
CCC8 90 E1     BCC $CCAB   verzweige, wenn nein
CCCA 8C 79 02  STY $0279   Zeiger abspeichern
CCCD 18        CLC
CCCE A9 00     LDA #$00
CCD0 E8        INX         Hex-Werte in einem Byte
CCD1 E0 03     CPX #$03    zusammenfassen
CCD3 B0 0F     BCS $CCE4
CCD5 B4 6F     LDY $6F,X
CCD7 88        DEY
CCD8 30 F6     BMI $CCD0
CCDA 7D F2 CC  ADC $CCF2,X Wert aus Dezimaltabelle addieren
CCDD 90 F8     BCC $CCD7   verzweige, wenn kein Überlauf
CCDF 18        CLC
```

```
CCE0 E6 72     INC $72
CCE2 D0 F3     BNE $CCD7
CCE4 48        PHA
CCE5 AE 77 02  LDX $0277   Zeiger in String zurückholen
CCE8 A5 72     LDA $72     errechnete Tracknummer
CCEA 9D 80 02  STA $0280,X in Tabelle eintragen
CCED 68        PLA         errechnete Sektornummer
CCEE 9D 85 02  STA $0285,X in Tabelle eintragen
CCF1 60        RTS         Ende
-------------------------------------------------------------
CCF2 01 0A 64              Dezimal Umwandlungstabelle
-------------------------------------------------------------
CCF5                       B-F-Befehl
CCF5 20 F5 CD  JSR $CDF5   Track- und Sektorparameter holen
CCF8 20 5F EF  JSR $EF5F   BAM ändern; Dirty-Flag setzen
CCFB 4C 94 C1  JMP $C194   Diskstatus bereitstellen
-------------------------------------------------------------
CCFE                       Programmrest (n.v.)
CCFE A9 01     LDA #$01    Flag für BAM nicht auf Diskette
CD00 8D F9 02  STA $02F9   schreiben setzen
-------------------------------------------------------------
CD03                       B-A-Befehl
CD03 20 F5 CD  JSR $CDF5   Track- und Sektorparameter holen
CD06 A5 81     LDA $81     Sektornummer
CD08 48        PHA         merken
CD09 20 FA F1  JSR $F1FA   nächsten freien Sektor suchen
CD0C F0 0B     BEQ $CD19   verzweige, wenn kein freier Block
CD0E 68        PLA         gewünschte Sektornummer holen
CD0F C5 81     CMP $81     mit gefundenem freien Block vergl.
CD11 D0 19     BNE $CD2C   verzweige, wenn nicht identisch
CD13 20 90 EF  JSR $EF90   Block in BAM belegen
CD16 4C 94 C1  JMP $C194   Diskstatus bereitstellen; Ende
CD19 68        PLA         gewünschte Nummer löschen
CD1A A9 00     LDA #$00    Sektor 0 neu setzen
CD1C 85 81     STA $81     und als aktuellen Sektor nehmen
CD1E E6 80     INC $80     nächsthöheren Track nehmen
CD20 A5 80     LDA $80     und den neuen Wert
CD22 CD D7 FE  CMP $FED7   mit Maximalwert (36) vergleichen
CD25 B0 0A     BCS $CD31   verzweige, wenn größer gleich 36
CD27 20 FA F1  JSR $F1FA   erneut freien Block suchen
CD2A F0 EE     BEQ $CD1A   nächsten Versuch bei Mißerfolg
CD2C A9 65     LDA #$65    neuen freien Block ausgeben
CD2E 20 45 E6  JSR $E645   '65, NO BLOCK' ausgeben;
CD31 A9 65     LDA #$65    kein weiterer freier Block mehr
```

```
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 Diskette.
CD36 20 F2 CD  JSR $CDF2   Track und Sektor setzen
CD39 4C 60 D4  JMP $D460   Block lesen
-------------------------------------------------------------
CD3C                       Unterroutine des B-R-Befehls zum Holen
                           eines Bytes aus dem Puffer in den Akku.
CD3C 20 2F D1  JSR $D12F   Zeiger in Puffer setzen
CD3F A1 99     LDA ($99,X) Byte aus Puffer holen
CD41 60        RTS         Ende
-------------------------------------------------------------
CD42                       Unterroutine des B—R—Befehls zum Lesen
                           des Blocks von 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
CD56 20 42 CD  JSR $CD42   Block lesen
CD59 20 EC D3  JSR $D3EC   Byte aus Puffer zur Ausgabe
CD5C 4C 94 C1  JMP $C194   Diskstatus bereitstellen
-------------------------------------------------------------
CD5F                       U1-Befehl (B-R-Ersatz)
CD5F 20 6F CC  JSR $CC6F   Parameter des Befehls holen
CD62 20 42 CD  JSR $CD42   Block lesen
CD65 B9 44 02  LDA $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
CD6D 99 44 02  STA $0244,Y als Endekennzeichen benutzen
CD70 4C 94 C1  JMP $C194   Diskstatus bereitstellen
-------------------------------------------------------------
CD73                       B-W-Befehl
CD73 20 F2 CD  JSR $CDF2   Kanal zum Schreiben öffnen
CD76 20 E8 D4  JSR $D4E8   Pufferzeiger setzen
CD79 A8        TAY         und 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 2
CD7F A0 01    LDY #$01    Index mit 1 setzen
CD81 A9 00    LDA #$00    Wert für Pufferzeiger Lo
CD83 20 C8 D4 JSR $D4C8   Pufferzeiger setzen
CD86 98       TYA         A=1
CD87 20 F1 CF JSR $CFF1   Inhalt von A in Puffer schreiben
CD8A 8A       TXA         Puffernummer mal 2
CD8B 48       PHA         merken
CD8C 20 64 D4 JSR $D464   Block schreiben
CD8F 68       PLA         Puffernummer mal 2 holen
CD90 AA       TAX         als Index benutzen
CD91 20 EE D3 JSR $D3EE   Byte aus Puffer in Ausgaberegister
CD94 4C 94 C1 JMP $C194   Diskstatus bereitstellen; Ende
-------------------------------------------------------------
CD97                      U2-Befehl (B-W-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 Diskette schreiben
CDA0 4C 94 C1 JMP $C194   Diskstatus bereitstellen; Ende
-------------------------------------------------------------
CDA3 20 58 F2 JSR $F258   B-E-Befehl
CDA6 20 36 CD JSR $CD36   (RTS)
CDA9 A9 00    LDA #$00    Block in Puffer lesen
CDAB 85 6F    STA $6F     Pufferadresse Lo = $00
CDAD A6 F9    LDX $F9     ab speichern
CDAF BD E0 FE LDA $FEE0,X zugehörige Adresse Hi holen
CDB2 85 70    STA $70     und ebenfalls abspeichern
CDB4 20 BA CD JSR $CDBA   Programm ausführen
CDB7 4C 94 C1 JMP $C194   Diskstatus bereitstellen; Ende
-------------------------------------------------------------
CDBA                      Ausführen des Programms bei B—E
CDBA 6C 6F 00 JMP ($006F) Sprung in Puffer
-------------------------------------------------------------
CDBD                      B-P-Befehl
CDBD 20 D2 CD JSR $CDD2   Kanal öffnen; Puffernummer holen
CDC0 A5 F9    LDA $F9     Puffernummer
CDC2 0A       ASL         mal 2 und
CDC3 AA       TAX         als Index nehmen
CDC4 AD 86 02 LDA $0286   neuer Wert des Pufferzeigers
CDC7 95 99    STA $99,X   übernehmen
CDC9 20 2F D1 JSR $D12F   Puffer— und Kanalnummer holen
CDCC 20 EE D3 JSR $D3EE   Byte aus dem Puffer holen
```

```
CDCF 4C 94 C1  JMP $C194   Diskstatus bereitstellen; Ende
-------------------------------------------------------------
CDD2                       Kanal öffnen und Puffer belegen; ggf.
                           'NO CHANNEL'
CDD2 A6 D3     LDX $D3     Befehlsstringzeiger
CDD4 E6 D3     INC $D3     erhöhen
CDD6 BD 85 02  LDA $0285,X zugehörige Kanalnummer holen
CDD9 A8        TAY         nach Y
CDDA 88        DEY
CDDB 88        DEY         minus 2
CDDC C0 0C     CPY #$0C    war Nummer größer gleich 14?
CDDE 90 05     BCC $CDE5   verzweige, wenn nein
CDE0 A9 70     LDA #$70    Fehlernummer in A
CDE2 4C C8 C1  JMP $C1C8   '70, NO CHANNEL' ausgeben
CDE5 85 83     STA $83     Sekundäradresse abspeichern
CDE7 20 EB D0  JSR $D0EB   zugehörigen Kanal prüfen
CDEA B0 F4     BCS $CDE0   Fehler, wenn schon geöffnet
CDEC 20 93 DF  JSR $DF93   Puffernummer holen
CDEF 85 F9     STA $F9     und als neue Nummer abspeichern
CDF1 60        RTS         fertig
-------------------------------------------------------------
CDF2                       Testen aller Parameter auf legalen
                           Block und belegten Puffer. Sind diese
                           Werte in Ordnung, so werden die Werte
                           zum Lesen bereitgestellt.
CDF2 20 D2 CD  JSR $CDD2   Kanal öffnen; Puffernummer holen
CDF5 A6 D3     LDX $D3     Befehlsstringzeiger
CDF7 BD 85 02  LDA $0285,X Puffernummer
CDFA 29 01     AND #$01    gibt immer 0
CDFC 85 7F     STA $7F     als aktuelle Drivenummer speichern
CDFE BD 87 02  LDA $0287,X Sektornummer aus Tabelle
CE01 85 81     STA $81     als aktuellen Sektor übernehmen
CE03 BD 86 02  LDA $0286,X Tracknummer aus Tabelle
CE06 85 80     STA $80     als aktuellen Track übernehmen
CE08 20 5F D5  JSR $D55F   Track— und Sektorwerte ok ?
CE0B 4C 00 C1  JMP $C100   LED am Laufwerk einschalten
-------------------------------------------------------------
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
CE1D E6 D7    INC $D7     Linker übergangen werden muß
CE1F A5 8B    LDA $8B     Errechnete Side-Sektornummer
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     $00 nach $92
CE31 A6 82    LDX $82     aktuelle Kanalnummer
CE33 B5 B5    LDA $B5,X   Recordnummer Lo
CE35 85 90    STA $90     merken
CE37 B5 BB    LDA $BB,X   Recordnummer Hi
CE39 85 91    STA $91     merken
CE3B D0 04    BNE $CE41   verzweige, wenn Hi ungleich Null
CE3D A5 90    LDA $90     Recordnummer Lo
CE3F F0 0B    BEQ $CE4C   verzweige, wenn gleich Null
CE41 A5 90    LDA $90     Recordnummer Lo
CE43 38       SEC
CE44 E9 01    SBC #$01    minus 1
CE46 85 90    STA $90     ist Recordnummer Lo
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 zwischenspeichern
CE50 46 6F    LSR $6F     Test auf ungeraden Wert
CE52 90 03    BCC $CE57   verzweige, wenn Zahl gerade
CE54 20 ED CE JSR $CEED   Ergebnisse plus Registerwerte
CE57 20 E5 CE JSR $CEE5   Registerinhalt mal 2
CE5A A5 6F    LDA $6F     Resultat schon erhalten ?
CE5C D0 F2    BNE $CE50   verzweige, wenn nein
CE5E A5 D4    LDA $D4     Zeiger auf Beginn des Rekord
CE60 18       CLC
CE61 65 8B    ADC $8B     zu Registerinhalt addieren
CE63 85 8B    STA $8B
CE65 90 06    BCC $CE6D
CE67 E6 8C    INC $8C
CE69 D0 02    BNE $CE6D
```

```
CE6B E6 8D    INC $8D
CE6D 60       RTS         fertig
-------------------------------------------------------------
CE6E A9 FE    LDA #$FE    Divisionsroutine. Bei Einsprung ab
                          $CE6E erfolgt Division durch 254
CE71                      254
CE71 2C       .BYTE $2C   nächsten Befehl überspringen
CE71 A9 78    LDA #$78    Bei Einsprung ab $CE71 erfolgt die
                          Division durch 120.
                          Nach Berechnung steht der Quotient in
                          $8B,8C,8D und der Rest in $90.
CE73                      120
CE73 85 6F    STA $6F     als Divisor speichern
CE75 A2 03    LDX #$03
CE77 B5 8F    LDA $8F,X
CE79 48       PHA
CE7A B5 8A    LDA $8A,X
CE7C 95 8F    STA $8F,X
CE7E 68       PLA
CE7F 95 8A    STA $8A,X
CE81 CA       DEX
CE82 D0 F3    BNE $CE77
CE84 20 D9 CE JSR $CED9   Ergebnisspeicher löschen
CE87 A2 00    LDX #$00
CE89 B5 90    LDA $90,X
CE8B 95 8F    STA $8F,X
CE8D E8       INX
CE8E E0 04    CPX #$04
CE90 90 F7    BCC $CE89
CE92 A9 00    LDA #$00
CE94 85 92    STA $92
CE96 24 6F    BIT $6F
CE98 30 09    BMI $CEA3
CE9A 06 8F    ASL $8F
CE9C 08       PHP
CE9D 46 8F    LSR $8F
CE9F 28       PLP
CEA0 20 E6 CE JSR $CEE6   Register $90/91/92 mal 2
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
CEAB 30 03    BMI $CEB0
CEAD 20 E2 CE JSR $CEE2   Register $90/91/92 mal 4
CEB0 A5 8F    LDA $8F
```

```
CEB2 18       CLC
CEB3 65 90    ADC $90
CEB5 85 90    STA $90
CEB7 90 06    BCC $CEBF
CEB9 E6 91    INC $91
CEBB D0 02    BNE $CEBF
CEBD E6 92    INC $92
CEBF A5 92    LDA $92
CEC1 05 91    ORA $91
CEC3 D0 C2    BNE $CE87
CEC5 A5 90    LDA $90
CEC7 38       SEC
CEC8 E5 6F    SBC $6F
CECA 90 0C    BCC $CED8
CECC E6 8B    INC $8B
CECE D0 06    BNE $CED6
CED0 E6 8C    INC $8C
CED2 D0 02    BNE $CED6
CED4 E6 8D    INC $8D
CED6 85 90    STA $90     Ergebnis in $8B/8C/8D; Rest in $90
CED8 60       RTS         Ende
-------------------------------------------------------------
CED9                      Ergebnisspeicher $8B,8C,8D löschen.
CED9 A9 00    LDA #$00    $00
CEDB 85 8B    STA $8B     als Inhalt des Registers
CEDD 85 8C    STA $8C     setzen
CEDF 85 8D    STA $8D
CEE1 60       RTS         Ende
-------------------------------------------------------------
CEE2                      Akkumulator für Berechnungen ($90,
                          91,92) mit 4 multiplizieren.
CEE2 20 E5 CE  JSR $CEE5  Akkumulator mal 2; danach ...
-------------------------------------------------------------
CEE5                      Akkumulator für Berechnungen ($90,
                          91,92) mit 2 multiplizieren.
CEE5 18       CLC
CEE6 26 90    ROL $90     Register einmal links verschieben
CEE8 26 91    ROL $91
CEEA 26 92    ROL $92
CEEC 60       RTS
-------------------------------------------------------------
CEED                      Ergebnisspeicher $8B,8C,8D zum
                          Akkumulator $90,91,92 addieren.
CEED 18       CLC         Addition vorbereiten
CEEE A2 FD    LDX #$FD
CEF0 B5 8E    LDA $8E,X
```

```
CEF2 75 93     ADC $93,X
CEF4 95 8E     STA $8E,X
CEF6 E8        INX
CEF7 D0 F7     BNE $CEF0
CEF9 60        RTS          Ende
-------------------------------------------------------------
CEFA                        Herstellen der Zwischenspeichertabelle.
CEFA A2 00     LDX #$00
CEFC 8A        TXA          Bereich von $FA bis $FD mit $00
CEFD 95 FA     STA $FA,X    bis $03 füllen
CEFF E8        INX
CF00 E0 04     CPX #$04
CF02 D0 F8     BNE $CEFC
CF04 A9 06     LDA #$06
CF06 95 FA     STA $FA,X    $06 nach $FE speichern
CF08 60        RTS          Ende
-------------------------------------------------------------
CF09                        Aktualisieren der
                            Zwischenspeichertabelle.
CF09 A0 04     LDY #$04
CF0B A6 82     LDX $82      aktuelle Kanalnummer
CF0D B9 FA 00  LDA $00FA,Y  Konstante für entsprechenden Kanal
CF10 96 FA     STX $FA,Y    Kanalnummer neu in Tabelle
CF12 C5 82     CMP $82      Vergleich der alten Werte
CF14 F0 07     BEQ $CF1D    Ende, wenn identisch
CF16 88        DEY          sonst
CF17 30 E1     BMI $CEFA    neue Tabelle erstellen
CF19 AA        TAX
CF1A 4C 0D CF  JMP $CF0D
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 Puffer ok?
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, wenn 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         ebenfalls 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 aktuell übernehmen
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 aktuell übernehmen
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 Schreib Job
CF51 20 8C CF JSR $CF8C   Puffer wechseln
CF54 4C 5D CF JMP $CF5D   weiter
CF57 20 8C CF JSR $CF8C   Puffer wechseln
CF5A 20 57 DE JSR $DE57   Befehlscode Lesen prüfen und an DC
CF5D 68       PLA         Sektornummer zurückholen
CF5E 85 81    STA $81     und speichern
CF60 68       PLA         Tracknummer zurückholen
CF61 85 80    STA $80     und speichern
CF63 4C 6F CF JMP $CF6F   weiter
CF66 68       PLA         Sektornummer zurückholen
CF67 85 81    STA $81     und speichern
CF69 68       PLA         Tracknummer zurückholen
CF6A 85 80    STA $80     und speichern
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 prüfen
-------------------------------------------------------------
CF76                      'NO CHANNEL' ausgeben.
CF76 A9 70    LDA #$70    Fehlernummer in A
CF78 4C C8 C1 JMP $C1C8   '70, NO CHANNEL' ausgeben
-------------------------------------------------------------
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   Ende, wenn ja
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, ok
-------------------------------------------------------------
```

```
CF8C                        Wechseln des Betriebszustandes eines
                            Puffer von aktiv nach inaktiv und
                            umgekehrt.
CF8C A6 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 in Tabelle
CF94 B5 AE     LDA $AE,X    zweite Statustabelle
CF96 49 80     EOR #$80     ebenfalls Status wechseln
CF98 95 AE     STA $AE,X    und abspeichern
CF9A 60        RTS          Ende
-------------------------------------------------------------
CF9B                        Schreiben eines Bytes über den internen
                            Schreibkanal in einen Puffer.
CF9B A2 12     LDX #$12     18 (Schreibkanal)
CF9D 86 83     STX $83      als aktuelle Sekundaradresse
CF9F 20 07 D1  JSR $D107    Schreibkanal suchen und offnen
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
CFAC 20 9D DD  JSR $DD9D    Tabelle umdrehen
CFAF A5 83     LDA $83      aktuelle Sekundäradresse
CFB1 C9 0F     CMP #$0F     15 (Kommandokanal) ?
CFB3 F0 23     BEQ $CFD8    verzweige, wenn ja
CFB5 D0 08     BNE $CFBF    unbedingter Sprung
CFB7 A5 84     LDA $84      Sekundäradresse
CFB9 29 8F     AND #$8F
CFBB C9 0F     CMP #$0F     größer oder gleich 15 ?
CFBD B0 19     BCS $CFD8    verzweige, wenn ja
CFBF 20 25 D1  JSR $D125    Dateityp holen
CFC2 B0 05     BCS $CFC9    verzweige, wenn kein SEQ-File
CFC4 A5 85     LDA $85      aktuelles Datenbyte
CFC6 4C 9D D1  JMP $D19D    in Puffer schreiben
CFC9 D0 03     BNE $CFCE    verzweige, wenn USR-File
CFCB 4C AB E0  JMP $E0AB    Datenbyte in REL-File schreiben
CFCE A5 85     LDA $85      aktuelles Datenbyte
CFD0 20 F1 CF  JSR $CFF1    in Puffer schreiben
CFD3 A4 82     LDY $82      aktuelle Kanalnummer
CFD5 4C EE D3  JMP $D3EE    nächstes Byte zur Ausgabe bereiten
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         alles fertig; Ende
CFED EE 55 02  INC $0255   weiterer Eefehl ist auszuführen
CFF0 60        RTS         Ende für diesen Durchgang
-------------------------------------------------------------
CFF1                       Datenbyte in den 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         Stack wiederherstellen
CFF8 A9 61     LDA #$61    Fehlernummer in A
CFFA 4C C8 C1  JMP $C1C8   '61, FILE NOT OFEN' ausgeben
CFFD 0A        ASL         Puffernummer mal 2
CFFE AA        TAX         als Index in Tabelle
CFFF 68        PLA         Datenbyte zurückholen
D000 81 99     STA ($99,X) und in Puffer schreiben
D002 F6 99     INC $99,X   Pufferzeiger erhöhen
D004 60        RTS         und Ende
-------------------------------------------------------------
```

```
D005                        INITIALIZE-Befehl
D005 20 D1 C1  JSR $C1D1    Parameter für Befehl prüfen
D008 20 42 D0  JSR $D042    Diskette initialisieren
D00B 4C 94 C1  JMP $C194    Diskstatus bereitstellen; Ende
-------------------------------------------------------------
D00E                        Initialisieren eines Laufwerks, dessen
                            Nummer in $7F abgespeichert ist.
D00E 20 0F F1  JSR $F10F    BAM-Pointer holen
D011 A8        TAY          als Index in Tabelle
D012 B6 A7     LDX $A7,Y    entsprechende Kanalnummer holen
D014 E0 FF     CPX #$FF     Puffer reserviert ?
D016 D0 14     BNE $D02C    verzweige, wenn ja
D018 48        PHA          Puffernummer auf Stack
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     Fehlernummer in A
D021 20 48 E6  JSR $E648    '70, NO CHANNEL' ausgeben
D024 68        PLA          Puffernummer wieder vom Stack
D025 A8        TAY          als Index in Tabelle
D026 8A        TXA          neue Puffernummer
D027 09 80     ORA #$80     Puffer als belegt kennzeichnen
D029 99 A7 00  STA $00A7,Y  und in Tabelle abspeichern
D02C 8A        TXA          neue Puffernummer
D02D 29 0F     AND #$0F     Nummer isolieren
D02F 85 F9     STA $F9      als aktuelle Puffernummer merken
D031 A2 00     LDX #$00
D033 86 81     STX $81      Sektor 0 setzen
D035 AE 85 FE  LDX $FE85    Wert 18
D038 86 80     STX $80      als Tracknummer setzen
D03A 20 D3 D6  JSR $D6D3    Parameter an DC für Jobcode
D03D A9 B0     LDA #$B0     Befehlscode Suchen eines Sektors
D03F 4C 8C D5  JMP $D58C    Job ausführen
-------------------------------------------------------------
D042                        BAM lesen und 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      Drivenummer
D04D A9 00     LDA #$00
D04F 9D 51 02  STA $0251,X  'BAM dirty flag' löschen
D052 8A        TXA          Drivenummer nach A
D053 0A        ASL          mal 2
```

```
D054 AA       TAX         als Index in Tabelle
D055 A5 16    LDA $16     ID1
D057 95 12    STA $12,X   übernehmen
D059 A5 17    LDA $17     ID2
D05B 95 13    STA $13,X   übernehmen
D05D 20 86 D5 JSR $D586   Block 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 $0200 setzen
D068 A1 99    LDA ($99,X) Formatkennzeichen aus Puffer
D06A A6 7F    LDX $7F     Drivenummer
D06C 9D 01 01 STA $0101,X Formatkennzeichen merken
D06F A9 00    LDA #$00
D071 95 1C    STA $1C,X   kein Diskettenwechsel
D073 95 FF    STA $FF,X   Laufwerk inaktiv
-----------------------------------------------------------
D075                      Anzahl der freien Blöcke auf Diskette
                          berechnen.
D075 20 3A EF JSR $EF3A   Pufferadresse nach $6D/6E holen
D078 A0 04    LDY #$04    4; Beginn der BAM
D07A A9 00    LDA #$00    Erster Additionswert
D07C AA       TAX         Hi-Byte der Anzahl in X
D07D 18       CLC         Addition voroereiten
D07E 71 6D    ADC ($6D),Y plus Anzahl der freien Blocks/Track
D080 90 01    BCC $D083   verzweige, wenn kein Überlauf
D082 E8       INX         Hi-Byte der Anzahl erhöhen
D083 C8       INY         Index auf nächsten Wert für die
D084 C8       INY         Addition einstellen
D085 C8       INY         (alle 4 Bytes ein neuer Wert)
D086 C8       INY
D087 C0 48    CPY #$48    Indexwert für Track 18 erreicht?
D089 F0 F8    BEQ $D083   verzweige, wenn ja
D08B C0 90    CPY #$90    Indexwert für letzten Track?
D08D D0 EE    BNE $D07D   weitermachen, wenn nein
D08F 48       PHA         Lo-Byte der Anzahl merken
D090 8A       TXA         Hi-Byte der Anzahl nach A
D091 A6 7F    LDX $7F     aktuelle Drivenummer
D093 9D FC 02 STA $02FC,X Hi-Byte abspeichern
D096 68       PLA         Lo-Byte zurückholen
D097 9D FA 02 STA $02FA,X und ebenfalls abspeichern
D09A 60       RTS         Ende
-----------------------------------------------------------
```

```
D09B                    Beginn des Lesen eines Blocks
D09B 20 D0 D6  JSR $D6D0   Parameter an DC übergeben
D09E 20 C3 D0  JSR $D0C3   Befehl (Block Lesen) an DC geben
D0A1 20 99 D5  JSR $D599   Ausführung des Jobs abwarten
D0A4 20 37 D1  JSR $D137   erstes Byte aus Puffer holen
D0A7 85 80     STA $80     als Tracknummer merken
D0A9 20 37 D1  JSR $D137   zweites Byte aus Puffer holen
D0AC 85 81     STA $81     als Sektornummer merken
D0AE 60        RTS         fertig
-------------------------------------------------------------
D0AF                    Beginn des Lesen eines Blocks mit dem
                        Folgeblock in einen zweiten Puffer.
D0AF 20 9B D0  JSR $D09B   Lesen eines Block; Holen der Werte
D0B2 A5 80     LDA $80     Tracknummer des nächsten Blocks
D0B4 D0 01     BNE $D0B7   verzweige, wenn weiterer Block
D0B6 60        RTS         Ende, da letzter Block gelesen
D0B7 20 1E CF  JSR $CF1E   auf Zweipufferbetrieb umschalten
D0BA 20 D0 D6  JSR $D6D0   Parameter an DC übergeben
D0BD 20 C3 D0  JSR $D0C3   Block in anderen Puffer lesen
D0C0 4C 1E CF  JMP $CF1E   Puffer wieder umschalten
-------------------------------------------------------------
D0C3                    Einstieg für Block lesen
D0C3 A9 80     LDA #$80    $80; Code für Lesen gewählt
D0C5 D0 02     BNE $D0C9   unbedingter Sprung
D0C7 A9 90     LDA #$90    Einstieg für Block schreiben
D0C9                    $90; Code für Schreiben gewählt
D0C9 8D 4D 02  STA $024D   Code in Befehlspuffer
D0CC 20 93 DF  JSR $DF93   Puffernummer holen
D0CF AA        TAX         und nach X
D0D0 20 06 D5  JSR $D506   Werte prüfen; Befehl an DC
D0D3 8A        TXA         Puffernummer nach A
D0D4 48        PHA         und merken
D0D5 0A        ASL         Puffernummer mal 2
D0D6 AA        TAX         und als Index in Tabelle
D0D7 A9 00     LDA #$00
D0D9 95 99     STA $99,X   Pufferzeiger Lo auf Null
D0DB 20 25 D1  JSR $D125   Filetyp holen
D0DE C9 04     CMP #$04    Test auf SEQ-File
D0E0 B0 06     BCS $D0E8   verzweige, wenn kein SEQ—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         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    auf 15 begrenzen
D0F3 C9 0F    CMP #$0F    mit 15 (Kommandokanal) vergleichen
D0F5 D0 02    BNE $D0F9   verzweige, wenn nicht 15
D0F7 A9 10    LDA #$10    16
D0F9 AA       TAX         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   Ende mit SEC, wenn Kanal belegt
D100 29 0F    AND #$0F    Kanalnummer isolieren
D102 85 82    STA $82     und als aktuelle Nummer merken
D104 AA       TAX         Nummer nach X
D105 18       CLC         Flag für Kanal 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    vergleiche mit 19 (Maximum)
D10B 90 02    BCC $D10F   verzweige, wenn kleiner
D10D 29 0F    AND #$0F    auf 15 begrenzen
D10F AA       TAX         SA als Index in Tabelle
D110 BD 2B 02 LDA $022B,X Kanal Status holen
D113 A8       TAY         nach Y
D114 0A       ASL         mal 2
D115 90 0A    BCC $D121   verzweige, wenn Zahl positiv war
D117 30 0A    BMI $D123   verzweige, wenn Bit 6 gesetzt war
D119 98       TYA         Nummer zurückholen
D11A 29 0F    AND #$0F    auf 15 begrenzen
D11C 85 82    STA $82     und als aktuelle Nummer speichern
D11E AA       TAX         Nummer nach X
D11F 18       CLC         Flag für Kanal ok setzen
D120 60       RTS         Ende
D121 30 F6    BMI $D119   verzweige, wenn Schreib/Lesekanal
D123 38       SEC         Flag für Kanal belegt setzen
D124 60       RTS         Ende; Fehler
-------------------------------------------------------------
```

```
D125                        Aktuellen Filetyp holen und auf
                            relatives File prüfen.
D125 A6 82     LDX $82      aktuelle Kanalnummer
D127 B5 EC     LDA $EC,X    Kanalfiletyp holen
D129 4A        LSR          Wert halbieren
D12A 29 07     AND #$07     auf 7 begrenzen
D12C C9 04     CMP #$04     vergleiche mit REL—File
D12E 60        RTS          Ende
-----------------------------------------------------------------
D12F                        Puffer- und Kanalnummer holen und
                            setzen.
D12F 20 93 DF  JSR $DF93    Puffernummer holen
D132 0A        ASL          mal 2
D133 AA        TAX          nach X
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
D13D F0 12     BEQ $D151    verzweige, wenn Null
D13F A1 99     LDA ($99,X)  Byte aus Puffer holen
D141 48        PHA          merken
D142 B5 99     LDA $99,X    Pufferzeiger Lo holen
D144 D9 44 02  CMP $0244,Y  mit Endezeiger vergleichen
D147 D0 04     BNE $D14D    verzweige, wenn noch nicht Ende
D149 A9 FF     LDA #$FF     256
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 der 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 <>0
D15B 85 85     STA $85      Byte merken
```

```
D15D B9 44 02  LDA $0244,Y Endezeiger
D160 F0 08     BEQ $D16A   verzweige, wenn Null
D162 A9 80     LDA #$80    Wert für Lesen
D164 99 F2 00  STA $00F2,Y als Kanal Status speichern
D167 A5 85     LDA $85     Byte zurückholen
D169 60        RTS         Ende
D16A 20 1E CF  JSR $CF1E   Folgeblock lesen
D16D A9 00     LDA #$00
D16F 20 C8 D4  JSR $D4C8   Pufferzeiger auf Null (A) setzen
D172 20 37 D1  JSR $D137   erstes Byte aus Puffer holen
D175 C9 00     CMP #$00    Tracknummer gleich Null?
D177 F0 19     BEQ $D192   verzweige, wenn letzter Block
D179 85 80     STA $80     Tracknummer abspeichern
D17B 20 37 D1  JSR $D137   zweites Byte aus Puffer holen
D17E 85 81     STA $81     als Folgesektor merken
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 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     Byte zurückholen
D191 60        RTS         Ende
D192 20 37 D1  JSR $D137   nächstes Byte aus Puffer holen
D195 A4 82     LDY $82     Kanalnummer als Index in Tabelle
D197 99 44 02  STA $0244,Y Byte als Endezeiger merken
D19A A5 85     LDA $85     Byte zurückholen
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 BAM suchen
D1A9 A9 00     LDA #$00
D1AB 20 C8 D4  JSR $D4C8   Pufferzeiger auf Null (A) setzan
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
D1C1 A9 02     LDA #$02
D1C3 4C C8 D4  JMP $D4C8   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
D1CC 65 6F     ADC $6F     zu gespeichertem Wert addieren
D1CE 95 99     STA $99,X   und Pufferzeiger neu setzen
D1D0 85 94     STA $94     Directorypufferzeiger Lo setzen
D1D2 60        RTS         Ende
-------------------------------------------------------------
D1D3                       Holen und setzen der Laufwerksnummer.
D1D3 20 93 DF  JSR $DF93   Puffernummer holen
D1D6 AA        TAX         nach X
D1D7 BD 5B 02  LDA $025B,X Tabelle mit Pufferbefehlscodes
D1DA 29 01     AND #$01    daraus Drivenummer isolieren
D1DC 85 7F     STA $7F     und als aktuelle Nummer speichern
D1DE 60        RTS         Ende
-------------------------------------------------------------
D1DF                       Routine zum Suchen eines Kanals und des
                           zugehörigen Puffer. Erfolgt der
                           Einsprung bei $D1DF, so wird ein
                           Schreibkanal gesucht.
D1DF 38        SEC         Flag für Schreibkanal setzen
D1E0 B0 01     BCS $D1E3   unbedingter Sprung
D1E2                       Bei Einsprung ab $D1E2 wird ein
                           Lesekanal gesucht.
D1E2 18        CLC         Flag für Lesekanal setzen
D1E3 08        PHP         und merken
D1E4 85 6F     STA $6F     A muß 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         Flags 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 in Tabelle
```

```
D1FB A9 FF     LDA #$FF    Code für Puffer unbenutzt
D1FD 99 A7 00  STA $00A7,Y in Belegungstabellen schreiben
D200 99 AE 00  STA $00AE,Y
D203 99 CD 00  STA $00CD,Y
D206 C6 6F     DEC $6F     Anzahl der Puffer erniedrigen
D208 30 1C     BMI $D226   verzweige, wenn 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
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-/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 $022B,X Kanal Status holen
D233 C9 FF     CMP #$FF    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 Puffer unbenutzt
D23D 9D 2B 02  STA $022B,X in Tabelle eintragen
D240 A6 82     LDX $82     aktuelle Kanalnummer
D242 A9 00     LDA #$00
D244 95 F2     STA $F2,X   Flags in Tabelle löschen
D246 20 5A D2  JSR $D25A   Puffer wieder freigeben
D249 A6 82     LDX $82     Kanalnummer
D24B A9 01     LDA #$01    Bitflag für Kanalnummer
D24D CA        DEX
D24E 30 03     BMI $D253   an richtige Position schieben
D250 0A        ASL
```

```
D251 D0 FA     BNE $D24D
D253 0D 56 02  ORA $0256   und in Register eintragen
D256 8D 56 02  STA $0256
D259 60        RTS         Ende
-------------------------------------------------------------
D25A                       Puffer und dessen Kanalzuordnung
                           freigeben.
D25A A6 82     LDX $82     Kanalnummer als Index
D25C B5 A7     LDA $A7,X   zugehörige Puffernummer
D25E C9 FF     CMP #$FF    Puffer zugeordnet
D260 F0 09     BEQ $D26B   verzweige, wenn nein
D262 48        PHA         Puffernummer merken
D263 A9 FF     LDA #$FF
D265 95 A7     STA $A7,X   Pufferzuordnung löschen
D267 68        PLA         Puffernummer zurückholen
D268 20 F3 D2  JSR $D2F3   Puffer freigeben
D26B A6 82     LDX $82     Kanalnummmer
D26D B5 AE     LDA $AE,X   zugehörige Puffernummer
D26F C9 FF     CMP #$FF    Puffer belegt ?
D271 F0 09     BEQ $D27C   verzweige, wenn nein
D273 48        PHA         Puffernummer merken
D274 A9 FF     LDA #$FF
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
D27E B5 CD     LDA $CD,X   zugehörige Puffernummer
D280 C9 FF     CMP #$FF    Puffer belegt ?
D282 F0 09     BEQ $D28D   verzweige, wenn nein
D284 48        PHA         Puffernummer merken
D285 A9 FF     LDA #$FF
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
D292 20 BA D2  JSR $D2BA   freien Puffer suchen
D295 10 0C     BPL $D2A3   verzweige, wenn gefunden
D297 88        DEY
D298 20 BA D2  JSR $D2BA   noch einmal versuchen
```

```
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
D2A5 30 FC     BMI $D2A3   warten auf Ende des Jobs
D2A7 A5 7F     LDA $7F     aktuelle Drivenummer
D2A9 95 00     STA $00,X   Jobcode löschen
D2AB 9D 5B 02  STA $025B,X
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 be-
                           stimmt die Puffernummern:
                           Y=0 heißt Puffer 0-7;
                           Y=1 heißt Puffer 8-15.
                           War die Suche erfolgreicn, so enthält X
                           die Puffernummer; sonst enthält X den
                           Wert $FF.
D2BA A2 07     LDX #$07    Wert für Bittest
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
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 merken
D2D1 8A        TXA         Puffernummer
D2D2 88        DEY
D2D3 30 03     BMI $D2D8   verzweige, wem Puffer 0—7
D2D5 18        CLC
D2D6 69 08     ADC #$08    Puffer 8-15
D2D8 AA        TAX         Puffernummer
D2D9 60        RTS         Ende
D2DA A6 82     LDX $82     Kanalnummer als Index
```

```
D2DC B5 A7    LDA $A7,X   zugehörige Puffernummer
D2DE 30 09    BMI $D2E9   verzweige, wenn Puffer frei
D2E0 8A       TXA         Kanalnummer
D2E1 18       CLC
D2E2 69 07    ADC #$07    ergibt maximale Anzahl
D2E4 AA       TAX
D2E5 B5 A7    LDA $A7,X   alternative Puffernummer holen
D2E7 10 F0    BPL $D2D9   Ende, wenn auch belegt
D2E9 C9 FF    CMP #$FF    Puffer frei ?
D2EB F0 EC    BEQ $D2D9   verzweige, wenn ja
D2ED 48       PHA         Puffernummer merken
D2EE A9 FF    LDA #$FF
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
D2F7 A2 10    LDX #$10
D2F9 6E 50 02 ROR $0250
D2FC 6E 4F 02 ROR $024F   Belegung in Register eintragen
D2FF 88       DEY
D300 D0 01    BNE $D303
D302 18       CLC
D303 CA       DEX
D304 10 F3    BPL $D2F9
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 Sekundaradresse
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 Sekundaradresse
D317 A6 83    LDX $83     SA 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 in Tabelle
D33E B6 FA     LDX $FA,Y   Kanalnummer holen
D340 B5 A7     LDA $A7,X   zugehörige Puffernummer
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    voraussetzen
D34C AA        TAX         um alternative Puffernummer
D34D B5 A7     LDA $A7,X   zu 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
D367 C9 02     CMP #$02    Fehler aufgetreten ?
```

```
D369 90 08     BCC $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
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
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
D389 0A        ASL         Flag für nächsten Kanal
D38A D0 F7     BNE $D383   und weitermachen
D38C A9 70     LDA #$70    Fehlernummer in A
D38E 4C C8 C1  JMP $C1C8   "70, NO CHANNEL" ausgeben
D391 49 FF     EOR #$FF    Kanal belegen
D393 2D 56 02  AND $0256
D396 8D 56 02  STA $0256   und in Tabelle 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     Kanalnummer
D3AC 20 25 D1  JSR $D125   Filetyp holen
D3AF D0 03     BNE $D3B4   verzweige, wenn keine REL-Datei
D3B1 4C 20 E1  JMP $E120   REL-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
D3D0 95 F2     STA $F2,X   Kanal Status 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   verzweige, wenn nein
D3E8 A9 00     LDA #$00
D3EA 95 99     STA $99,X   Pufferzeiger Lo 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
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  JSR $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    Wert für Error-Puffer-Zeiger Lo
D43C 20 C8 D4  JSR $D4C8   Pufferzeiger setzen
D43F A9 02     LDA #$02
D441 95 9A     STA $9A,X   Pufferzeiger Hi 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 setze ein EOF-Signal, 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 in Tabelle
D452 A9 00     LDA #$00
D454 95 99     STA $99,X   Pufferzeiger Lo auf Null
D456 A1 99     LDA ($99,X) Tracknummer aus Puffer holen
D458 F0 05     BEQ $D45F   verzweige, wenn kein Folgebloch
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
D460 A9 80     LDA #$80     Jobcode für Lesen
D462 D0 02     BNE $D466    unbedingter Sprung
D464                        wird der Kode zum Lesen ($83); bei
                            Einsprung ab $D464 der Kode zum
                            Schreiben ($90) übergeben. Anschließend
                            wird das Kommando ausgeführt.
D464 A9 90     LDA #$90     Jobcode für 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 Disk—Controller
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 gesetzt.
                            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
D492 85 6F     STA $6F      ein 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 holen
D49E 68        PLA          Schrittweite zurückholen
D49F 85 69     STA $69      und abspeichern
```

```
D4A1 A9 00     LDA #$00
D4A3 20 C8 D4  JSR $D4C8   Pufferzeiger auf 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
D4B8 20 C8 D4  JSR $D4C8   Pufferzeiger auf Null setzen
D4BB 20 F1 CF  JSR $CFF1   Puffer mit $00 füllen
D4BE D0 FB     BNE $D4BB
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, auf den 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 in Tabelle
D4CF B5 9A     LDA $9A,X   Pufferzeiger Hi
D4D1 85 95     STA $95     übernehmen
D4D3 A5 6F     LDA $6F     neuen Wert für Pufferzeiger Lo
D4D5 95 99     STA $99,X   übernehmen
D4D7 85 94     STA $94
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 in 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 enthalt 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
D503 B1 71     LDA ($71),Y Byte aus Puffer holen
D505 60        RTS         Ende
-------------------------------------------------------------
D506                       Überprüfen von Spur- und Sektornummer
                           für Jobschleife. X enthält die
                           Puffernummer und $024D den Kode 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
D512 0A        ASL         mal 2
D513 AA        TAX         als Index in Tabelle
D514 B5 07     LDA $07,X   Sektornummer das Jobs
D516 8D 4D 02  STA $024D   merken
D519 B5 06     LDA $06,X   Tracknummer das Jobs
D51B F0 2D     BEQ $D54A   Fehler, wenn Null
D51D CD D7 FE  CMP $FED7   vergleiche mit 36
D520 B0 28     BCS $D54A   Fehler, wenn größer gleich
D522 AA        TAX
D523 68        PLA         Jobcode zurückholen
D524 48        PHA         Stack wiederherstellen
D525 29 F0     AND #$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    Fehlernummer in A
D54F 4C 45 E6 JMP $E645   "66, ILLEGAL TRACK OR SECTOR"
-------------------------------------------------------------
D552                      Holt Track— und Sektornummer für den
                          aktuellen Job.
D552 A5 F9    LDA $F9     aktuelle Puffernummer
D554 0A       ASL         mal 2
D555 AA       TAX         als Index in 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                      Überprüfen auf legale Track— und
                          Sektornummern.
D55F A5 80    LDA $80     Tracknummer
D561 F0 EA    BEQ $D54D   Fehler, wenn Null
D563 CD D7 FE CMP $FED7   mit 36 vergleichen
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 V2.6 1541" 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
D582 9D 5B 02  STA $025B,X
D585 60        RTS         Ende
-------------------------------------------------------------
D586                       Jobkodes an DC übergeben. Bei Ein-
                           Sprung ab $D586 wird der Kode für Lesen
                           ($80); bei Einsprung ab
D586 A9 80     LDA #$80    Code für Lesen
D588 D0 02     BNE $D58C   unbedingter Sprung
D58A                       $D58A der Kode für Schreiben übergeben.
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 DC; Ende
-------------------------------------------------------------
D599                       Wartet 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
D5A1 8D 98 02  STA $0298   Fehlerflag 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
                           Kopfdejustierung, 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    15 (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 für LED holen
D5D6 8D 6D 02  STA $026D   und abspeichern
D5D9 20 A6 D6  JSR $D6A6   $6A (5) Leseversuche 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 Kopf Positionierung
D600 AC 99 02  LDY $0299   Index in Tabelle holen
D603 AD 9A 02  LDA $029A   Byte für Kopf Positionierung holen
D606 38        SEC
```

```
D607 F9 DB FE  SBC $FEDB,Y Wert für Kopfjustage bilden
D60A 8D 9A 02  STA $029A   und merken
D60D B9 DB FE  LDA $FEDB,Y Wert für Positionierung holen
D610 20 76 D6  JSR $D676   Kopf neben den Track positionieren
D613 EE 99 02  INC $0299   Index erhöhen
D616 20 A6 D6  JSR $D6A6   wieder $6A (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 Ende
D625 AD 9A 02  LDA $029A   Wert für Wiederherstellung holen
D628 20 76 D6  JSR $D676   Kopf wieder auf Track positionieren
D62B B5 00     LDA $00,X   Code aus Jobspeicher
D62D C9 02     CMP #$02    Rückmeldung holen
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 erfolgen soll
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 B5 00     LDA $00,X   Rückmeldung
D653 30 FC     BMI $D651   auf Ende des Jobs warten
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
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 zuückholen
D66E 85 7F     STA $7F     und setzen
D670 68        PLA         Index zurückholen
D671 A8        TAY
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<128 bewegt sich
                           der Kopf nach außen; bei A>=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   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
D682 E9 01     SBC #$01    Anzahl der Steps vermindern
D684 D0 F6     BNE $D67C   weitermachen
D686 F0 0A     BEQ $D692   unbedingter Sprung; Ende
D688 A0 FF     LDY #$FF    einen Step nach innen
D68A 20 93 D6  JSR $D693   Kopf bewegen
D68D 18        CLC
D68E 69 01     ADC #$01    Anzahl der Steps erhöhen
D690 D0 F6     BNE $D688   weitermachen
D692 60        RTS         Ende
-------------------------------------------------------------
D693                       Bewegt den Kopf um eine Spur nach innen
                           oder nach 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 DC
D69A D9 FE 02  CMP $02FE,Y Rückmeldung des DC
D69D F0 FB     BEQ $D69A   Ende abwarten
D69F A9 00     LDA #$00
D6A1 99 FE 02  STA $02FE,Y Parameter 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 (normal 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 B5 00     LDA $00,X    Rückmeldung prüfen
D6BB 30 FC     BMI $D6B9    und Ende 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 nocheinmal versuchen
D6C4 48        PHA          Rückmeldung merken
D6C5 AD 6D 02  LDA $026D    Maske für LED am Laufwerk
D6C8 0D 00 1C  ORA $1C00    LED einschalten
D6CB 8D 00 1C  STA $1C00
D6CE 68        PLA          Rückmeldung zurückholen
D6CF 60        RTS          Ende
-------------------------------------------------------------
D6D0                        Parameter für nächsten Job an DC
                            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  in Jobspeicher
D6DF A5 7F     LDA $7F      Drivenummer
D6E1 0A        ASL          mal 2
D6E2 AA        TAX          als Index
D6E3 60        RTS          Ende
-------------------------------------------------------------
D6E4                        Neue Datei im Directory eintragen.
D6E4 A5 83     LDA $83      Sekundäradresse
D6E6 48        PHA          merken
D6E7 A5 82     LDA $82      Kanalnummer
D6E9 48        PHA          merken
```

```
D6EA A5 81     LDA $81     Sektornummer
D6EC 48        PHA         merken
D6ED A5 80     LDA $80     Tracknummer
D6EF 48        PHA         merken
D6F0 A9 11     LDA #$11    17 (READ)
D6F2 85 83     STA $83     internen Lesekanal setzen
D6F4 20 3B DE  JSR $DE3B   Track— und Sektornummer holen
D6F7 AD 4A 02  LDA $024A   Filetyp
D6FA 48        PHA         merken
D6FB A5 E2     LDA $E2     Drivenummer für neues File
D6FD 29 01     AND #$01
D6FF 85 7F     STA $7F     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
D70B 8E 92 02  STX $0292   Zeiger in Directory setzen
D70E 20 AC C5  JSR $C5AC   freien Platz für Eintrag suchen
D711 F0 1D     BEQ $D730   alles voll; neuen Block anlegen
D713 D0 28     BNE $D73D   Eintrag in Directory schreiben
D715 AD 91 02  LDA $0291   Sektornummer des ersten Eintrages
D718 F0 0C     BEQ $D726   verzweige, wenn Null
D71A C5 81     CMP $81     gleich der des neuen Eintrages ?
D71C F0 1F     BEQ $D73D   verzweige, wenn ja
D71E 85 81     STA $81     neuen Sektor anlegen
D720 20 60 D4  JSR $D460   Block lesen
D723 4C 3D D7  JMP $D73D   neuen Eintrag herstellen
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 gefunden
D730 20 8D D4  JSR $D48D   neuen Sektor anlegen
D733 A5 81     LDA $81     Sektornummer
D735 8D 91 02  STA $0291   als Sektor für Directory setzen
D738 A9 02     LDA #$02
D73A 8D 92 02  STA $0292   Zeiger 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 setzen
D747 C9 04     CMP #$04    REL-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   und für Directory merken
D754 20 F1 CF  JSR $CFF1   und 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 in Puffer schreiben
D75E 20 93 DF  JSR $DF93   Puffernummer holen
D761 A8        TAY         als Index
D762 AD 7A 02  LDA $027A   Zeiger in Filetabelle
D765 AA        TAX         nach X
D766 A9 10     LDA #$10    16, Länge des Filenamens
D768 20 6E C6  JSR $C66E   Filename in Puffer schreiben
D76B A0 10     LDY #$10
D76D A9 00     LDA #$00    Fileeintrag ab Position 16 bis
D76F 91 94     STA ($94),Y Position 27 mit Nullen füllen
D771 C8        INY
D772 C0 1B     CPY #$1B
D774 90 F9     BCC $D76F
D776 AD 4A 02  LDA $024A   Filetyp
D779 C9 04     CMP #$04    REL-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 speichern
D79A AD 91 02  LDA $0291
D79D 85 D8     STA $D8     Sektornummer des Fileeintrags im
D79F 9D 60 02  STA $0260,X Directory merken
D7A2 AD 92 02  LDA $0292
D7A5 85 DD     STA $DD     Zeiger auf Fileeintrag im
D7A7 9D 66 02  STA $0266,X Directory merken
D7AA AD 4A 02  LDA $024A   Filetyp
```

```
D7AD 85 E7     STA $E7     in Tabelle schreiben
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, d.h. öffnen einer
                           Datei für LOAD, SAVE oder als
                           Datendatei.
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
D7BF AE 00 02  LDX $0200   erstes Zeichen 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
D7D6 85 E2     STA $E2
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    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
D7FF 20 D1 C1  JSR $C1D1   Befehlsstring analysieren
D802 AD 85 FE  LDA $FE85   18, Directorytrack
D805 85 80     STA $80     als aktuelle Tracknummer setzen
```

```
D807 A9 00     LDA #$00
D809 85 81     STA $81     0; Sektor für BAM
D80B 20 46 DC  JSR $DC46   Puffer belegen; Block lesen
D80E A5 7F     LDA $7F     Drivenummer
D810 09 02     ORA #$02    mit Filetyp SEQ '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 eröffnen
D81C A9 02     LDA #$02    Code für Filetyp PRG
D81E 8D 96 02  STA $0296   setzen
D821 A9 00     LDA #$00
D823 85 7F     STA $7F     Drive 0 setzen
D825 8D 8E 02  STA $028E
D828 20 42 D0  JSR $D042   Drive 0 initialisieren
D82B 20 E5 C1  JSR $C1E5   Befehlsstring nach ':' durchsuchen
D82E D0 04     BNE $D834   verzweige, wenn nicht gefunden
D830 A2 00     LDX #$00
D832 F0 0C     BEQ $D840   unbedingter Sprung
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 ':'
D83D F0 01     BEQ $D840   verzweige, wenn ':' am Anfang
D83F 88        DEY         Y zeigt jetzt auf die Drivenummer
D840 8C 7A 02  STY $027A   Zeiger merken
D843 A9 8D     LDA #$8D    'shift RETURN'
D845 20 68 C2  JSR $C268   Befehlsstring bis Ende analysieren
D848 E8        INX         Anzahl der Kommas 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   Drivenummern prüfen
D852 20 9D C4  JSR $C49D   Eintrag im 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
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
D86A EC 77 02  CPX $0277   Sonderzeichen im String ?
```

```
D86D B0 07     BCS $D876   verzweige, wenn nur ein Zeichen
D86F C0 04     CPY #$04    REL-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 übernehmen
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
D887 AD 4A 02  LDA $024A   Filetyp
D88A D0 1B     BNE $D8A7   verzweige, wenn nicht DEL
D88C A9 02     LDA #$02    Code für PRG-File
D88E 8D 4A 02  STA $024A   übernehmen
D891 AD 4A 02  LDA $024A   Filetyp
D894 D0 11     BNE $D8A7   verzweige, wenn nicht DEL
D896 A5 E7     LDA $E7     Filetyp aus Tabelle holen
D898 29 07     AND #$07    isolieren und
D89A 8D 4A 02  STA $024A   ubernehmen
D89D AD 80 02  LDA $0280   Tracknummer des Files
D8A0 D0 05     BNE $D8A7   verzweige, wenn ungleich Null
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
D8B1 BC 7A 02  LDY $027A,X Zeiger 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 ungleich Null
D8BF A9 01     LDA #$01    Code für WRITE
D8C1 8D 97 02  STA $0297   als Betriebsart setzen
D8C4 D0 B0     BNE $D876   unbedingter Sprung
D8C6 A5 E7     LDA $E7     Filetypparameter aus Tabelle
D8C8 29 80     AND #$80    prüft, ob Datei vorhanden
D8CA AA        TAX
D8CB D0 14     BNE $D8E1   verzweige, wenn ja
D8CD A9 20     LDA #$20
D8CF 24 E7     BIT $E7     Joker im Filenamen ?
D8D1 F0 06     BEQ $D8D9   verzweige, wenn ja
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 Eintrags
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
D8E4 C9 40     CMP #$40    ASCII-Code für '@' (REPLACE)?
D8E6 F0 0D     BEQ $D8F5   verzweige, wenn ja
D8E8 8A        TXA         File 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     REL-File ?
D900 F0 63     BEQ $D965    Fehler, wenn ja
D902 20 DA DC  JSR $DCDA    neuen Sektor anlegen
D905 A5 82     LDA $82      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
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  und merken
D91F A0 1A     LDY #$1A     26
D921 A5 80     LDA $80      aktuelle Tracknummer
D923 91 94     STA ($94),Y  in Fileeintrag schreiben
D925 C8        INY          27
D926 A5 81     LDA $81      aktuelle Sektornummer
D928 91 94     STA ($94),Y  in Fileeintrag schreiben
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  übernehmen
D937 20 3B DE  JSR $DE3B    Track und Sektor setzen
D93A 20 64 D4  JSR $D464    Block schreiben
D93D 4C EF D9  JMP $D9EF    Datei ist jetzt geöffnet
D940 AD 80 02  LDA $0280    Tracknummer des Files
D943 D0 05     BNE $D94A    verzweige, wenn File existiert
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
D953 24 E7     BIT $E7      Datei geöffnet ?
D955 F0 05     BEQ $D95C    verzweige, wenn nein
```

```
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 TYFE MISMATCH" ausgeben
D96A A0 00     LDY #$00
D96C 8C 79 02  STY $0279
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    REL-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 und merken
D980 A5 83     LDA $83     Sekundäradresse
D982 48        PHA         merken
D983 A9 11     LDA #$11    17 (READ)
D985 85 83     STA $83     internen Lesekanal setzen
D987 20 3B DE  JSR $DE3B   Track und Sektor holen
D98A 20 64 D4  JSR $D464   Block schreiben
D98D 68        PLA         Sekundaradresse zurückholen
D98E 85 83     STA $83     und merken
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 äffnen.
D9A0 A0 13     LDY #$13    19
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 Langen 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     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
D9E7 85 7F    STA $7F     und übernehmen
D9E9 20 DA DC JSR $DCDA   Block anlegen
D9EC 20 E4 D6 JSR $D6E4   File im Directory eintragen
D9EF A5 83    LDA $83     Sekundaradresse
D9F1 C9 02    CMP #$02    größer gleich 0 oder 1 ?
D9F3 B0 11    BCS $DA06   verzweige, wenn ja
D9F5 20 3E DE JSR $DE3E   Track und Sektor holen
D9F8 A5 80    LDA $80     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 und Filemodi prüfen
                          und Parameter setzen.
DA09 BC 7A 02 LDY $027A,X Zeiger auf in Befehlszeile
DA0C B9 00 02 LDA $0200,Y Zeichen aus Befehlsstring holen
DA0F A0 04    LDY #$04    Anzahl der Betriebsarten
DA11 88       DEY
```

```
DA12 30 08     BMI $DA1C   weiter
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    Anzahl der Filetypen
DA1E 88        DEY
DA1F 30 08     BMI $DA29   weiter
DA21 D9 B6 FE  CMP $FEB6,Y Filetypen D, S, P, U, 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
DA2F 20 A6 DD  JSR $DDA6   letztes Byte gelesen ?
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 prüfen
DA3A 8A        TXA
DA3B D0 05     BNE $DA42   verzweige, wenn nein
DA3D 20 A3 D1  JSR $D1A3   nächsten Sektor holen
DA40 A9 02     LDA #$02
DA42 20 C8 D4  JSR $D4C8   Pufferzeiger auf 2 setzen
DA45 A6 82     LDX $82     Kanalnummer
DA47 A9 01     LDA #$01    Flag für WRITE
DA49 95 F2     STA $F2,X   in Tabelle setzen
DA4B A9 80     LDA #$80    Flag für WRITE
DA4D 05 82     ORA $82     verbunden mit Kanalnummer
DA4F A6 83     LDX $83     Sekundäradresse als Index
DA51 9D 2B 02  STA $022B,X in Statustabelle eintragen
DA54 60        RTS         Ende
-------------------------------------------------------------
DA55                       Directory laden
DA55 A9 0C     LDA #$0C    12
DA57 8D 2A 02  STA $022A   als Befehlsnummer setzen
DA5A A9 00     LDA #$00
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 (Drivenummer)
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
DA7A 85 E7     STA $E7     gültigen Filetyp setzen
DA7C A9 2A     LDA #$2A    ASCII-Code 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
DA90 88        DEY
DA91 88        DEY
DA92 8C 7A 02  STY $027A   Zeiger ist 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 Directoryzeile 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     Kanalnummer
DAAF 9D 3E 02  STA $023E,X Byte für Ausgabe bereitstellen
DAB2 A5 7F     LDA $7F     Drivenummer
DAB4 8D 8E 02  STA $028E   als letzte Drivenummer merken
DAB7 09 04     ORA #$04    Wert für PRG File
DAB9 95 EC     STA $EC,X   Filetyp in Tabelle setzen
DABB A9 00     LDA #$00
DABD 85 A3     STA $A3     Zeiger in INPUT-PUFFER löschen
DABF 60        RTS         Ende
-------------------------------------------------------------
DAC0                       File mit der geg. Sekundaradresse
                           schließen.
DAC0 A9 00     LDA #$00
DAC2 8D F9 02  STA $02F9   Flag für BAM nicht schreiben
DAC5 A5 83     LDA $83     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 schließen
DAD1 4C DA D4  JMP $D4DA   interne Kanäle schließen; Ende
DAD4 C9 0F     CMP #$0F    15 (Kommmandokanal)?
DAD6 F0 14     BEQ $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
                           Kommandokanal geschlossen werden soll.
DAEC A9 0E     LDA #$0E    14
DAEE 85 83     STA $83     als Sekundäradresse
DAF0 20 02 DB  JSR $DB02   Datei schließen
DAF3 C6 83     DEC $83     nächste Sekundäradresse
DAF5 10 F9     BPL $DAF0   weitermachen, wenn vorhanden
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         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    Directzugriff ?
DB15 F0 0F     BEQ $DB26   verzweige, wenn ja
DB17 C9 04     CMP #$04    REL 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
DB20 20 62 DB  JSR $DB62   Schreiben abschließen
```

```
DB23 20 A5 DB  JSR $DBA5   Eintrag im Directory abschließen
DB26 20 F4 EE  JSR $EEF4   BAM auf Diskette schreiben
DB29 4C 27 D2  JMP $D227   Kanal schließen; Ende
DB2C 20 F1 DD  JSR $DDF1   BAM schreiben, wenn 'dirty'
DB2F 20 1E CF  JSR $CF1E   Zweipufferbetrieb herstellen
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 erhöhen
DB3B A9 00     LDA #$00
DB3D 85 70     STA $70     Zwischenspeicher löschen
DB3F 85 71     STA $71
DB41 A5 D6     LDA $D6     Zeiger in Side—Sektor
DB43 38        SEC
DB44 E9 0E     SBC #$0E    minus 14
DB46 85 72     STA $72     und speichern
DB48 20 51 DF  JSR $DF51   Anzahl der Blöcke berechnen
DB4B A6 82     LDX $82     Kanalnummer
DB4D A5 70     LDA $70
DB4F 95 B5     STA $B5,X   Recordnummer Lo
DB51 A5 71     LDA $71
DB53 95 BB     STA $BB,X   Recordnummer Hi
DB55 A9 40     LDA #$40    Dirty—Flag für Record
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   Recordnummer Hi
DB68 D0 0C     BNE $DB76   verzweige, wenn ungleich Null
DB6A 20 E8 D4  JSR $D4E8   Pufferzeiger holen
DB6D C9 02     CMP #$02    2 (noch kein Byte geschrieben) ?
DB6F D0 05     BNE $DB76   verzweige, wenn nein
DB71 A9 0D     LDA #$0D    'RETURN'
DB73 20 F1 CF  JSR $CFF1   in Puffer schreiben
DB76 20 E8 D4  JSR $D4E8   Pufferzeiger holen
DB79 C9 02     CMP #$02    2 (noch kein Byte geschrieben) ?
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 Lo
```

```
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
DB8C 38        SEC
DB8D E9 01     SBC #$01    Anzahl der Bytes im Block berechnen
DB8F 48        PHA         Wert merken
DB90 A9 00     LDA #$00
DB92 20 C8 D4  JSR $D4C8   Pufferzeiger auf Null setzen
DB95 20 F1 CF  JSR $CFF1   $00 als Folgetrack in Puffer
DB98 68        PLA         Anzahl der Bytes zurückholen
DB99 20 F1 CF  JSR $CFF1   und in Puffer schreiben
DB9C 20 C7 D0  JSR $D0C7   Puffer auf Diskette schreiben
DB9F 20 99 D5  JSR $D599   Ausführung des Jobs abwarten
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; Directorytrack
DBC1 85 80     STA $80     übernehmen
DBC3 20 93 DF  JSR $DF93   Puffernummer holen
DBC6 48        PHA         merken
DBC7 85 F9     STA $F9     und als aktuell übernehmen
DBC9 20 60 D4  JSR $D460   Directorysektor lesen
DBCC A0 00     LDY #$00
DBCE BD E0 FE  LDA $FEE0,X Pufferadresse Hl
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 Filetyp aus Puffer holen
DBDA 29 20     AND #$20    Datei geschlossen "
DBDC F0 43     BEQ $DC21   verzweige, wenn nein
DBDE 20 25 D1  JSR $D125   Filetyp prüfen
```

```
DBE1 C9 04     CMP #$04    REL Datei ?
DBE3 F0 44     BEQ $DC29   verzweige, wenn ja
DBE5 B1 86     LDA ($86),Y Filetyp aus Puffer holen
DBE7 29 8F     AND #$8F    Bits 4, 5 und 6 löschen
DBE9 91 86     STA ($86),Y und wieder abspeichern
DBEB C8        INY
DBEC B1 86     LDA ($86),Y Tracknummer des Files
DBEE 85 80     STA $80     übernehmen
DBF0 84 71     STY $71     Zeiger merken
DBF2 A0 1B     LDY #$1B    27
DBF4 B1 86     LDA ($86),Y Sektornummer beim Überschreiben
DBF6 48        PHA         merken
DBF7 88        DEY         26
DBF8 B1 86     LDA ($86),Y Tracknummer beim Überschreiben
DBFA D0 0A     BNE $DC06   verzweige, wenn vorhanden
DBFC 85 80     STA $80     als Tracknummer übernehmen
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"
DC06 48        PHA         Tracknummer beim Überschreiben
DC07 A9 00     LDA #$00
DC09 91 86     STA ($86),Y Tracknummer 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 neue Tracknummer der Datei setzen
DC13 C8        INY
DC14 B1 86     LDA ($86),Y Sektornummer holen
DC16 85 81     STA $81     und übernehmen
DC18 68        PLA         Sektornummer beim Überschreiben
DC19 91 86     STA ($86),Y als neue Tracknummer 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 als Kennzeichen für
DC27 91 86     STA ($86),Y geschlossenes File setzen
DC29 AE 70 02  LDX $0270   Kanalnummer
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
DC39 A9 90    LDA #$90    Code für Schreiben
DC3B 05 7F    ORA $7F     mit Drivenummer verknüpfen
DC3D 20 90 D5 JSR $D590   und an Disk—Controller
DC40 68       PLA         Sekundäradresse zurückholen
DC41 85 83    STA $83     und wieder übernehmen
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
DC48 20 E2 D1 JSR $D1E2   1 Kanal 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
DC53 05 7F    ORA $7F     mit Drivenummer verknüpfen
DC55 95 EC    STA $EC,X   und in Tabelle ablegen
DC57 20 9B D0 JSR $D09B   Block lesen
DC5A A6 82    LDX $82     Kanalnummer
DC5C A5 80    LDA $80     Tracknummer
DC5E D0 05    BNE $DC65   verzweige, wenn 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    REL 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
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 DC 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
DC9C 95 C1     STA $C1,X   nächster zu bearbeitender Record
DC9E A9 00     LDA #$00
DCA0 20 C8 D4  JSR $D4C8   Pufferzeiger 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 Kanal Status 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
DCBA 0A        ASL         mal 2
DCBB A8        TAY         als Index
DCBC A9 02     LDA #$02
DCBE 99 99 00  STA $0099,Y Pufferzeiger Lo auf $02
DCC1 B5 AE     LDA $AE,X   Puffernummer aus zweiter Tabelle
DCC3 09 80     ORA #$80    Bit 7 setzen; Puffer frei
DCC5 95 AE     STA $AE,X
DCC7 0A        ASL         Puffernummer mal 2
DCC8 A8        TAY         als Index
DCC9 A9 02     LDA #$02
DCCB 99 99 00  STA $0099,Y Pufferzeiger Lo auf $02
DCCE A9 00     LDA #$00
DCD0 95 B5     STA $B5,X   Blockzahl Lo löschen
DCD2 95 BB     STA $BB,X   Blockzahl Hi löschen
DCD4 A9 00     LDA #$00
DCD6 9D 44 02  STA $0244,X Endezeiger löschen
DCD9 60        RTS         Ende
-----------------------------------------------------------------
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
```

```
DCDF 20 DF D1  JSR $D1DF   1 Puffer zum Schreiben suchen
DCE2 20 D0 D6  JSR $D6D0   Parameter an DC übergeben
DCE5 20 B6 DC  JSR $DCB6   Kanal Parameter setzen
DCE8 A6 82     LDX $82     Kanal nummer
DCEA AD 4A 02  LDA $024A   Filetyp
DCED 48        PHA         merken
DCEE 0A        ASL
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    REL 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 Kanal Status 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
DD06 99 2B 02  STA $022B,Y
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—Saktor 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     Kanal nummer
DD2C B5 CD     LDA $CD,X   zugehörige Puffernummer
DD2E 20 D3 D6  JSR $D6D3   Parameter an DC übergeben
DD31 A9 00     LDA #$00
DD33 20 E9 DE  JSR $DEE9   Pufferzeiger auf Null setzen
DD36 A9 00     LDA #$00    0 als Zeiger auf 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
DD57 20 E9 DE  JSR $DEE9   Pufferzeiger auf 16 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
DD6A 20 99 D5  JSR $D599   Jobausführung abwarten und prüfen
DD6D A9 02     LDA #$02
DD6F 20 C8 D4  JSR $D4C8   Pufferzeiger auf 2
DD72 A6 82     LDX $82     Kanalnummer
DD74 38        SEC
DD75 A9 00     LDA #$00
DD77 F5 C7     SBC $C7,X   Recordlänge aus Tabelle
DD79 95 C1     STA $C1,X   nächste zu bearbeitende Nummer
DD7B 20 E2 E2  JSR $E2E2   Records auf Null setzen
DD7E 20 19 DE  JSR $DE19   Linker in Puffer schreiben
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 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.
DD95 90 06     BCC $DD9D   wenn C=0, dann Flags löschen
DD97 A6 82     LDX $82     Kanalnummer
DD99 15 EC     ORA $EC,X   Filetyp Flags setzen
DD9B D0 06     BNE $DDA3
DD9D A6 82     LDX $82     Kanalnummer
DD9F 49 FF     EOR #$FF    invertieren
DDA1 35 EC     AND $EC,X   Filetyp Flags löschen
```

```
DDA3 95 EC     STA $EC,X   neuen 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 Jobkode 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
DDB9 86 71     STX $71     Zähler
DDBB BD 2B 02  LDA $022B,X Kanalstatus holen
DDBE C9 FF     CMP #$FF    Kanal belegt ?
DDC0 D0 08     BNE $DDCA   verzweige, nein ja
DDC2 A6 71     LDX $71     Zähler
DDC4 E8        INX         erhöhen
DDC5 E0 10     CPX #$10    16; höchste Sekundaradresse plus 1
DDC7 90 F0     BCC $DDB9   weitermachen, wenn kleiner
DDC9 60        RTS         Ende; kein aktuelles File gefunden
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
DDD9 B5 E2     LDA $E2,X   Standard für Drivenummer (0)
DDDB 29 01     AND #$01    Drivenummer isolieren
DDDD C5 70     CMP $70     gleich dem Wert in der Tabelle ?
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 mit Sektor der Datei ?
DDE6 D0 DA     BNE $DDC2   verzweige, wenn nein
DDE8 B9 66 02  LDA $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 Änderungen
                          im Puffer erforderlich
                          (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  Block auf Diskette schreiben
DDF9 20 99 D5  JSR $D599  Jobausführung abwarten und prüfen
DDFC 60        RTS        Ende
-------------------------------------------------------------
DDFD                      Linker für folgenden Sektor in Puffer
                          schreiben.
DDFD 20 2B DE  JSR $DE2B  Pufferzeiger setzen
DE00 A5 80     LDA $80    Tracknummer
DE02 91 94     STA ($94),Y in Puffer schreiben
DE04 C8        INY        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 für Spur und Sektor des nächsten
                          Sektors aus Puffer holen.
DE0C 20 2B DE  JSR $DE2B  Pufferzeiger setzen
DE0F B1 94     LDA ($94),Y Linker für Folgetrack
DE11 85 80     STA $80    als Tracknummer übernehmen
DE13 C8        INY
DE14 B1 94     LDA ($94),Y Linker für nächsten Sektor
DE16 85 81     STA $81    als Sektornummer übernehmen
DE18 60        RTS        Ende
-------------------------------------------------------------
DE19                      Linker für den letzten Block der Datei
                          setzen (Spur = $00).
DE19 20 2B DE  JSR $DE2B  Pufferzeiger setzen
DE1C A9 00     LDA #$00
DE1E 91 94     STA ($94),Y Tracknummer gleich 0 setzen
DE20 C8        INY
DE21 A6 82     LDX $82    Kanalnummer
DE23 B5 C1     LDA $C1,X  Endekennzeichen holen
DE25 AA        TAX
DE26 CA        DEX        minus 1
DE27 8A        TXA
DE28 91 94     STA ($94),Y gleich Anzahl der Bytes im Block
```

```
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
DE36 85 94    STA $94     Pufferzeiger Lo = $00
DE38 A0 00    LDY #$00
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 merken
DE43 0A       ASL         mal 2
DE44 A8       TAY         als Index
DE45 B9 06 00  LDA $0006,Y Tracknummer für Job
DE48 85 80    STA $80     übernehmen
DE4A B9 07 00  LDA $0007,Y Sektornummer  für Job
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
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 DC übergeben
DE82 20 93 DF  JSR $DF93   Puffernummer holen
DE85 AA        TAX         Puffernummer
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
DE92 4C 06 D5  JMP $D506   Befehlscode prüfen und an DC; Ende
-------------------------------------------------------------
DE95                       Nächsten Sektor der Datei anhand des
                           Linkers im Puffer feststellen.
DE95 A9 00     LDA #$00
DE97 20 C8 D4  JSR $D4C8   Pufferzeiger auf Null setzen
DE9A 20 37 D1  JSR $D137   Byte aus Puffer holen
DE9D 85 80     STA $80     als Tracknummer übernehmen
DE9F 20 37 D1  JSR $D137   Byte aus Puffer holen
DEA2 85 81     STA $81     als Sektornummer übernehmen
DEA4 60        RTS         Ende
-------------------------------------------------------------
DEA5                       Pufferinhalte aus einem Puffer in einen
                           anderen übertragen. A enthalt 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
DEA8 85 6F     STA $6F     Pufferadressen Lo setzen
DEAA 85 71     STA $71
DEAC B9 E0 FE  LDA $FEE0,Y Adresse des Quellpuffers Hi holen
DEAF 85 70     STA $70     und merken
DEB1 BD E0 FE  LDA $FEE0,X Adresse des Zielpuffers Hi holen
DEB4 85 72     STA $72     und merken
DEB6 68        PLA         Anzahl der Bytes zurückholen
DEB7 A8        TAY         und als Zähler
DEB8 88        DEY
DEB9 B1 6F     LDA ($6F),Y Byte aus Quellpuffer holen
DEBB 91 71     STA ($71),Y und in Zielpuffer abspeichern
```

```
DEBD 88       DEY         nächstes Byte
DEBE 10 F9    BPL $DEB9
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
DEC5 85 70    STA $70     setzen
DEC7 A9 00    LDA #$00    Pufferadresse Lo
DEC9 85 6F    STA $6F     setzen
DECB A8       TAY         Puffernummer
DECC 91 6F    STA ($6F),Y $00 in Puffer schreiben
DECE C8       INY
DECF D0 FB    BNE $DECC
DED1 60       RTS         Ende
-------------------------------------------------------------
DED2                      Side—Sektor Nummer holen.
DED2 A9 00    LDA #$00
DED4 20 DC DE JSR $DEDC   Pufferzeiger auf Null setzen
DED7 A0 02    LDY #$02    Byte 2
DED9 B1 94    LDA ($94),Y Side—Sektor Nummer in 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
DEE6 85 95    STA $95     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 setzen
DF17 2C CD FE  BIT $FECD   V-Flag löschen
DF1A 60        RTS         Ende; Status ok
-------------------------------------------------------------
DF1B                       Routine zur Übergabe von Jobs an den
                           DC. Bei Einsprung in $DF1B wird
                           gelesen; bei Einsprung über $DF21
DF1B 85 F9     STA $F9     Puffernummer merken
DF1D A9 80     LDA #$80    Jobcode für Lesen eines Blocks
DF1F D0 04     BNE $DF25   unbedingter Sprung
DF21                       wird geschrieben. A muß die Nummer des
                           Puffers zum Schreiben/Lesen und X die
                           Nummer des aktiven Puffers enthalten.
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 merken
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
```

```
DF37 B1 94    LDA ($94),Y Folgesektor aus Puffer holen
DF39 85 81    STA $81     und merken
DF3B A5 F9    LDA $F9     Puffernummer
DF3D 20 D3 D6 JSR $D6D3   Parameter an DC übergeben
DF40 A6 F9    LDX $F9     Puffernummer
DF42 4C 93 D5 JMP $D593   Jobcode an DC übergeben
-----------------------------------------------------------
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 Blocke einer relativen
                          Datei berechnen.
DF4C A9 78    LDA #$78    120 (Anzahl der Blockzeiger/SS)
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         durch 2 (da 2 Bytes/Linker)
DF57 20 5C DF JSR $DF5C   Anzahl wieder addieren
DF5A A5 73    LDA $73     Zahl der belegten Side-Sektoren
DF5C 18       CLC
DF5D 65 70    ADC $70
DF5F 85 70    STA $70     addieren
DF61 90 02    BCC $DF65
DF63 E6 71    INC $71
DF65 60       RTS         Ende
-----------------------------------------------------------
DF66                      Zustand eines Side-Sektcr-Blocks im
                          Puffer prüfen.
DF66 20 D2 DE JSR $DED2   Nummer des Side-Sektors holen
DF69 C5 D5    CMP $D5     identisch mit SS 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 Folgetrack
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-Sektar
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 $04 setzen
DF87 B1 94    LDA ($94),Y Tracknummer holen
DF89 D0 04    BNE $DF8F   verzweige, wenn vorhanden
DF8B 2C D0 FE BIT $FED0   N- und V-Flag setzen
DF8E 60       RTS         Ende; Fehlerstatus
DF8F 2C CE FE BIT $FECE   V—Flag setzen
DF92 60       RTS         Ende; SS 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 $AE,X   Puffer aktive oder nur belegt ?
DF9B 29 BF    AND #$BF    testen
DF9D 60       RTS         Ende
-------------------------------------------------------------
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
DFA9 69 07    ADC #$07    plus 7
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 prüfen
DFB4 24 70    BIT $70
DFB6 60       RTS         Ende
-------------------------------------------------------------
DFB7                      Nummer eines inaktiven Puffers 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
DFC1 60       RTS         Ende
-------------------------------------------------------------
DFC2                      Puffer freigeben oder inaktivieren
                          Puffernummer in A
```

```
DFC2 A6 82     LDX $82     Kanalnummer
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     Kanalnummer
DFDE F6 B5     INC $B5,X   Recordnummer Lo plus 1
DFE0 D0 02     BNE $DFE4   verzweige, wenn ungleich Null
DFE2 F6 BB     INC $BB,X   Recordnummer 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     Kanalnummer
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     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
E00E 20 F6 D4  JSR $D4F6   Folgetrack aus Puffer holen
E011 D0 21     BNE $E034   verzweige, wenn Block vorhanden
E013 68        PLA         Zeiger zurückholen
E014 C9 02     CMP #$02
```

```
E016 F0 12    BEQ $E02A   verzweige, wenn Ergebnis gleich 2
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
E031 95 C1    STA $C1,X   Zeiger auf nächsten Record = 0
E033 60       RTS         Ende
E034 68       PLA         Zeiger
E035 A6 82    LDX $82     Kanalnummer
E037 95 C1    STA $C1,X   als Zeiger auf nächsten Record
E039 4C 6E E1 JMP $E16E   Zeiger auf letztes Zeichen; Ende
-----------------------------------------------------------------
```

```
E03C                    Nächsten Record in Puffer generieren;
                        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' ?
E045 50 16     BVC $E05D verzweige, wenn nein
E047 20 5E DE  JSR $DE5E Puffer auf Diskette schreiben
E04A 20 1E CF  JSR $CF1E Fuffer wechseln
E04D A9 02     LDA #$02
E04F 20 C8 D4  JSR $D4C8 Pufferzeiger auf $02 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 Recordpuffer schreiben.
E07C 20 05 E1  JSR $E105 Puffer 'dirty' setzen, da 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 vom seriellen Bus
E086 81 99     STA ($99,X) in 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  Pufferzeiger neu setzen
E096 98        TYA
E097 A4 82     LDY $82   Kanalnummer
E099 D9 C1 00  CMP $00C1,Y Zeiger 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 plus 1
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   testen
E0B0 D0 27     BNE $E0D9   verzweige, wenn eines gesetzt
E0B2 A5 85     LDA $85     Byte vom seriellen 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, OVERFLOW 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   weitermachen, wenn nein
E0E1 60        RTS         Ende
E0E2 A5 85     LDA $85     Byte vom seriellen 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
E0FC 85 85     STA $85     Datenbyte gleich $00
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 anzeigen, daß der Pufferinhalt
                           aktualisiert wurde, so daß er mit dem
                           Block auf Diskette nicht mehr
                           übereinstimmt ('dirty' Puffer)
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,
                           daß die Änderungen an einem Puffer
                           anzeigte ('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   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 plus 1
E133 D0 06     BNE $E13B   verzweige, wenn ungleich Null
E135 20 3C E0  JSR $E03C   Record schreiben; nächsten 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     Byte 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 ist
E169 A9 50     LDA #$50    Nummer der Fehlermeldung
E16B 20 C8 C1  JSR $C1C8   "53, 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
E178 D0 04     BNE $E17E   verzweige, wenn ungleich 2
E17A A9 FF     LDA #$FF
E17C 85 87     STA $87     Zeiger auf $FF setzen
E17E B5 C7     LDA $C7,X   Recordlänge
E180 85 88     STA $88     merken
E182 20 E8 D4  JSR $D4E8   Pufferzeiger entsprechend setzen
E185 A6 82     LDX $82     Kanalnummer
E187 C5 87     CMP $87     Pufferzeiger größer als R—Zeiger 7
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
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
E1A2 85 87     STA $87     Recordzeiger auf $FF
E1A4 20 B2 E1  JSR $E1B2   Byte an Zeigerposition aus Puffer
E1A7 B0 03     BCS $E1AC   verzweige, wenn nicht letztes Byte
E1A9 20 E8 D4  JSR $D4E8   Pufferzeiger neu setzen
E1AC A6 82     LDX $82     Kanalnununer
E1AE 9D 44 02  STA $0244,X Endezeiger setzen
E1B1 60        RTS         Ende
-------------------------------------------------------------
E1B2                       Letztes Byte ungleich Null 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         fertig
E1C8 98        TYA         Recordzeiger
E1C9 38        SEC         Flag für Ende noch nicht gefunden
E1CA 60        RTS         zurück
-------------------------------------------------------------
E1CB                       Ende des letzten Records feststellen
                           und Zeiger entsprechend setzen.
E1CB 20 D2 DE  JSR $DED2   Side—Sektor Nummer holen
E1CE 85 D5     STA $D5     und merken
E1D0 A9 04     LDA #$04
E1D2 85 94     STA $94     Zeiger in Puffer Lo
E1D4 A0 0A     LDY #$0A
E1D6 D0 04     BNE $E1DC   unbedingter Sprung
E1D8 88        DEY
E1D9 88        DEY
```

```
E1DA 30 26     BMI $E202   Fehler, wenn Y < 0
E1DC B1 94     LDA ($94),Y Tracknummer des vorherigen Blocks
E1DE F0 F8     BEQ $E1D8   verzweige, wenn keiner vorhanden
E1E0 98        TYA
E1E1 4A        LSR         geteilt 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 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
E1F1 84 94     STY $94     Pufferzeiger gleich Null
E1F3 B1 94     LDA ($94),Y Tracknummer des Folgeblocks
E1F5 D0 0B     BNE $E202   verzweige, wenn Block folgt
E1F7 C8        INY
E1F8 B1 94     LDA ($94),Y Sektornummer jetzt Anzahl der Bytes
E1FA A8        TAY
E1FB 88        DEY
E1FC 84 D6     STY $D6     als Endezeiger merken
E1FE 98        TYA
E1FF 4C E9 DE  JMP $DEE9   Pufferzeiger setzen; Ende
E202 A9 67     LDA #$67    Nummer der Fehlermeldung
E204 20 45 E6  JSR $E645   "67, ILLEGAL TRACK OR SECTOR"
-------------------------------------------------------------
E207                       POSITION-Befehl
E207 20 B3 C2  JSR $C2B3   Befehlsstring prüfen
E20A AD 01 02  LDA $0201   zweites Zeichen aus Befehlsstring
E20D 85 83     STA $83     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 REL 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 Zeichern aus Befehlsstring
E231 95 B5     STA $B5,X   als Recordnummer Lo merken
```

```
E233 AD 03 02  LDA $0203   viertes Zeichen aus Befehlsstring
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
E241 F0 10     BEQ $E253   verzweige, wenn Null
E243 38        SEC
E244 E9 01     SBC #$01    teste, ob Wert gleich 1
E246 F0 0B     BEQ $E253   verzweige, wenn ja
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
E253 85 D4     STA $D4     Zeiger auf Beginn des Record
E255 20 0E CE  JSR $CE0E   Zeiger in REL 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" ausgeben
E272 4C 94 C1  JMP $C194   Diskstatus bereitstellen; Ende
-------------------------------------------------------------
E275                       Record in aktuellen Puffer holen und
                           den nächsten Block gleich in den
                           zweiten Puffer zur späteren 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
E282 E5 D4     SBC $D4     minus Beginn des Record
E284 B0 03     BCS $E289   verzweige, wenn Ergebnis positiv
E286 4C 02 E2  JMP $E202   "67, ILLEGAL TRACK OR SECTOR"
E289 18        CLC
E28A 65 D7     ADC $D7     Zeiger in Record addieren
```

```
E28C 90 03     BCC $E291   verzweige, wenn kleiner gleich 255
E28E 69 01     ADC #$01    Überlauf plus 1 addieren
E290 38        SEC
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                       Datenblöcke 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 Folgablock
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
E2C4 B1 89     LDA ($89),Y Tracknummer
E2C6 85 80     STA $80     übernehmen
E2C8 C8        INY
E2C9 B1 89     LDA ($89),Y Sektornummer des Folgeblocks
E2CB 85 81     STA $81     ebenfalls 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
E2D3 A0 00     LDY #$00
E2D5 B1 89     LDA ($89),Y Tracknummer
E2D7 C5 80     CMP $80     vergleichen
E2D9 F0 01     BEQ $E2DC   verzweige, wenn gleich
E2DB 60        RTS         Ende
```

```
E2DC C8       INY
E2DD B1 89    LDA ($89),Y Sektornummer
E2DF C5 81    CMP $81     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
E2E7 A9 00    LDA #$00
E2E9 91 94    STA ($94),Y Puffer löschen
E2EB C8       INY
E2EC D0 FB    BNE $E2E9
E2EE 20 04 E3 JSR $E304   Position des nächsten R berechnen
E2F1 95 C1    STA $C1,X   und merken
E2F3 A8       TAY
E2F4 A9 FF    LDA #$FF
E2F6 91 94    STA ($94),Y $FF als Kennzeichen in Record
E2F8 20 04 E3 JSR $E304   Zeiger auf nächsten R berechnen
E2FB 90 F4    BCC $E2F1   verzweige, wenn R in diesem Block
E2FD D0 04    BNE $E303   verzweige, wenn Block voll
E2FF A9 00    LDA #$00
E301 95 C1    STA $C1,X   Recordzeiger löschen
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
E308 38       SEC
E309 F0 0D    BEQ $E318   verzweige, wenn Zeiger gleich Null
E30B 18       CLC
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
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 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
E332 85 88     STA $88     Flag für einen Block löschen
E334 A9 00     LDA #$00
E336 85 D4     STA $D4     Zeiger auf Beginn des Record
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
E344 18        CLC
E345 65 D7     ADC $D7     plus Zeiger in Record
E347 90 0C     BCC $E355   verzweige, wenn kein Überlauf
E349 E6 D6     INC $D6     Zeiger in Side-Sektor erhöhen
E34B E6 D6     INC $D6     2 mal, da Track und Sektor
E34D D0 06     BNE $E355   verzweige, wenn kein Übertrag
E34F E6 D5     INC $D5     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     vorheriger Zeiger in Side—Sektor
E357 18        CLC
E358 69 02     ADC #$02    plus 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
E36B E5 87     SBC $87     minus letztem Zeiger in SS
E36D B0 03     BCS $E372   verzweige, wenn kein Borrow
E36F E9 0F     SBC #$0F    minus 16
E371 18        CLC
E372 85 72     STA $72     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
E37C 86 70     STX $70     Summenspeicher löschen
E37E 86 71     STX $71
E380 AA        TAX         Differenz nach X
E381 20 51 DF  JSR $DF51   Blockzahl der REL Datei berechnen
E384 A5 71     LDA $71     Blochzahl Hi
E386 D0 07     BNE $E38F   verzweige, wenn ungleich Null
E388 A6 70     LDX $70     Blockzahl Lo
E38A CA        DEX         testen, ob Zahl gleich 1
E38B D0 02     BNE $E38F   verzweige, wenn nein
E38D E6 88     INC $88     Flag für einen Block setzen
E38F CD 73 02  CMP $0273   Anzahl der Blocks auf Diskette Hi
E392 90 09     BCC $E39D   verzweige, wenn REL Datei kleiner
E394 D0 CD     BNE $E363   Fehler, wenn Datei größer
E396 AD 72 02  LDA $0272   Blockzahl auf Diskette Lo
E399 C5 70     CMP $70     vergleichen mit Zahl der REL Datei
E39B 90 C6     BCC $E363   verzweige, wenn REL Datei größer
E39D A9 01     LDA #$01
E39F 20 F6 D4  JSR $D4F6   zweites Byte aus Puffer holen
E3A2 18        CLC
E3A3 69 01     ADC #$01    plus 1
E3A5 A6 82     LDX $82     Kanalnummer
E3A7 95 C1     STA $C1,X   als Recordzeiger setzen
E3A9 20 1E F1  JSR $F11E   freien Block in BAM suchen
E3AC 20 FD DD  JSR $DDFD   Track und Sektor in Puffer
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 DC übergeben
E3BC 20 1E F1  JSR $F11E   freien Block in 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
E3C8 20 1E CF  JSR $CF1E   Puffer wechseln
E3CB 20 D0 D6  JSR $D6D0   Parameter an DC 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    SS schreiben; nächsten SS anlegen
E3F2 A9 10    LDA #$10
E3F4 20 E9 DE JSR $DEE9    Pufferzeiger auf 16 setzen
E3F7 E6 86    INC $86      Side—Sektor Nummer plus 1
E3F9 68       PLA          Tracknummer zurückholen
E3FA 20 8D DD JSR $DD8D    und in SS eintragen
E3FD 68       PLA          Sektornummer zurückholen
E3FE 20 8D DD JSR $DD8D    und in SS eintragen
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 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 SS setzen
E412 C5 D6    CMP $D6      vergleiche mit SS—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 SS setzen
E41B 48       PHA          als Endezeiger merken
E41C A9 00    LDA #$00
E41E 20 DC DE JSR $DEDC    Pufferzeiger auf Null setzen
E421 A9 00    LDA #$00
E423 A8       TAY
E424 91 94    STA ($94),Y  Null als Folgetrack in Puffer
E426 C8       INY
E427 68       PLA          Endezeiger zurückholen
E428 38       SEC
E429 E9 01    SBC #$01     minus 1 gleich Anzahl der Bytes
E42B 91 94    STA ($94),Y  in Puffer schreiben
E42D 20 6C DE JSR $DE6C    Block auf Diskette schreiben
```

```
E430 20 99 D5  JSR $D599   Jobausfuhrung abwarten und prüfen
E433 20 F4 EE  JSR $EEF4   BAM neu schreiben
E436 20 0E CE  JSR $CE0E   Zeiger für REL Datei neu setzen
E439 20 1E CF  JSR $CF1E   Puffer wechseln
E43C 20 F8 DE  JSR $DEF8   richtiger SS 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 Elock in der BAM suchen
E451 20 1E CF  JSR $CF1E   Puffer wechseln
E454 20 F1 DD  JSR $DDF1   alten Side—Sektar 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 SS
E462 A8        TAY
E463 68        PLA         merken
E464 AA        TAX
E465 A9 10     LDA #$10    16 Bytes des alten Side—Sektors
E467 20 A5 DE  JSR $DEA5   in neuen SS übernehmen
E46A A9 00     LDA #$00
E46C 20 DC DE  JSR $DEDC   Pufferzeiger auf Null
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
E476 20 C8 D4  JSR $D4C8   Pufferzeiger auf Null (neuer SS)
E479 68        PLA         Side-Sektor Nummer zurückholen
E47A 18        CLC
E47B 69 01     ADC #$01    plus 1
E47D 91 94     STA ($94),Y als Nummer des neuen Side Sektors
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
E485 38        SEC
E486 E9 02     SBC #$02    minus 2
```

```
E488 85 8A     STA $8A     Zeiger auf alten Side—Sektor
E48A A5 80     LDA $80     Tracknummer
E48C 85 87     STA $87     merken
E48E 91 94     STA ($94),Y und in Puffer schreiben
E490 C8        INY
E491 A5 81     LDA $81     Sektornummer
E493 85 88     STA $88     merken
E495 91 94     STA ($94),Y und in Puffer schreiben
E497 A0 00     LDY #$00
E499 98        TYA
E49A 91 94     STA ($94),Y Marke für letzten Side-Sektor
E49C C8        INY
E49D A9 11     LDA #$11    17
E49F 91 94     STA ($94),Y als Anzahl der Bytes im Block
E4A1 A9 10     LDA #$10
E4A3 20 C8 D4  JSR $D4C8   Pufferzeiger auf 16 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 des SS
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 SS 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
E4C0 20 C8 D4  JSR $D4C8   Pufferzeiger auf Null setzen
E4C3 A0 00     LDY #$00
E4C5 A5 80     LDA $80     Tracknummer
E4C7 91 94     STA ($94),Y in Puffer schreiben
E4C9 C8        INY
E4CA A5 81     LDA $81     Sektornummer
E4CC 91 94     STA ($94),Y in Puffer schreiben
E4CE 4C DE E4  JMP $E4DE
E4D1 20 93 DF  JSR $DF93   Puffernummer holen
E4D4 A6 82     LDX $82     Kanalnummer
E4D6 20 1B DF  JSR $DF1B   nächsten SS von Diskette lesen
E4D9 A9 00     LDA #$00
E4DB 20 C8 D4  JSR $D4C8   Pufferzeiger auf Null 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
E4E9 A5 88    LDA $88     Sektornummer holen
E4EB 91 94    STA ($94),Y und in Puffer schreiben
E4ED 20 5E DE JSR $DE5E   SS auf Diskette schreiben
E4F0 20 99 D5 JSR $D599   Jobausführung abwarten und prüfen
E4F3 A4 8A    LDY $8A     Zähler
E4F5 C0 03    CPY #$03
E4F7 B0 D8    BCS $E4D1   verzweige, wenn größer gleich 3
E4F9 4C 1E CF JMP $CF1E   Puffer wechseln
-------------------------------------------------------------
E4FC                      Tabelle mit den ASCII-Codes aller
                          Fehlermeldungen des DOS.
E4FC 00 A0 4F CB              00, oK
E500 20 21 22 23 24 27 D5 45  20/21/22/23/24/27, Read ErroR
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 4F CE
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 sectoR
E578 4C 20 54 52 41 43 4B 20
E580 4F 52 20 53 45 43 54 4F
E588 D2
E589 61 83 06 84              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 45 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 v2.6 1541
E5BE 20 56 32 2E 36 20 31 35
E5C6 34 B1
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 48       PHA         Fehlernummer merken
E60B 86 F9    STX $F9     Puffernummer merken
E60D 8A       TXA
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 merken
E614 B5 07    LDA $07,X   Sektornummer aus Jobspeicher holen
E616 85 81    STA $81     und ebenfalls merken
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    Fehlernummer 15 ?
E61F D0 06    BNE $E627   verzweige, wenn nein
E621 A9 74    LDA #$74    Nummer der Fehlermeldung
E623 D0 08    BNE $E62D   unbedingter Sprung: DRIVE NOT READY
E625 A9 06    LDA #$06    6
E627 09 20    ORA #$20    plus $20
E629 AA       TAX
E62A CA       DEX
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 auszuführenden Befehls
E631 C9 00    CMP #$00    VALIDATE ?
E633 D0 0F    BNE $E644   verzweige, wenn nein
E635 A9 FF    LDA #$FF
E637 8D 2A 02 STA $022A   Befehlsnummer 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
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
E658 85 A3    STA $A3     Zeiger in INPUT-Puffer löschen
E65A A2 45    LDX #$45
E65C 9A       TXS         Stackpointer 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    15; Kommandokanal ?
E665 F0 31    BEQ $E698   verzweige, wenn ja
E667 78       SEI
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 $022B,X zugehöriger Kanal Status
E675 C9 FF    CMP #$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 20 4E EA JSR $EA4E   Bus freimachen; zur Warteschleife
E686 D0 06    BNE $E68E   wird hier nicht ausgeführt
```

```
------------------------------------------------------------
E688                        Fehlerbehandlung bei LISTEN.
E688 20 07 D1  JSR $D107    Kanal zum Schreiben suchen
E68B 20 4E EA  JSR $EA4E    Bus freimachen; zur Warteschleife
E68E 20 25 D1  JSR $D125
E691 C9 04     CMP #$04     bei der VC 1541 nicht benutzt !
E693 B0 03     BCS $E698
E695 20 27 D2  JSR $D227
E698 4C E7 EB  JMP $EBE7
------------------------------------------------------------
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 der
                            Umwandlung.
E69B AA        TAX
E69C A9 00     LDA #$00
E69E F8        SED          auf Dezimalrechnung umschalten
E69F E0 00     CPX #$00     Hex-Byte gleich Null ?
E6A1 F0 07     BEQ $E6AA    verzweige, wenn ja
E6A3 18        CLC
E6A4 69 01     ADC #$01
E6A6 CA        DEX          minus 1
E6A7 4C 9F E6  JMP $E69F    weitermachen
E6AA D8        CLD          auf Hexadezimalrechnung umschalten
E6AB AA        TAX
E6AC 4A        LSR          Hi-Nybble; Wert holen
E6AD 4A        LSR
E6AE 4A        LSR
E6AF 4A        LSR
E6B0 20 B4 E6  JSR $E6B4    in ASCII-Wert umwandeln
E6B3 8A        TXA
E6B4 29 0F     AND #$0F     Lo—Nybble; Wert holen
E6B6 09 30     ORA #$30     in ASCII-Zahl umwandeln
E6B8 91 A5     STA ($A5),Y und in Puffer schreiben
E6BA C8        INY          Pufferzeiger plus 1
E6BB 60        RTS          Ende
------------------------------------------------------------
E6BC                        Fehlermeldung in Fehlerpuffer
                            schreiben.
E6BC 20 23 C1  JSR $C123    Bei Einsprung ab $E6BC wird die
E6BF A9 00     LDA #$00     'OK'—Meldung im Puffer generiert.
E6C1 A0 00     LDY #$00
E6C3 84 80     STY $80      Track = 0
```

```
E6C5 84 81     STY $81     Sektor = 0
E6C7 A0 00     LDY #$00    A muß die Fehlernummer enthalten
E6C9 A2 D5     LDX #$D5    Lo-Byte für ERROR-Puffer
E6CB 86 A5     STX $A5     setzen
E6CD A2 02     LDX #$02    Hi-Byte für ERROR-Puffer
E6CF 86 A6     STX $A6     setzen
E6D1 20 AB E6  JSR $E6AB   Fehlernummer als ASCII in Puffer
E6D4 A9 2C     LDA #$2C    ',' Komma
E6D6 91 A5     STA ($A5),Y in Puffer schreiben
E6D8 C8        INY
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 schreiben
E6E7 C8        INY
E6E8 A5 80     LDA $80     Tracknummer für Fehlermeldung
E6EA 20 9B E6  JSR $E69B   nach ASCII und in Puffer
E6ED A9 2C     LDA #$2C    ',' Komma
E6EF 91 A5     STA ($A5),Y in Puffer schreiben
E6F1 C8        INY
E6F2 A5 81     LDA $81     Sektornummer für Fehlermeldung
E6F4 20 9B E6  JSR $E69B   nach ASCII und in Puffer
E6F7 88        DEY
E6F8 98        TYA
E6F9 18        CLC
E6FA 69 D5     ADC #$D5    Länge der Fehlermeldung setzen
E6FC 8D 49 02  STA $0249   und abspeichern
E6FF E6 A5     INC $A5     Pufferadresse Lo auf 2. Byte setzen
E701 A9 88     LDA #$88    READY TO TALK Status setzen
E703 85 F7     STA $F7
E705 60        RTS         Ende
-------------------------------------------------------------
E706                       Schreibt den Text für die Fehlermeldung
                           aus der ASCII-Tabelle in den
                           Fehlerpuffer.
E706 AA        TAX         Fehlernummer
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
E716 A2 00    LDX #$00
E718 C1 86    CMP ($86,X) Fehlernummer in 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   warten auf entsprechendes Byte
E727 A5 87    LDA $87     Zeiger in Tabelle Hi
E729 C9 E6    CMP #$E6    mit $E6 vergleichen
E72B 90 08    BCC $E735   verzweige, wenn kleiner
E72D D0 0A    BNE $E739   verzweige, wenn gleich
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
E73D 20 67 E7 JSR $E767   Zeichen holen; Bit 7 ins Carry
E740 90 FB    BCC $E73D   warten auf Bit 7 gesetzt
E742 20 54 E7 JSR $E754   Zeichen in ERROR-Puffer schreiben
E745 20 67 E7 JSR $E767   Zeichen holen; Bit 7 ins Carry
E748 90 F8    BCC $E742   weitermachen bis zum letzten Byte
E74A 20 54 E7 JSR $E754   letztes Zeichen in Puffer
E74D 68       PLA         Wert zurückholen
E74E 85 87    STA $87     und abspeichern
E750 68       PLA         Wert zurückholen
E751 85 86    STA $86     und abspeichern
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 ERROR-Puffer schreiben
```

```
E75D C8       INY
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 ERROR—Puffer schreiben
E765 C8       INY
E766 60       RTS          Ende
-------------------------------------------------------------
E767                       Zeiger in Tabelle erhöhen und Byte aus
                           der Tabelle holen. Bit 7 dieses Bytes
                           ins Carry holen.
E767 E6 86    INC $86      Zeiger Lo erhöhen
E769 D0 02    BNE $E76D    verzweige, wenn Null erreicht
E76B E6 87    INC $87      Zeiger Hi erhöhen
E76D A1 86    LDA ($86,X)  Zeichen aus Tabelle holen
E76F 0A       ASL          Bit 7 ins Carry schieben
E770 A1 86    LDA ($86,X)  Zeichen aus Tabelle
E772 29 7F    AND #$7F     Bit 7 löschen
E774 60       RTS          Ende
-------------------------------------------------------------
E775                       Fehlerzeiger erhöhen.
E775 20 6D E7 JSR $E76D    Byte aus Tabelle; Bit 7 ins Carry
E778 E6 86    INC $86      Zeiger Lo erhöhen
E77A D0 02    BNE $E77E    verzweige, wenn Null erreicht
E77C E6 87    INC $87      Zeiger Hi erhöhen
E77E 60       RTS          Ende
E77F 60       RTS          Ende
-------------------------------------------------------------
E780                       AUTOBOOT Routine. Diese Routine wurde
                           in Floppies der neueren Generation, die
                           mit neuen ROMs der Endnummer —05
                           ausgestattet sind, nicht mehr
                           implementiert, da beim Einschalten der
                           Floppy Probleme auftreten können. ROMs
                           der Endnummer —03 enthalten diese
                           Routine noch.
E780 AD 00 18 LDA $1800    Zustand des Bus abfragen
E783 AA       TAX
E784 29 04    AND #$04     CLOCK IN isolieren
E786 F0 F7    BEQ $E77F    verzweige, wenn Bit = 0
E788 8A       TXA          Zustand des Bus zurückholen
E789 29 01    AND #$01     DATA IN isolieren
E78B F0 F2    BEQ $E77F    verzweige, wenn Bit = 0
```

```
E78D 58       CLI
E78E AD 00 18 LDA $1800   Zustand des Bus abfragen
E791 29 05    AND #$05    DATA IN und CLOCK IN isolieren
E793 D0 F9    BNE $E78E   warten bis ein Bit = 1 ist
E795 EE 78 02 INC $0278   Dateiname im INPUT-Puffer
E798 EE 74 02 INC $0274   Länge des Namens 1 Zeichen
E79B A9 2A    LDA #$2A    ASCII-Code für '*'
E79D 8D 00 02 STA $0200   als Filename abspeichern
E7A0 4C A8 E7 JMP $E7A8   Programm laden und starten
-------------------------------------------------------------
E7A3                      &-Befehl
E7A3 A9 8D    LDA #$8D    'Shift RETURN'
E7A5 20 68 C2 JSR $C268   Befehlszeile bis Ende analysieren
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
E7B1 8D 78 02 STA $0278   Anzahl ist jetzt 1
E7B4 A9 FF    LDA #$FF
E7B6 85 86    STA $86
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 abspeichern
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
E7DA 85 87    STA $87     Prüfsumme mit Standard vorbesetzen
E7DC 20 39 E8 JSR $E839   Byte aus Puffer holen
E7DF 85 88    STA $88     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     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 setzen, daß die Startadresse
E7F8 85 86    STA $86     schon geholt worden ist
E7FA 20 39 E8 JSR $E839   Byte aus Puffer holen
E7FD 85 8A    STA $8A     als Anzahl der 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
E807 91 88    STA ($88),Y Byte als Pogrammbyte abspeichern
E809 20 4B E8 JSR $E84B   Wert zu Prüfsumme addieren
E80C A5 88    LDA $88     Programmstartadresse Lo
E80E 18       CLC
E80F 69 01    ADC #$01    erhöhen
E811 85 88    STA $88     und wieder speichern
E813 90 02    BCC $E817   verzweige, wenn noch kein Überlauf
E815 E6 89    INC $89     Programmstartadresse Hi erhöhen
E817 C6 8A    DEC $8A     Anzahl der Bytes vermindern
E819 D0 E7    BNE $E802   weitere Bytes holen, wenn nicht 0
E81B 20 35 CA JSR $CA35   noch ein Byte holen; ohne EOI-Test
E81E A5 85    LDA $85     Datenbyte
E820 C5 87    CMP $87     mit Prüfsumme vergleichen
E822 F0 08    BEQ $E82C   verzweige bei Übereinstimmung
E824 20 3E DE JSR $DE3E   Parameter an DC für Fehlermeldung
E827 A9 50    LDA #$50    Nummer der Fehlermeldung
E829 20 45 E6 JSR $E645   "50, RECORD NOT PRESENT" ausgeben
E82C A5 F8    LDA $F8     auf EOI testen
E82E D0 A8    BNE $E7D8   nächster Block, wenn kein EOI
E830 68       PLA         Programmstartadresse Hi holen
E831 85 89    STA $89     und setzen
E833 68       PLA         Programmstartadresse Lo holen
E834 85 88    STA $88     und setzen
E836 6C 88 00 JMP ($0088) Programm ausführen
E839 20 35 CA JSR $CA35   Byte aus Puffer; ohne EOI-Test
E83C A5 F8    LDA $F8     EOI gesendet ?
E83E D0 08    BNE $E848   verzweige, wenn ja
E840 20 3E DE JSR $DE3E   Parameter an DC für Fehlermeldung
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
E84A 60       RTS         Ende
-------------------------------------------------------------
E84B                      Prüfsumme bilden
```

```
E84B 18       CLC
E84C 65 87    ADC $87     Byte zu Prüfsumme addieren
E84E 69 00    ADC #$00    Überlauf addieren (0 oder 1)
E850 85 87    STA $87     und Prüfsumme neu setzen
E852 60       RTS         Ende
-----------------------------------------------------------------
E853                      IRQ-Routine für den seriellen Bus, die
                          Jedesmal bei Auftreten eines ATN-
                          Signals vom Computer dieses anzeigt.
E853 AD 01 18 LDA $1801   Port A (BC) lesen; IRQ-Flag löschen
E856 A9 01    LDA #$01    Flag für ATN
E858 85 7C    STA $7C     setzen
E85A 60       RTS         Rückkehr zum IRQ-Programm
-----------------------------------------------------------------
```

```
E85B                    Routine zur Bedienung des seriellen Bus
                        nach Auftreten eines ATN.
E85B 78       SEI
E85C A9 00    LDA #$00
E85E 85 7C    STA $7C     Flag für ATN löschen
E860 85 79    STA $79     Flag für LISTEN löschen
E862 85 7A    STA $7A     Flag für TALK löschen
E864 A2 45    LDX #$45
E866 9A       TXS         Stackpointer zurücksetzen
E867 A9 80    LDA #$80
E869 85 F8    STA $F8     EOI—Flag 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 auf Bus setzen
E87B AD 00 18 LDA $1800   Bus wieder lesen
E87E 10 57    BPL $E8D7   verzweige, wenn ATN rückgesetzt 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
E88D 85 79    STA $79     Flag für LISTEN löschen
E88F F0 71    BEQ $E902   unbedingter Sprung
E891 C9 5F    CMP #$5F    UNTALK ?
E893 D0 06    BNE $E89B   verzweige, wenn nein
E895 A9 00    LDA #$00
E897 85 7A    STA $7A     Flag für TALK 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
E8A1 85 7A    STA $7A     Flag für TALK setzen
E8A3 A9 00    LDA #$00
E8A5 85 79    STA $79     Flag für LISTEN 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
E8AF 85 79    STA $79     Flag für LISTEN setzen
E8B1 A9 00    LDA #$00
```

```
E8B3 85 7A     STA $7A     Flag für TALK löschen
E8B5 F0 1B     BEQ $E8D2   unbedingter Sprung
E8B7 AA        TAX         Kommandobyte merken
E8B8 29 60     AND #$60    Bit 5 und 6
E8BA C9 60     CMP #$60    testen
E8BC D0 3F     BNE $E8FD   verzweige, wenn ein Bit gesetzt
E8BE 8A        TXA         Kommandobyte zurückholen
E8BF 85 84     STA $84     als Befehlssekundäradresse setzen
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    Kommandobits isolieren
E8C9 C9 E0     CMP #$E0    CLOSE-Koinmando ?
E8CB D0 35     BNE $E902   verzweige, wenn nein
E8CD 58        CLI
E8CE 20 C0 DA  JSR $DAC0   CLOSE-Routine
E8D1 78        SEI
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    Antwort—Signal 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 von Bus in Puffer schreiben
E8EA 4C E7 EB  JMP $EBE7   zurück zur Warteschleife
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 (freimachen)
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
E8FF 8D 00 18  STA $1800   (ATN-Antwortleitung) löschen
E902 2C 00 18  BIT $1800   Bus prüfen
E905 10 D0     BPL $E8D7   Ende, wenn ATN—Signal Lo
E907 30 F9     BMI $E902   warten, bis ATN zurückgesetzt
```

---

```
E909                       Routine zum Senden von Daten auf den
                           Bus als Folge eines TALK—Kommandos vom
                           Computer.
E909 78        SEI
```

```
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   Wärten bis CLOCK IN Lo wird
E91C 29 01     AND #$01    Datenbit isolieren
E91E 08        PHP         und dessen Wertigkeit merken
E91F 20 B7 E9  JSR $E9B7   CLOCK OUT Leitung Hi setzen
E922 28        PLP         Datenbit zurückholen
E923 F0 12     BEQ $E937   verzweige, wenn Bit = 0
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 = 0
E92F A6 82     LDX $82     Kanalnummer
E931 B5 F2     LDA $F2,X   Kanal Status 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   warten auf Reaktion des Computers
E958 A9 08     LDA #$08    Zähler auf 8 Bits für seriellen Bus
E95A 85 98     STA $98     setzen
E95C 20 C0 E9  JSR $E9C0   Port lesen
E95F 29 01     AND #$01    Datenbit isolieren
E961 D0 36     BNE $E999   verzweige, wenn Bit = 1
E963 A6 82     LDX $82     Kanalnummer
E965 BD 3E 02  LDA $023E,X
```

```
E968 6A       ROR         Bit in Datenregister schieben
E969 9D 3E 02 STA $023E,X
E96C B0 05    BCS $E973   verzweige, wenn Bit = 1
E96E 20 A5 E9 JSR $E9A5   DATA OUT Leitung Lo setzen
E971 D0 03    BNE $E976   unbedingter Sprung
E973 20 9C E9 JSR $E99C   DATA OUT Leitung Hi setzen
E976 20 B7 E9 JSR $E9B7   CLOCK OUT Leitung Hi setzen
E979 A5 23    LDA $23     auf Busverzögerung prüfen
E97B D0 03    BNE $E980   verzweige, wenn keine gewünscht
E97D 20 F3 FE JSR $FEF3   Verzögerung für seriellen Bus
E980 20 FB FE JSR $FEFB   CLOCK OUT Lo, DATA OUT Hi setzen
E983 C6 98    DEC $98     Zähler vermindern
E985 D0 D5    BNE $E95C   verzweige, wenn noch Bits folgen
E987 20 59 EA JSR $EA59   DATA-Leitung setzen
E98A 20 C0 E9 JSR $E9C0   auf Antwort vom Computer testen
E98D 29 01    AND #$01    Datenbit isolieren
E98F F0 F6    BEQ $E987   warten, bis Reaktion vom Computer
E991 58       CLI
E992 20 AA D3 JSR $D3AA   nächstes Byte aus Puffer holen
E995 78       SEI
E996 4C 0F E9 JMP $E90F   und über Bus senden
E999 4C 4E EA JMP $EA4E   zurück zur Warteschleife
-------------------------------------------------------------
E99C                      DATA OUT Leitung auf Hi setzen.
E99C AD 00 18 LDA $1800   Port lesen
E99F 29 FD    AND #$FD    DATA OUT Bit löschen
E9A1 8D 00 18 STA $1800   und 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
E9AA 8D 00 18 STA $1800   und ausgeben
E9AD 60       RTS         Ende
-------------------------------------------------------------
E9AE                      CLOCK Leitung auf Lo setzen.
E9AE AD 00 18 LDA $1800   Port lesen
E9B1 09 08    ORA #$08    CLOCK OUT Bit setzen
E9B3 8D 00 18 STA $1800   und ausgeben
E9B6 60       RTS         Ende
-------------------------------------------------------------
E9B7                      CLOCK Leitung auf Hl setzen.
E9B7 AD 00 18 LDA $1800   Port lesen
E9BA 29 F7    AND #$F7    CLOCK OUT Bit löschen
```

```
E9BC 8D 00 18  STA $1800   und 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   konstanten Wert abwarten
E9C6 D0 F8     BNE $E9C0
E9C8 60        RTS         Ende
-------------------------------------------------------------
E9C9                       Routine holt ein Datenbyte vom Bus als
                           Folge eines LISTEN vom Computer
E9C9 A9 08     LDA #$08
E9CB 85 98     STA $98     Zähler für 8 Bit Übertragung 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 = 0 wird
E9D7 20 9C E9  JSR $E99C   DATA OUT Leitung Hi setzen
E9DA A9 01     LDA #$01    255 Mikrosekunden Verzögerung
E9DC 8D 05 18  STA $1805   durch Timer 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 = 1
E9F0 D0 19     BNE $EA0B   unbedingter Sprung
E9F2 20 A5 E9  JSR $E9A5   DATA OUT Lo setzen; Antwortmeldung
E9F5 A2 0A     LDX #$0A    Verzögerung für Talker
E9F7 CA        DEX         von ca. 50 Mikrosekunden
E9F8 D0 FD     BNE $E9F7   ausführen
E9FA 20 9C E9  JSR $E99C   DATA OUT Hi setzen; Signal löschen
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 = 1
EA07 A9 00     LDA #$00
EA09 85 F8     STA $F8     EOI—Signal anzeigen; Flag setzen
EA0B AD 00 18  LDA $1800   Port lesen
EA0E 49 01     EOR #$01    Datenbit invertieren
EA10 4A        LSR         und ins Carry schieben
EA11 29 02     AND #$02    CLOCK Bit auf gültige Daten testen
EA13 D0 F6     BNE $EA0B   nach einmal, wenn Bit ungültig
```

```
EA15 EA       NOP
EA16 EA       NOP
EA17 EA       NOP
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 = 1
EA24 C6 98    DEC $98     Zähler für 8 Bits vermindern
EA26 D0 E3    BNE $EA0B   verzweige, wenn noch Bits folgen
EA28 20 A5 E9 JSR $E9A5   DATA OUT Lo setzen; Antwortsignal
EA2B A5 85    LDA $85     Datenbyte holen
EA2D 60       RTS         Ende
-------------------------------------------------------------
EA2E                      Holen von Daten vom Bus und Schreiben
                          der Daten in den aktuellen Puffer.
EA2E 78       SEI
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 ?
EA3F F0 03    BEQ $EA44   verzweige, wenn ja
EA41 4C 4E EA JMP $EA4E   zurück zur Warteschleife
EA44 20 C9 E9 JSR $E9C9   Datenbyte vom Bus holen
EA47 58       CLI
EA48 20 B7 CF JSR $CFB7   und in Puffer schreiben
EA4B 4C 2E EA JMP $EA2E   weitermachen; nächstes Byte
EA4E A9 00    LDA #$00
EA50 8D 00 18 STA $1800   Port freimachen
EA53 4C E7 EB JMP $EBE7   zurück zur Warteschleife
EA56 4C 5B E8 JMP $E85B   zur Busbedienung bei ATN
-------------------------------------------------------------
EA59                      Testet auf ATN-Modus.
EA59 A5 7D    LDA $7D     Flag für ATN-Modus
EA5B F0 06    BEQ $EA63   verzweige, wenn kein ATN anliegt
EA5D AD 00 18 LDA $1800   Port lesen
EA60 10 09    BPL $EA6B   verzweige, wenn kein ATN mehr
EA62 60       RTS         Ende
EA63 AD 00 18 LDA $1800   Port lesen
```

```
EA66 10 FA     BPL $EA62   verzweige, wenn kein ATN mehr
EA68 4C 5B E8  JMP $E85B   zur Busbedienung bei ATN
EA6B 4C D7 E8  JMP $E8D7   Bus nach ATN bedienen
-------------------------------------------------------------
EA6E                       Routine zur Behandlung von
                           Hardwaredefekten oder zum Selbsttest.
                           Erfolgt der Einsprung bei $EA6E, so
                           liegt kein Fehler vor sondern es
                           erfolgt ein Selbsttest.
EA6E A2 00     LDX #$00
EA70 2C        .BYTE $2C
EA71                       Beim Einsprung nach $EA71 wurde ein
                           Hardwarefehler festgestellt; die LED
                           blinkt langsam; die Floppy wird
                           verriegelt.
EA71 A6 6F     LDX $6F     Fehlernummer holen
EA73 9A        TXS         Wert merken
EA74 BA        TSX         Registerinhalt zurückholen
EA75 A9 08     LDA #$08    LED-Bit setzen
EA77 0D 00 1C  ORA $1C00   und
EA7A 4C EA FE  JMP $FEEA   LED einschalten; Ruckkehr mit JMP
EA7D 98        TYA         Verzögerung einleiten
EA7E 18        CLC
EA7F 69 01     ADC #$01
EA81 D0 FC     BNE $EA7F
EA83 88        DEY
EA84 D0 F8     BNE $EA7E
EA86 AD 00 1C  LDA $1C00
EA89 29 F7     AND #$F7    LED am Laufwerk ausschalten
EA8B 8D 00 1C  STA $1C00
EA8E 98        TYA         Verzögerung einleiten
EA8F 18        CLC
EA90 69 01     ADC #$01
EA92 D0 FC     BNE $EA90
EA94 88        DEY
EA95 D0 F8     BNE $EA8F
EA97 CA        DEX
EA98 10 DB     BPL $EA75
EA9A E0 FC     CPX #$FC
EA9C D0 F0     BNE $EA8E   Verzögerung abwarten
EA9E F0 D4     BEQ $EA74   LED am Laufwerk wieder einschalten
-------------------------------------------------------------
EAA0                       RESET—Routine der gesamten Floppy-
                           Station.
```

```
EAA0 78       SEI
EAA1 D8       CLD
EAA2 A2 FF    LDX #$FF    Wert für DDRA
EAA4 4C 10 FF JMP $FF10   VIA's setzen; Rückkehr JMP $EAA7
EAA7 E8       INX         X=0
EAA8 A0 00    LDY #$00
EAAA A2 00    LDX #$00
EAAC 8A       TXA
EAAD 95 00    STA $00,X   Zeropage löschen
EAAF E8       INX
EAB0 D0 FA    BNE $EAAC
EAB2 8A       TXA
EAB3 D5 00    CMP $00,X   RAM—Test für Zeropage
EAB5 D0 B7    BNE $EA6E   verzweige, wenn RAM—Fehler
EAB7 F6 00    INC $00,X
EAB9 C8       INY
EABA D0 FB    BNE $EAB7   alle Werte durchprobieren
EABC D5 00    CMP $00,X
EABE D0 AE    BNE $EA6E   verzweige, wenn RAM—Fehler
EAC0 94 00    STY $00,X
EAC2 B5 00    LDA $00,X
EAC4 D0 A8    BNE $EA6E   verzweige, wenn RAM—Fehler
EAC6 E8       INX
EAC7 D0 E9    BNE $EAB2   nächste Zeropage—Adresse
EAC9 E6 6F    INC $6F     Zeiger für ROM-Test
EACB 86 76    STX $76     Hl setzen
EACD A9 00    LDA #$00    Lo Adresse
EACF 85 75    STA $75     setzen
EAD1 A8       TAY
EAD2 A2 20    LDX #$20    32 Pages
EAD4 18       CLC
EAD5 C6 76    DEC $76     Adresse Hi minus 1
EAD7 71 75    ADC ($75),Y Prüfsumme für ROM bilden
EAD9 C8       INY
EADA D0 FB    BNE $EAD7
EADC CA       DEX         nächste Page prüfen
EADD D0 F6    BNE $EAD5
EADF 69 00    ADC #$00    Überlauf addieren
EAE1 AA       TAX         Prüfsumme merken
EAE2 C5 76    CMP $76     mit Prüfwert vergleichen
EAE4 D0 39    BNE $EB1F   verzweige, wenn Fehler
EAE6 E0 C0    CPX #$C0    schon ROM—Anfang erreicht ?
EAE8 D0 DF    BNE $EAC9   verzweige, wenn nein
EAEA A9 01    LDA #$01    Adresse für RAM-Test
```

```
EAEC 85 76     STA $76     Hi setzen
EAEE E6 6F     INC $6F
EAF0 A2 07     LDX #$07    7 Pages sind zu testen
EAF2 98        TYA
EAF3 18        CLC
EAF4 65 76     ADC $76     Prüfwert bilden
EAF6 91 75     STA ($75),Y und in Speicherstelle schreiben
EAF8 C8        INY
EAF9 D0 F7     BNE $EAF2   gesamte Page vollschreiben
EAFB E6 76     INC $76     nächste Page
EAFD CA        DEX         schon Ende erreicht ?
EAFE D0 F2     BNE $EAF2   verzweige, wenn nein
EB00 A2 07     LDX #$07    wieder 7 Pages
EB02 C6 76     DEC $76     Adresse Hi vermindern
EB04 88        DEY
EB05 98        TYA
EB06 18        CLC
EB07 65 76     ADC $76     Prüfwert wieder herstellen
EB09 D1 75     CMP ($75),Y mit Speicherinhalt vergleichen
EB0B D0 12     BNE $EB1F   verzweige, wenn Fehler
EB0D 49 FF     EOR #$FF    Prüfwert invertieren
EB0F 91 75     STA ($75),Y und abspeichern
EB11 51 75     EOR ($75),Y nochmalige Verknüpfung
EB13 91 75     STA ($75),Y muß Null ergeben
EB15 D0 08     BNE $EB1F   verzweige, wenn Fehler
EB17 98        TYA
EB18 D0 EA     BNE $EB04   nächste Speicherstelle
EB1A CA        DEX         nächste Page
EB1B D0 E5     BNE $EB02   weitermachen, wenn noch nicht Ende
EB1D F0 03     BEQ $EB22   unbedingter Sprung
EB1F 4C 71 EA  JMP $EA71   Sprung zu Fehlerblinken

EB22                       Einsprung der kurzen RESET-Routine,
                           ohne Löschen der Pufferspeicher
EB22 A2 45     LDX #$45
EB24 9A        TXS         Stackpointer wieder herstellen
EB25 AD 00 1C  LDA $1C00
EB28 29 F7     AND #$F7    LED am Laufwerk ausschalten
EB2A 8D 00 1C  STA $1C00
EB2D A9 01     LDA #$01
EB2F 8D 0C 18  STA $180C   ATN IN auf negative Flanke triggern
EB32 A9 82     LDA #$82
EB34 8D 0D 18  STA $180D   IRQ in IFR ermöglichen
EB37 8D 0E 18  STA $180E   IER für IRQ setzen
EB3A AD 00 18  LDA $1800   Port lesen
```

```
EB3D 29 60    AND #$60    Bit 5 und 6 für Gerätenummer
EB3F 0A       ASL         isolieren und
EB40 2A       ROL         nach Bitposition
EB41 2A       ROL         0 und 1 schieben
EB42 2A       ROL
EB43 09 48    ORA #$48    Nummer für TALK herstellen
EB45 85 78    STA $78     und abspeichern
EB47 49 60    EOR #$60    Nummer für LISTEN herstellen
EB49 85 77    STA $77     und abspeichern
EB4B A2 00    LDX #$00
EB4D A0 00    LDY #$00
EB4F A9 00    LDA #$00
EB51 95 99    STA $99,X   Lo Bytes der Pufferadressen setzen
EB53 E8       INX
EB54 B9 E0 FE LDA $FEE0,Y Hi Bytes aus Tabelle
EB57 95 99    STA $99,X   und ebenfalls setzen
EB59 E8       INX
EB5A C8       INY
EB5B C0 05    CPY #$05    schon vierter Puffer erledigt ?
EB5D D0 F0    BNE $EB4F   verzweige, wenn nein
EB5F A9 00    LDA #$00
EB61 95 99    STA $99,X
EB63 E8       INX         Zeiger für INPUT-Puffer
EB64 A9 02    LDA #$02    $0200 setzen
EB66 95 99    STA $99,X
EB68 E8       INX
EB69 A9 D5    LDA #$D5
EB6B 95 99    STA $99,X
EB6D E8       INX         Zeiger für ERROR-Puffer
EB6E A9 02    LDA #$02    $02D5 setzen
EB70 95 99    STA $99,X
EB72 A9 FF    LDA #$FF
EB74 A2 12    LDX #$12
EB76 9D 2B 02 STA $022B,X Kanalstatus mit $FF vorbelegen
EB79 CA       DEX         $FF - Kanal nicht belegt
EB7A 10 FA    BPL $EB76
EB7C A2 05    LDX #$05
EB7E 95 A7    STA $A7,X   alle Puffer als frei deklarieren
EB80 95 AE    STA $AE,X   und Tabellen deshalb mit $FF
EB82 95 CD    STA $CD,X   vorbesetzen
EB84 CA       DEX
EB85 10 F7    BPL $EB7E
EB87 A9 05    LDA #$05    Puffer 5
EB89 85 AB    STA $AB     Kanal 4 zuordnen
```

```
EB8B A9 06     LDA #$06    Puffer 6
EB8D 85 AC     STA $AC     Kanal 5 zuordnen
EB8F A9 FF     LDA #$FF
EB91 85 AD     STA $AD     Puffer 7 ist unbenutzt
EB93 85 B4     STA $B4     und inaktiv
EB95 A9 05     LDA #$05    ERROR—Kanal zum Lesen benutzt
EB97 8D 3B 02  STA $023B
EB9A A9 84     LDA #$84    Kommandokanal zum Schreiben benutzt
EB9C 8D 3A 02  STA $023A
EB9F A9 0F     LDA #$0F
EBA1 8D 56 02  STA $0256   Kanäle 0 bis 3 freigeben
EBA4 A9 01     LDA #$01
EBA6 85 F6     STA $F6     READY TO LISTEN setzen
EBA8 A9 88     LDA #$88
EBAA 85 F7     STA $F7     READY TO TALK setzen
EBAC A9 E0     LDA #$E0
EBAE 8D 4F 02  STA $024F   Puffer 0 bis 4 freigeben
EBB1 A9 FF     LDA #$FF
EBB3 8D 50 02  STA $0250   alle restlichen Puffer sind belegt
EBB6 A9 01     LDA #$01
EBB8 85 1C     STA $1C     Flags für WRITE PROTECT setzen
EBBA 85 1D     STA $1D
EBBC 20 63 CB  JSR $CB63   Sprungtabelle für U-BefehIe setzen
EBBF 20 FA CE  JSR $CEFA   Kanaltabelle .initialisieren
EBC2 20 59 F2  JSR $F259   Disk-Controller initialisieren
EBC5 A9 22     LDA #$22
EBC7 85 65     STA $65
EBC9 A9 EB     LDA #$EB    Zeiger für NMI auf $EB22 setzen
EBCB 85 66     STA $66
EBCD A9 0A     LDA #$0A
EBCF 85 69     STA $69     10 als Blockabstand auf Diskette
EBD1 A9 05     LDA #$05
EBD3 85 6A     STA $6A     5 Leseversuche bei Lesefehlern
EBD5 A9 73     LDA #$73    Nummer der Meldung
EBD7 20 C1 E6  JSR $E6C1   "73, CBM DOS V2.6 1541" ausgeben
EBDA A9 1A     LDA #$1A
EBDC 8D 02 18  STA $1802   DDRB für Bus-Controller setzen
EBDF A9 00     LDA #$00
EBE1 8D 00 18  STA $1800   Bus freimachen
EBE4 20 80 E7  JSR $E780   auf AUTOBOOT prüfen (nur ROM -03)
```

---

EBE7 — Eingang der Warteschleife, die solange durchlaufen wird, bis ein Befehl erkannt wird; ansonsten befindet sich die Floppy im STANDBY-Modus.

```
EBE7 58       CLI
EBE8 AD 00 18 LDA $1800   CLOCK, DATA und ATN ACK Leitungen
EBEB 29 E5    AND #$E5    Hi setzen und den Bus damit in den
EBED 8D 00 18 STA $1800   definierten Zustand versetzen
EBF0 AD 55 02 LDA $0255   liegt Kommando an ?
EBF3 F0 0A    BEQ $EBFF   verzweige, wenn nein
EBF5 A9 00    LDA #$00
EBF7 8D 55 02 STA $0255   Kommandoflag löschen
EBFA 85 67    STA $67     NMI-Zustand löschen
EBFC 20 46 C1 JSR $C146   Befehl analysieren und ausführen
EBFF 58       CLI
EC00 A5 7C    LDA $7C     ATN vom Bus-Controller ?
EC02 F0 03    BEQ $EC07   verzweige, wenn nein
EC04 4C 5B E8 JMP $E85B   zur Busbedienung (zurück mit JMP)
EC07 58       CLI
EC08 A9 0E    LDA #$0E
EC0A 85 72    STA $72     maximale SA für Files setzen
EC0C A9 00    LDA #$00
EC0E 85 6F    STA $6F     Zähler für anliegende Jobs; Drive 0
EC10 85 70    STA $70     Zähler für anliegende Jobs; Drive 1
EC12 A6 72    LDX $72     prüft auf aktiven Kanal des DC
EC14 BD 2B 02 LDA $022B,X
EC17 C9 FF    CMP #$FF
EC19 F0 10    BEQ $EC2B   verzweige, wenn Kanal inaktiv
EC1B 29 3F    AND #$3F    Kanalnummer isolieren
EC1D 85 82    STA $82     und abspeichern
EC1F 20 93 DF JSR $DF93   zugehörige Puffernummer holen
EC22 AA       TAX         nach X
EC23 BD 5B 02 LDA $025B,X Drivenummer für Puffer
EC26 29 01    AND #$01    isolieren
EC28 AA       TAX         als Index
EC29 F6 6F    INC $6F,X   entsprechenden Jobzähler erhöhen
EC2B C6 72    DEC $72     nächste Sekundäradresse nehmen
EC2D 10 E3    BPL $EC12   verzweige, wenn noch SA übrig
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    Drivenummer isolieren
EC38 AA       TAX         als Index
EC39 F6 6F    INC $6F,X   entsprechenden Jobzähler erhöhen
EC3B 88       DEY         nächsten Puffer
EC3C 10 F3    BPL $EC31   weitermachen; wenn weiterer Puffer
```

```
EC3E 78       SEI
EC3F AD 00 1C LDA $1C00   Port des DC lesen
EC42 29 F7    AND #$F7   LED-Bit löschen
EC44 48       PHA        neuen Wert merken
EC45 A5 7F    LDA $7F    aktuelle Drivenummer
EC47 85 86    STA $86    merken
EC49 A9 00    LDA #$00
EC4B 85 7F    STA $7F    Drive 0 setzen
EC4D A5 6F    LDA $6F    Jobs vorhanden ?
EC4F F0 0B    BEQ $EC5C  verzweige, wenn nein
EC51 A5 1C    LDA $1C    wurde Diskette gewechselt ?
EC53 F0 03    BEQ $EC58  verzweige, wenn nein
EC55 20 13 D3 JSR $D313  alle Kanäle für Drive 0 schließen
EC58 68       PLA        Maske für DC-Port zurückholen
EC59 09 08    ORA #$08   LED-Bit setzen
EC5B 48       PHA        Maske wieder merken
EC5C E6 7F    INC $7F    Drive 1 setzen
EC5E A5 70    LDA $70    Jobs vorhanden ?
EC60 F0 0B    BEQ $EC6D  verzweige, wenn nein
EC62 A5 1D    LDA $1D    wurde 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        Maske für DC-Port zurückholen
EC6A 09 00    ORA #$00   (LED-Bit für Drive 1 setzen)
EC6C 48       PHA        Maske wieder merken
EC6D A5 86    LDA $86    Drivenummer zurückholen
EC6F 85 7F    STA $7F    und wieder übernehmen
EC71 68       PLA        Maske für DC-Port
EC72 AE 6C 02 LDX $026C  ERRoR-Flag gesetzt ?
EC75 F0 21    BEQ $EC98  verzweige, wenn nein; kein Blinken
EC77 AD 00 1C LDA $1C00
EC7A E0 80    CPX #$80   neuer ERROR ?
EC7C D0 03    BNE $EC81  verzweige, wenn neuer ERROR erkannt
EC7E 4C 8B EC JMP $EC8B  LED-Blinken steuern
EC81 AE 05 18 LDX $1805  Timer auslesen; abgelaufen ?
EC84 30 12    BMI $EC98  verzweige, wenn nein
EC86 A2 A0    LDX #$A0   Timerwert
EC88 8E 05 18 STX $1805  Timer neu setzen
EC8B CE 6C 02 DEC $026C  Fehlerzähler vermindern
EC8E D0 08    BNE $EC98  verzweige, wenn nicht abgelaufen
EC90 4D 6D 02 EOR $026D  LED—Maske umdrehen
EC93 A2 10    LDX #$10
EC95 8E 6C 02 STX $026C  Zähler neu setzen
EC98 8D 00 1C STA $1C00  LED-Status neu setzen
```

```
EC9B 4C FF EB  JMP $EBFF   weiter in Marteschleife
-------------------------------------------------------------
EC9E                       Laden und Aufbereiten des Directory
EC9E A9 00     LDA #$00
ECA0 85 83     STA $83     Sekundaradresse 0 (LOAD)
ECA2 A9 01     LDA #$01    einen
ECA4 20 E2 D1  JSR $D1E2   Kanal suchen; Puffer belegen
ECA7 A9 00     LDA #$00
ECA9 20 C8 D4  JSR $D4C8   Pufferzeiger auf Null setzen
ECAC A6 82     LDX $82     Kanalnummer
ECAE A9 00     LDA #$00
ECB0 9D 44 02  STA $0244,X Endezeiger löschen
ECB3 20 93 DF  JSR $DF93   Puffernummer holen
ECB6 AA        TAX         als Index
ECB7 A5 7F     LDA $7F     Drivenummer
ECB9 9D 5B 02  STA $025B,X in Tabelle schreiben
ECBC A9 01     LDA #$01
ECBE 20 F1 CF  JSR $CFF1   $01 in Puffer schreiben
ECC1 A9 04     LDA #$04
ECC3 20 F1 CF  JSR $CFF1   $04 in Puffer schreiben ($0401)
ECC6 A9 01     LDA #$01
ECC8 20 F1 CF  JSR $CFF1   $0l zweimal
ECCB 20 F1 CF  JSR $CFF1   in Puffer schreiben
ECCE AD 72 02  LDA $0272   Drivenummer für Directory
ECD1 20 F1 CF  JSR $CFF1   in Puffer (1. Zeilennummer)
ECD4 A9 00     LDA #$00    zweiter Teil der 'Zeilennummer'
ECD6 20 F1 CF  JSR $CFF1   ebenfalls in Puffer
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 minus 2
ECE3 D6 99     DEC $99,X
ECE5 A9 00     LDA #$00
ECE7 20 F1 CF  JSR $CFF1   $00 (BASIC Zeilenende) in Puffer
ECEA A9 01     LDA #$01    $010l als Linkadresse der BASIC
ECEC 20 F1 CF  JSR $CFF1   Zeilen; wird nach dem Laden vom
ECEF 20 F1 CF  JSR $CFF1   Interpreter richtiggestellt
ECF2 20 CE C6  JSR $C6CE   Directoryeintrag holen
ECF5 90 2C     BCC $ED23   verzweige, 'wenn kein Eintrag mehr
ECF7 AD 72 02  LDA $0272   Blockzahl Lo
ECFA 20 F1 CF  JSR $CFF1   als Zeilennummer Lo in Puffer
ECFD AD 73 02  LDA $0273   Blockzahl Hi
ED00 20 F1 CF  JSR $CFF1   als Zeilennummer Hi in Puffer
```

```
ED03 20 59 ED  JSR $ED59   Directoryeintrag in Puffer
ED06 A9 00     LDA #$00
ED08 20 F1 CF  JSR $CFF1   $00 als Zeilenendekennzeichen
ED0B D0 DD     BNE $ECEA   verzweige, wenn Puffer 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
ED14 95 99     STA $99,X   Pufferzeiger auf Null setzen
ED16 A9 88     LDA #$88    Flag für READY TO TALK setzen
ED18 A4 82     LDY $82     Kanalnummer
ED1A 8D 54 02  STA $0254   Status für Directorykanal
ED1D 99 F2 00  STA $00F2,Y Status für allgemeinen Kanal
ED20 A5 85     LDA $85     Datenbyte
ED22 60        RTS         Ende
-------------------------------------------------------------
ED23                       Abschluß des Directory herstellen.
ED23 AD 72 02  LDA $0272   Blockzahl Lo
ED26 20 F1 CF  JSR $CFF1   in Puffer schreiben
ED29 AD 73 02  LDA $0273   Blockzahl Hi
ED2C 20 F1 CF  JSR $CFF1   ebenfalls in Puffer schreiben
ED2F 20 59 ED  JSR $ED59   'BLOCKS FREE' 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   Pufferzähler minus 2
ED39 D6 99     DEC $99,X
ED3B A9 00     LDA #$00
ED3D 20 F1 CF  JSR $CFF1   dreimal $00 als
ED40 20 F1 CF  JSR $CFF1   Kennzeichen für BASIC-Programmende
ED43 20 F1 CF  JSR $CFF1   in Puffer 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 als Endezeiger für Puffer
ED53 DE 44 02  DEC $0244,X minus 1
ED56 4C 0D ED  JMP $ED0D   Ende
-------------------------------------------------------------
ED59                       Directoryzeile in den Ausgabepuffer zur
                           Übertragung an den Computer schreiben.
ED59 A0 00     LDY #$00
```

```
ED5B B9 B1 02  LDA $02B1,Y Zeichen aus Directorypuffer
ED5E 20 F1 CF  JSR $CFF1   in Programmpuffer schreiben
ED61 C8        INY         nächstes Zeichen
ED62 C0 1B     CPY #$1B    schon 27 Zeichen
ED64 D0 F5     BNE $ED5B   weitermachen, wenn nein
ED66 60        RTS         Ende
-----------------------------------------------------------------
ED67                       Byte aus Directory holen; ggf. nächsten
                           Block nachladen.
ED67 20 37 D1  JSR $D137   Byte aus Datei; ggf. 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
ED71 B9 44 02  LDA $0244,Y Endezeiger holen
ED74 F0 08     BEQ $ED7E   verzweige bei Kennzeichen f. LOAD $
ED76 A9 80     LDA #$80
ED78 99 F2 00  STA $00F2,Y EOI—Kennzeichen erzeugen
ED7B A5 85     LDA $85     Datenbyte
ED7D 60        RTS         Ende

ED7E 48        PHA         Endezeiger ($00) merken
ED7F 20 EA EC  JSR $ECEA   Directoryzeile in Puffer
ED82 68        PLA         Endezeiger zurückholen
ED83 60        RTS         Ende
-----------------------------------------------------------------
ED84                       VALIDATE-Befehl
ED84 20 D1 C1  JSR $C1D1   Drivenummer aus Befehlsstring holen
ED87 20 42 D0  JSR $D042   Diskette initialisieren
ED8A A9 40     LDA #$40    Flag für BAM 'dirty' setzen
ED8C 8D F9 02  STA $02F9   BAM wurde geändert !!!
ED8F 20 B7 EE  JSR $EEB7   neue BAM in Puffer erzeugen
ED92 A9 00     LDA #$00    Flag für Suche nach gültigem
ED94 8D 92 02  STA $0292   Eintrag im Directory setzen
ED97 20 AC C5  JSR $C5AC   Eintrag in Directory suchen
ED9A D0 3D     BNE $EDD9   verzweige, wenn gefunden
ED9C A9 00     LDA #$00
ED9E 85 81     STA $81     Sektornummer Null setzen
EDA0 AD 85 FE  LDA $FE85   18
EDA3 85 80     STA $80     Tracknummer für BAM setzen
EDA5 20 E5 ED  JSR $EDE5   Directoryblöcke in 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
                            durchgehen und in BAM belegen
EDB3 C8        INY
EDB4 B1 94     LDA ($94),Y Track holen
EDB6 48        PHA         und merken
EDB7 C8        INY
EDB8 B1 94     LDA ($94),Y Sektor 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 SS holen
EDBF F0 0A     BEQ $EDCB   verzweige, wenn kein SS vorhanden
EDC1 85 80     STA $80     Track des SS übernehmen
EDC3 C8        INY
EDC4 B1 94     LDA ($94),Y Sektornummer des SS holen
EDC6 85 81     STA $81     und übernehmen
EDC8 20 E5 ED  JSR $EDE5   SS-Blöcke als belegt kennzeichnen
EDCB 68        PLA         Sektornummer zurückholen
EDCC 85 81     STA $81     und übernehmen
EDCE 68        PLA         Tracknummer zurückholen
EDCF 85 80     STA $80     und übernehmen
EDD1 20 E5 ED  JSR $EDE5   Blöcke des Files in BAM belegen
EDD4 20 04 C6  JSR $C604   nächsten gültigen Fileeintrag holen
EDD7 F0 C3     BEQ $ED9C   verzweige, wenn Directory zu Ende
EDD9 A0 00     LDY #$00
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   File löschen (SCRATCH)
EDE2 4C D4 ED  JMP $EDD4   weitermachen
------------------------------------------------------------------
EDE5                        File anhand der Linker nachverfolgen
                            und Blöcke in BAM belegen.
EDE5 20 5F D5  JSR $D55F   Track und Sektor prüfen
EDE8 20 90 EF  JSR $EF90   Block in BAM belegen
EDEB 20 75 D4  JSR $D475   Kanal öffnen; Block lesen
EDEE A9 00     LDA #$00
EDF0 20 C8 D4  JSR $D4C8   Pufferzeiger auf Null setzen
EDF3 20 37 D1  JSR $D137   Byte aus Puffer holen
EDF6 85 80     STA $80     Tracknummer des nächsten Blocks
EDF8 20 37 D1  JSR $D137   Byte aus Puffer holen
EDFB 85 81     STA $81     Sektornummer des nächsten Blocks
EDFD A5 80     LDA $80     Ende des Files erreicht ?
EDFF D0 03     BNE $EE04   verzweige, wenn nein
EE01 4C 27 D2  JMP $D227   Kanal schließen; Ende
```

```
EE04 20 90 EF  JSR $EF90   Block in BAM belegen
EE07 20 4D D4  JSR $D44D   nächsten Block lesen
EE0A 4C EE ED  JMP $EDEE   weitermachen
-------------------------------------------------------------
EE0D                       NEW-Befehl
EE0D 20 12 C3  JSR $C312   Driveparameter setzen
EE10 A5 E2     LDA $E2     Drivenummer
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 übernehmen
EE1D 20 00 C1  JSR $C100   LED am Laufwerk 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   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   übernehmen
EE31 B9 01 02  LDA $0201,Y zweites Zeichen der neuen Id
EE34 95 13     STA $13,X   übernehmen
EE36 20 07 D3  JSR $D307   alle Kanäle schließen
EE39 A9 01     LDA #$01
EE3B 85 80     STA $80     bei Track 1 beginnen
EE3D 20 C6 C8  JSR $C8C6   Formatierung ausführen
EE40 20 05 F0  JSR $F005   BAM-Puffer löschen
EE43 4C 56 EE  JMP $EE56   weitermachen
EE46 20 42 D0  JSR $D042   Diskette initialisieren
EE49 A6 7F     LDX $7F     Drivenummer
EE4B BD 01 01  LDA $0101,X Formatkennseichen 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 V2.6 1541" ausgeben
EE56 20 B7 EE  JSR $EEB7   neue BAM erzeugen
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 $90; Position des
EE61 95 99     STA $99,X   Disknamens setzen
EE63 AE 7A 02  LDX $027A   Puffernummer holen
EE66 A9 1B     LDA #$1B    27; Länge des Disknamens
```

```
EE68 20 6E C6  JSR $C66E   Disknamen in BAM-Puffer übernehmen
EE6B A0 12     LDY #$12    18
EE6D A6 7F     LDX $7F     Drivenummer
EE6F AD D5 FE  LDA $FED5   ASCII-Code für 'A'; 1541 Format
EE72 9D 01 01  STA $0101,X festlegen
EE75 8A        TXA         Drivenummer
EE76 0A        ASL         mal 2
EE77 AA        TAX         wieder als Index
EE78 B5 12     LDA $12,X   erstes Zeichen der ID
EE7A 91 94     STA ($94),Y in Puffer schreiben
EE7C C8        INY
EE7D B5 13     LDA $13,X   zweites Zeichen der ID
EE7F 91 94     STA ($94),Y ebenfalls in Puffer schreiben
EE81 C8        INY
EE82 C8        INY         21
EE83 A9 32     LDA #$32    ASCII-Code für '2'
EE85 91 94     STA ($94),Y in Puffer schreiben
EE87 C8        INY
EE88 AD D5 FE  LDA $FED5   ASCII-Code für 'A'
EE8B 91 94     STA ($94),Y in Puffer schreiben
EE8D A0 02     LDY #$02    2
EE8F 91 6D     STA ($6D),Y und abermals in Puffer schreiben
EE91 AD 85 FE  LDA $FE85   18
EE94 85 80     STA $80     als Tracknummer übernehmen
EE96 20 93 EF  JSR $EF93   Block 18,0 in BAM belegen
EE99 A9 01     LDA #$01
EE9B 85 81     STA $81     Sektornummer 1 setzen
EE9D 20 93 EF  JSR $EF93   Block 18,1 in 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 #$01
EEA8 A9 FF     LDA #$FF    ersten Directoryblock herstellen
EEAA 91 6D     STA ($6D),Y mit $FF als Anzahl der Bytes
EEAC 20 64 D4  JSR $D464   Block auf Diskette schreiben
EEAF C6 81     DEC $81     Sektornummer jetzt 0
EEB1 20 60 D4  JSR $D460   Block lesen
EEB4 4C 94 C1  JMP $C194   Diskstatus bereitstellen; Ende
-------------------------------------------------------------
EEB7                       Neue BAM erzeugen
EEB7 20 D1 F0  JSR $F0D1   BAM-Puffer löschen
EEBA A0 00     LDY #$00
EEBC A9 12     LDA #$12    18
EEBE 91 6D     STA ($6D),Y Zeiger auf ersten Block des
EEC0 C8        INY         Directory setzen; Block 18,1
```

```
EEC1 98       TYA
EEC2 91 6D    STA ($6D),Y
EEC4 C8       INY
EEC5 C8       INY
EEC6 C8       INY
EEC7 A9 00    LDA #$00
EEC9 85 6F    STA $6F     24 Bits für die Belegung der
EECB 85 70    STA $70     Blöcke pro Track reservieren
EECD 85 71    STA $71
EECF 98       TYA         Zeiger in BAM
EED0 4A       LSR
EED1 4A       LSR         geteilt durch 4 = Tracknummer
EED2 20 4B F2 JSR $F24B   maximale Anzahl der Sektoren holen
EED5 91 6D    STA ($6D),Y und in BAM schreiben
EED7 C8       INY
EED8 AA       TAX
EED9 38       SEC         Bitmuster der belegten Blöcke
EEDA 26 6F    ROL $6F     pro Track in $6F/70/71 erzeugen
EEDC 26 70    ROL $70
EEDE 26 71    ROL $71
EEE0 CA       DEX
EEE1 D0 F6    BNE $EED9
EEE3 B5 6F    LDA $6F,X   Belegung ans Zwischenspeicher
EEE5 91 6D    STA ($6D),Y in BAM schreiben
EEE7 C8       INY
EEE8 E8       INX
EEE9 E0 03    CPX #$03
EEEB 90 F6    BCC $EEE3
EEED C0 90    CPY #$90    schon Ende der BAM erreicht ?
EEEF 90 D6    BCC $EEC7   weitermachen, wenn nein
EEF1 4C 75 D0 JMP $D075   Anzahl der 'BLOCKS FREE' berechnen
-----------------------------------------------------------------
EEF4                      Wenn BAM im Puffer dirty, dann BAM auf
                          Diskette schreiben.
EEF4 20 93 DF JSR $DF93   Puffernummer holen
EEF7 AA       TAX         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 BAM 'dirty' Flag gesetzt ?
EF04 D0 01    BNE $EF07   verzweige, wenn ja
EF06 60       RTS         Ende

EF07 A9 00    LDA #$00
```

```
EF09 99 51 02  STA $0251,Y 'dirty' Flag 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
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
EF22 85 80     STA $80     Track 1 setzen
EF24 0A        ASL
EF25 0A        ASL         mal 4
EF26 85 6D     STA $6D     Anzahl der Bytes/Track in der BAM
EF28 20 20 F2  JSR $F220   BLOCKS FREE auf Richtigkeit prüfen
EF2B E6 80     INC $80     Tracknummer plus 1
EF2D A5 80     LDA $80
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 übernehmen
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
EF4A 85 6D     STA $6D     setzen
EF4C 60        RTS         Ende
-------------------------------------------------------------
EF4D                       Anzahl der freien Blöcke auf Diskette
                           aus $02FA/02FC holen.
EF4D A6 7F     LDX $7F     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'
EF5F 20 CF EF  JSR $EFCF   Zeiger in Bitmuster für Block holem
EF62 38        SEC
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/Tr.
EF71 18        CLC
EF72 B1 6D     LDA ($6D),Y Anzahl der freien Blöcke/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   gleich Track 18
EF7D F0 3B     BEQ $EFBA   übergehen, wenn ja
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 Anzahl Hi erhöhen
EF87 60        RTS         Ende
-------------------------------------------------------------
EF88                       Flag für BAM geändert (Dirty flag)
                           setzen ($025l =1).
EF88 A6 7F     LDX $7F     Drivenummer
EF8A A9 01     LDA #$01
EF8C 9D 51 02  STA $0251,X 'dirty' Flag = 1
EF8F 60        RTS         Ende
-------------------------------------------------------------
EF90                       Block in der BAM als belegt
                           kennzeichnen.
EF90 20 F1 EF  JSR $EFF1   BAM schreiben, wenn 'dirty'
EF93 20 CF EF  JSR $EFCF   Zeiger in Bitmuster des Block holen
EF96 F0 36     BEQ $EFCE   verzweige, wenn Block 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 (belegen)
EF9D 91 6D     STA ($6D),Y neuen Wert wieder abspeichern
EF9F 20 88 EF  JSR $EF88   BAM 'dirty' Flag setzen
EFA2 A4 6F     LDY $6F     Zeiger auf Anzahl der BLOCKS FREE
EFA4 B1 6D     LDA ($6D),Y Anzahl der freien Blocks/Track
EFA6 38        SEC
```

```
EFA7 E9 01     SBC #$01    vermindern
EFA9 91 6D     STA ($6D),Y und wieder abspeichern
EFAB A5 80     LDA $80     Tracknummer
EFAD CD 85 FE  CMP $FE85   Track 18 ?
EFB0 F0 0B     BEQ $EFBD   übergehen, wenn ja
EFB2 BD FA 02  LDA $02FA,X Anzahl BLOCKS FREE Lo
EFB5 D0 03     BNE $EFBA   verzweige, wenn ungleich Null
EFB7 DE FC 02  DEC $02FC,X Anzahl vermindern
EFBA DE FA 02  DEC $02FA,X
EFBD BD FC 02  LDA $02FC,X Anzahl BLOCKS FREE Hl
EFC0 D0 0C     BNE $EFCE   verzweige, wenn mehr als 255 frei
EFC2 BD FA 02  LDA $02FA,X Anzahl BLOCKS FREE Lo
EFC5 C9 03     CMP #$03    weniger als 3 Blöcke frei ?
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 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 BAM suchen
EFD2 98        TYA         Zeiger auf Bitmuster in Y
EFD3 85 6F     STA $6F     merken
EFD5 A5 81     LDA $81     Sektornummer
EFD7 4A        LSR
EFD8 4A        LSR         geteilt durch 8 ergibt das für den
EFD9 4A        LSR         Block zuständige Byte in der BAM
EFDA 38        SEC         (0 bis 2)
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 in Tabelle
EFE3 B1 6D     LDA ($6D),Y Byte aus BAM holen
EFE5 3D E9 EF  AND $EFE9,X zuständiges Bit löschen (belegen)
EFE8 60        RTS         Ende
-----------------------------------------------------------------
EFE9 01 02 04 08 10 20 40 80  Tabelle der Bitmasken für jedes Bit
                           eines Bytes.
```

```
-------------------------------------------------------------
EFF1                        BAM bei Bedarf auf Diskette schreiben.
EFF1 A9 FF     LDA #$FF
EFF3 2C F9 02  BIT $02F9    Flag für BAM schreiben prüfen
EFF6 F0 0C     BEQ $F004    verzweige, wenn nicht gesetzt
EFF8 10 0A     BPL $F004
EFFA 70 08     BVS $F004
EFFC A9 00     LDA #$00
EFFE 8D F9 02  STA $02F9    Flag für BAM schreiben löschen
F001 4C 8A D5  JMP $D58A    BAM auf Diskette schreiben; Ende
F004 60        RTS          Ende
-------------------------------------------------------------
```

```
F005                        Puffer der BAM löschen.
F005 20 3A EF  JSR $EF3A    Zeiger für BAM setzen
F008 A0 00     LDY #$00
F00A 98        TYA
F00B 91 6D     STA ($6D),Y BAM-Puffer löschen
F00D C8        INY
F00E D0 FB     BNE $F00B
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 A6 7F     LDX $7F      aktuelle Drivenummer
F019 B5 FF     LDA $FF,X    Drivestatus holen
F01B F0 05     BEQ $F022    verzweige, wenn Status ok
F01D A9 74     LDA #$74     Nummer für Fehlermeldungen
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      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 $029B,X merken, daß BAM wiederherqestellt
F045 0A        ASL
F046 0A        ASL          mal 4
F047 18        CLC
F048 69 A1     ADC #$A1     plus 161
F04A 85 6D     STA $6D      Zeiger auf Bit Map Lo
F04C A9 02     LDA #$02
F04E 69 00     ADC #$00     Übertrag addieren
F050 85 6E     STA $6E      Zeiger auf Bit Map Hi
```

```
F052 A0 00     LDY #$00
F054 68        PLA          Parameter zurückholen
F055 85 70     STA $70      und speichern
F057 68        PLA          Parameter zurückholen
F058 85 6F     STA $6F      und speichern
F05A 60        RTS          Fertig
-------------------------------------------------------------
F05B                        BAM-Masken im Puffer vertauschen.
F05B A6 6F     LDX $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      Tracknummer
F076 0A        ASL
F077 0A        ASL          mal 4
F078 95 99     STA $99,X    als Pufferzeiger setzen
F07A A5 70     LDA $70      Bitmuster zurückholen
F07C 0A        ASL
F07D 0A        ASL          mal 4
F07E A8        TAY          als Index
F07F A1 99     LDA ($99,X)  Wert aus Puffer
F081 99 A1 02  STA $02A1,Y  merken
F084 A9 00     LDA #$00
F086 81 99     STA ($99,X)  Null in Puffer schreiben
F088 F6 99     INC $99,X    Pufferzeiger erhöhen
F08A C8        INY
F08B 98        TYA
F08C 29 03     AND #$03     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
F09C 4C 8A D5  JMP $D58A    BAM auf Diskette schreiben
```

```
F09F 09 80     ORA #$80    'dirty' Flag
F0A1 8D F9 02  STA $02F9   BAM nicht ok
F0A4 60        RTS         Ende
-------------------------------------------------------------
F0A5                       Maske der BAM in die richtige Position
                           im Speicher bringen.
F0A5 A8        TAY
F0A6 B9 9D 02  LDA $029D,Y BAM im Speicher ?
F0A9 F0 25     BEQ $F0D0   verzweige, wenn nein
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
F0B7 0A        ASL         mal 4
F0B8 95 99     STA $99,X   als Pufferzeiger Lo setzen
F0BA 98        TYA         Index zurückholen
F0BB 0A        ASL
F0BC 0A        ASL         mal 4
F0BD A8        TAY         wieder als Index nehmen
F0BE B9 A1 02  LDA $02A1,Y Wert aus Zwischenspeicher
F0C1 81 99     STA ($99,X) in Puffer schreiben
F0C3 A9 00     LDA #$00
F0C5 99 A1 02  STA $02A1,Y Zwischenspeicher löschen
F0C8 F6 99     INC $99,X   Pufferzeiger erhöhen
F0CA C8        INY         Index erhöhen
F0CB 98        TYA
F0CC 29 03     AND #$03    Bits 0 und 1 isolieren
F0CE D0 EE     BNE $F0BE   weitermachen, wenn gesetzt
F0D0 60        RTS         Ende
-------------------------------------------------------------
F0D1                       Spurnummer der BAM auf Null setzen.
F0D1 A5 7F     LDA $7F     Drivenummer
F0D3 0A        ASL         mal 2
F0D4 AA        TAX         als Index
F0D5 A9 00     LDA #$00
F0D7 9D 9D 02  STA $029D,X Tracknummer löschen
F0DA E8        INX
F0DB 9D 9D 02  STA $029D,X Tracknummer 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          als Index
F0F6 8A        TXA          Puffernummer
F0F7 09 80     ORA #$80     Puffer als belegt kennzeichnen
F0F9 99 A7 00  STA $00A7,Y  Status in Tabelle eintragen
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
F107 4C 86 D5  JMP $D586    Block lesen
F10A 29 0F     AND #$0F     Puffernummer isolieren
F10C 85 F9     STA $F9      und setzen
F10E 60        RTS          Ende
-------------------------------------------------------------
F10F                        Kanalnummer für Bearbeitung der BAM in
                            den Akku holen.
F10F A9 06     LDA #$06
F111 A6 7F     LDX $7F      Drivenummer
F113 D0 03     BNE $F118    Ende, wenn nicht Drive 0
F115 18        CLC
F116 69 07     ADC #$07     ergibt 13 (Kanal für BAM)
F118 60        RTS          Ende
-------------------------------------------------------------
F119                        Kanalnummer für die BAM holen und in X
                            übergeben.
F119 20 0F F1  JSR $F10F    Kanalnummer holen
F11C AA        TAX          nach X
F11D 60        RTS          Ende
-------------------------------------------------------------
```

```
F11E                        Mit der Angabe der aktuellen Spur— 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 setzen
F123 85 6F     STA $6F
F125 A9 01     LDA #$01     Bit 1 als Flag für BAM nicht auf
F127 0D F9 02  ORA $02F9    Diskette schreiben setzen
F12A 8D F9 02  STA $02F9
F12D A5 6F     LDA $6F      Zählwert holen
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
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      aktuelle Tracknummer
F13C CD 85 FE  CMP $FE85    18; Directorytrack ?
F13F F0 19     BEQ $F15A    '72, DISK FULL', wenn ja
F141 90 1C     BCC $F15F    verzweige, wenn kleiner 18
F143 E6 80     INC $80      Tracknummer +1
F145 A5 80     LDA $80
F147 CD D7 FE  CMP $FED7    36; höchste Tracknummer erreicht ?
F14A D0 E1     BNE $F12D    Track absuchen, wenn nein
F14C AE 85 FE  LDX $FE85    18; Directorytrack
F14F CA        DEX          minus 1
F150 86 80     STX $80      setzen
F152 A9 00     LDA #$00
F154 85 81     STA $81      Sektor 0 als Startwert
F156 C6 6F     DEC $6F      Zählerwert minus 1
F158 D0 D3     BNE $F12D    weitersuchen, wenn ungleich 0
F15A A9 72     LDA #$72     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    weitersuchen, wenn ungleich Null
F163 AE 85 FE  LDX $FE85    18; Directorytrack
F166 E8        INX          plus 18
F167 86 80     STX $80      als Tracknummer übernehmen
F169 A9 00     LDA #$00
F16B 85 81     STA $81      Sektor 0 als Startwert
F16D C6 6F     DEC $6F      Zählerwert minus 1
F16F D0 BC     BNE $F12D    weitersuchen, wenn ungleich Null
F171 F0 E7     BEQ $F15A    unbedingt; 'DISK FULL' ausgeben
```

```
------------------------------------------------------------------
F173                        Optimalen nächsten Sektor der aktuellen
                            Spur ausfindig machen.
F173 A5 81     LDA $81      aktueller Sektor
F175 18        CLC
F176 65 69     ADC $69      plus Schrittweite (normal 10)
F178 85 81     STA $81      als neuen Sektor merken
F17A A5 80     LDA $80      aktuelle 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
F185 C5 81     CMP $81      mit aktueller Nummer vergleichen
F187 B0 0C     BCS $F195    verzweige, wenn größer gleich
F189 38        SEC
F18A A5 81     LDA $81      aktueller Sektor
F18C ED 4E 02  SBC $024E    minus maximale Sektornummer
F18F 85 81     STA $81      als neue Sektornummer speichern
F191 F0 02     BEQ $F195    verzweige bei Ergebnis gleich 0
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 frei
F19A 4C 90 EF  JMP $EF90    Block in BAM belegen; Ende
F19D A9 00     LDA #$00
F19F 85 81     STA $81      Sektornummer Null 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
F1AB 0D F9 02  ORA $02F9    Flag für BAM nicht schreiben setzen
F1AE 8D F9 02  STA $02F9
F1B1 A5 86     LDA $86      Zwischenspeicherwert
F1B3 48        PHA          retten
F1B4 A9 01     LDA #$01     Zähler für Tracknummern setzen
F1B6 85 86     STA $86
F1B8 AD 85 FE  LDA $FE85    18; Directorytrack
F1BB 38        SEC
F1BC E5 86     SBC $86      minus Zähler für Tracks
F1BE 85 80     STA $80      als Tracknummer merken
F1C0 90 09     BCC $F1CB    verzweige, wenn Zähler kleiner 18
F1C2 F0 07     BEQ $F1CB    also auf Tracks größer 18 suchen
F1C4 20 11 F0  JSR $F011    richtiges Bitmuster in BAM holen
```

```
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
F1CF 65 86    ADC $86     plus Trackzähler
F1D1 85 80    STA $80     als aktuelle Tracknummer merken
F1D3 E6 86    INC $86     Zähler plus 1
F1D5 CD D7 FE CMP $FED7   36; höchste Tracknummer
F1D8 90 05    BCC $F1DF   weiter, wenn noch nicht erreicht
F1DA A9 67    LDA #$67    Nummer der Fehlermeldung
F1DC 20 45 E6 JSR $E645   '67, ILLEGAL TRACK OR SECTOR'
F1DF 20 11 F0 JSR $F011   richtiges Bitmuster in BAM holen
F1E2 B1 6D    LDA ($6D),Y Zahl der freien Blöcke des Tracks
F1E4 F0 D2    BEQ $F1B8   verzweige, wenn kein Block frei
F1E6 68       PLA         Zwischenspeicherwert zurückholen
F1E7 85 86    STA $86     und wieder abspeichern
F1E9 A9 00    LDA #$00
F1EB 85 81    STA $81     Sektor 0 als Startwert setzen
F1ED 20 FA F1 JSR $F1FA   und freien Sektor suchen
F1F0 F0 03    BEQ $F1F5   verzweige, wenn nicht gefunden
F1F2 4C 90 EF JMP $EF90   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 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     aktuelle Tracknummer
F204 20 4B F2 JSR $F24B   höchste Sektornummer holen
F207 8D 4E 02 STA $024E   und merken
F20A 68       PLA         Index für Bitmap zurückholen
F20B 85 6F    STA $6F     und merken
F20D A5 81    LDA $81     aktuelle Sektornummer
F20F CD 4E 02 CMP $024E   mit Maximalzahl 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
F21F 60       RTS         Ende
-------------------------------------------------------------
```

```
F220                    Gültigkeit der Blockangaben in der BAM
                        überprüfen.
F220 A5 6F     LDA $6F  Speicherwert
F222 48        PHA      retten
F223 A9 00     LDA #$00
F225 85 6F     STA $6F  Zähler gleich Null setzen
F227 AC 86 FE  LDY $FE86 4; Anzahl der Bytes pro Track
F22A 88        DEY      minus 1
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 und alle Sektoren überprüfen
F239 88        DEY      Zeiger auf die nächsten 8 Bits
F23A D0 EF     BNE $F22B und wiederum freie Sektoren holen
F23C B1 6D     LDA ($6D),Y eingetragene Zahl der freien Blöcke
F23E C5 6F     CMP $6F  mit neu errechneter Zahl vergl.
F240 D0 04     BNE $F246 Fehler, wenn ungleich
F242 68        PLA      Speicherwert zurückholen
F243 85 6F     STA $6F  und wieder abspeichern
F245 60        RTS      Ende; alles ok
F246 A9 71     LDA #$71 Nummer der Fehlermeldung
F248 20 45 E6  JSR $E645 '71, DIR ERROR' ausgeben
-------------------------------------------------------------
F24B                    Stellt die Maximalanzahl der Sektoren
                        dieser Spur fest; Spurnummer in A.
F24B AE D6 FE  LDX $FED6 4; Anzahl der BAM-Bytes pro Track
F24E DD D6 FE  CMP $FED6,X Sektornummer mit Maximum vergl.
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 holen
F257 60        RTS      Ende
F258 60        RTS
-------------------------------------------------------------
F259                    Initialisierung der DC-Register.
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 im Portregister löschen
F263 AD 0C 1C  LDA $1C0C PCR; Peripheriekontrollregister
F266 29 FE     AND #$FE CA2 auf negative Flanke triggern;
```

```
F268 09 0E    ORA #$0E    CB1 auf Eingang und CB2 als Kon-
F26A 09 E0    ORA #$E0    trolle für Schreiben/Lesen schalten
F26C 8D 0C 1C STA $1C0C
F26F A9 41    LDA #$41    Timer 1 auf 'free running mode'
F271 8D 0B 1C STA $1C0B   schalten
F274 A9 00    LDA #$00
F276 8D 06 1C STA $1C06   Timerwert für IRQ etwa alle 20 ms
F279 A9 3A    LDA #$3A    setzen
F27B 8D 07 1C STA $1C07
F27E 8D 05 1C STA $1C05   Timer Hi setzen; Timer starten
F281 A9 7F    LDA #$7F
F283 8D 0E 1C STA $1C0E   IRQ-Flag löschen
F286 A9 C0    LDA #$C0
F288 8D 0D 1C STA $1C0D   Interruptmaske setzen; IRQ's
F28B 8D 0E 1C STA $1C0E   zulassen
F28E A9 FF    LDA #$FF
F290 85 3E    STA $3E     alle Laufwerke inaktiv setzen
F292 85 51    STA $51     keine laufende Formatierung setzen
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 $FA05; Routine für
F29E 85 62    STA $62     Steppermotor; setzen
F2A0 A9 FA    LDA #$FA
F2A2 85 63    STA $63
F2A4 A9 C8    LDA #$C8    200; minimale Anzahl der Schritte
F2A6 85 64    STA $64     für 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
F2AE 85 5F    STA $5F
-------------------------------------------------------------
F2B0                      IRQ-Routine des Diskkontrollers. Prüft
                          auf Jobs und führt diese ggf. aus.
F2B0 BA       TSX
F2B1 86 49    STX $49     Stackpointer 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 BYTE-
F2B9 09 0E    ORA #$0E
F2BB 8D 0C 1C STA $1C0C   READY-Leitung zu initialisieren
F2BE A0 05    LDY #$05    Index in Jobspeicher
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    Programm im Puffer ausführen ?
F2C7 D0 04     BNE $F2CD   verzweige, wenn nein
F2C9 98        TYA         Puffernummmer 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 Job für Drive 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 ($00) nach X
F2D9 85 3D     STA $3D     Nummer für DC 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 DC
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 Geschwindigkeit?
F2EB 30 03     BMI $F2F0   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   weiter in Jobschleife
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     Trackdifferenz zu letztem Job
F311 85 4A     STA $4A     setzen
F313 06 4A     ASL $4A     Wert mal 2
F315 A9 60     LDA #$60    Flag für Steppermodus
F317 85 20     STA $20     setzen
F319 B1 32     LDA ($32),Y Tracknummer für Job aus Puffer
F31B 85 22     STA $22     und übernehmen
F31D 4C 9C F9  JMP $F99C   Kopfpositionierung vorbereiten
F320 29 01     AND #$01    Drivenummer
```

```
F322 C5 3D     CMP $3D     gleich zu Nummer des letzten Jobs?
F324 D0 E0     BNE $F306   verzweige, wenn nein
F326 A5 22     LDA $22     Tracknummer von letztem Job
F328 F0 12     BEQ $F33C   verzweige, wenn nicht gesetzt
F32A 38        SEC
F32B F1 32     SBC ($32),Y Abstand zum neuen 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     Drivenummer 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 Disk
F33E B1 32     LDA ($32),Y Tracknummer für Job
F340 85 40     STA $40     setzen
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
F34F 0A        ASL
F350 0A        ASL         mal 32
F351 0A        ASL
F352 0A        ASL
F353 85 44     STA $44     als Wert für Kontrollport setzen
F355 AD 00 1C  LDA $1C00   Kontrollport
F358 29 9F     AND #$9F    Timerkonstante für DC—Hardware
F35A 05 44     ORA $44     errechnen, um Timing beim Schreiben
F35C 8D 00 1C  STA $1C00   und Lesen zu triggern
F35F A6 3D     LDX $3D     Drivenummer
F361 A5 45     LDA $45     Jobcode
F363 C9 40     CMP #$40    BUMP des Tonkopfes ?
F365 F0 15     BEQ $F37C   ausführen, wenn ja
F367 C9 60     CMP #$60    Jobprogramm ausführen?
F369 F0 03     BEQ $F36E   verzweige, wenn ja
F36B 4C B1 F3  JMP $F3B1   Blockheader auf Track suchen
F36E A5 3F     LDA $3F     Puffernummer
F370 18        CLC
F371 69 03     ADC #$03    plus 3
F373 85 31     STA $31     als Pufferadresse Hi setzen
F375 A9 00     LDA #$00
```

```
F377 85 30     STA $30     Pufferadresse Lo setzen
F379 6C 30 00  JMP ($0030) Sprung in Puffer
-------------------------------------------------------------
F37C                       Routine zum Ausführen eines BUMP
F37C A9 60     LDA #$60    Flag für Steppermodus
F37E 85 20     STA $20     setzen
F380 AD 00 1C  LDA $1C00
F383 29 FC     AND #$FC    Steppermotor an
F385 8D 00 1C  STA $1C00
F388 A9 A4     LDA #$A4    164; entspricht —45; Anzahl der
F38A 85 4A     STA $4A     zu überfahrenden Tracks bei BUMP
F38C A9 01     LDA #$01    Tracknummer 1
F38E 85 22     STA $22     setzen
F390 4C 69 F9  JMP $F969   Abschluß der Jobschieife
-------------------------------------------------------------
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 Job
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 Pufferzeiger setzen
F3A5 98        TYA         Index in Jobspeicher
F3A6 18        CLC
F3A7 69 03     ADC #$03    plus 3
F3A9 85 31     STA $31     als Pufferadresse Hi setzen
F3AB A0 00     LDY #$00
F3AD 84 30     STY $30     Pufferadresse Lo setzen
F3AF 68        PLA         Jobcode zurückholen
F3B0 60        RTS         Ende
-------------------------------------------------------------
F3B1                       Suchroutine zum Finden einer Spur auf
                           Diskette. Hierbei wird nach einem
                           gültigen Blockheader gesucht, wobei 90
                           Leseversuche gemacht werden.
F3B1 A2 5A     LDX #$5A    90; Anzahl der Versuche
F3B3 86 4B     STX $4B     als Zähler setzen
F3B5 A2 00     LDX #$00
```

```
F3B7 A9 52     LDA #$52    GCR-Code für $08; Blockheader-
F3B9 85 24     STA $24     kennzeichen für Suche setzen
F3BB 20 56 F5  JSR $F556   SYNC-Signal abwarten
F3BE 50 FE     BVC $F3BE   Byte einlesen
F3C0 B8        CLV
F3C1 AD 01 1C  LDA $1C01
F3C4 C5 24     CMP $24     mit Kennzeichen vergleichen
F3C6 D0 3F     BNE $F407   weitersuchen, wenn ungleich
F3C8 50 FE     BVC $F3C8   Byte einlösen
F3CA B8        CLV
F3CB AD 01 1C  LDA $1C01
F3CE 95 25     STA $25,X   und abspeichern
F3D0 E8        INX
F3D1 E0 07     CPX #$07    7 Bytes des Blockheaders einlesen
F3D3 D0 F3     BNE $F3C8
F3D5 20 97 F4  JSR $F497   Headerbytes nach binär wandeln
F3D8 A0 04     LDY #$04    Index für Prüfsumme
F3DA A9 00     LDA #$00
F3DC 59 16 00  EOR $0016,Y Prüfsumme über Header bilden
F3DF 88        DEY
F3E0 10 FA     BPL $F3DC
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 vom gelesenen Header
F3EA 95 22     STA $22,X   übernehmen
F3EC A5 45     LDA $45     Jobcode
F3EE C9 30     CMP #$30    Code für 'Sektor suchen'?
F3F0 F0 1E     BEQ $F410   verzweige, wenn ja
F3F2 A5 3E     LDA $3E     Drivenummer für Job
F3F4 0A        ASL         mal 2
F3F5 A8        TAY         als Index
F3F6 B9 12 00  LDA $0012,Y ID 1 holen
F3F9 C5 16     CMP $16     und mit gelesener ID 1 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 2 vergleichen
F402 D0 17     BNE $F41B   '29, DISK ID MISMATCH', wenn falsch
F404 4C 23 F4  JMP $F423   nächstbesten Sektor zum bearbeiten
F407 C6 4B     DEC $4B     Zähler für Leseversuche minus 1
F409 D0 B0     BNE $F3BB   ggf. weitersuchen
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 Header
```

```
F412 85 12     STA $12     als neue ID 1 setzen
F414 A5 17     LDA $17     ID 2 vom Header
F416 85 13     STA $13     ebenfalls übernehmen
F418 A9 01     LDA #$01    Nummer der Rückmeldung
F41A 2C        .BYTE $2C
F41B A9 0B     LDA #$0B    Nummer für '29, DISK ID MISMATCH'
F41D 2C        .BYTE $2C
F41E A9 09     LDA #$09    Nummer für '27, WRITE ERROR'
F420 4C 69 F9  JMP $F969   Meldungen ausgeben; Abschluß
-------------------------------------------------------------
F423                       Sucht nach dem nächstbesten Sektor, der
                           bearbeitet werden kann. Optimal ist der
                           übernächste Sektor nach dem gerade
                           bearbeiteten.
F423 A9 7F     LDA #$7F    Sektornummer vorbesetzen
F425 85 4C     STA $4C
F427 A5 19     LDA $19     Sektornummer vom Blockheader
F429 18        CLC
F42A 69 02     ADC #$02    plus 2
F42C C5 43     CMP $43     kleiner als der Maximalwert?
F42E 90 02     BCC $F432   verzweige, wenn ja
F430 E5 43     SBC $43     sonst Maximalwert abziehen
F432 85 4D     STA $4D     nächsten Sektor abspeichern
F434 A2 05     LDX #$05
F436 86 3F     STX $3F     Pufferzeiger
F438 A2 FF     LDX #$FF    setzen
F43A 20 93 F3  JSR $F393   zugehörige Pufferadresse setzen
F43D 10 44     BPL $F483   verzweige, wenn kein Job anliegt
F43F 85 44     STA $44     Jobcode speichern
F441 29 01     AND #$01    Drivenummer isolieren
F443 C5 3E     CMP $3E     Job für dieses Laufwerk ?
F445 D0 3C     BNE $F483   verzweige, wenn nein
F447 A0 00     LDY #$00
F449 B1 32     LDA ($32),Y Tracknummer aus Puffer holen
F44B C5 40     CMP $40     gleich Tracknummer dieses Jobs?
F44D D0 34     BNE $F483   verzweige, wenn nein
F44F A5 45     LDA $45     Jobcode
F451 C9 60     CMP #$60    Jobprogramm im Puffer ausführen?
F453 F0 0C     BEQ $F461   verzweige, wenn ja
F455 A0 01     LDY #$01
F457 38        SEC
F458 B1 32     LDA ($32),Y Sektornummer aus Puffer holen
F45A E5 4D     SBC $4D     größer gleich neue Sektornummer?
F45C 10 03     BPL $F461   verzweige, wenn ja
```

```
F45E 18       CLC
F45F 65 43    ADC $43     Maximale Sektornummer addieren
F461 C5 4C    CMP $4C     Entfernung für diesen Job testen
F463 B0 1E    BCS $F483   ggf. sehen ob anderer Job näher
F465 48       PHA         Vergleichswert merken
F466 A5 45    LDA $45     Jobcode
F468 F0 14    BEQ $F47E   verzweige, wenn kein Job anliegt
F46A 68       PLA         Entfernungswert zurückholen
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
F478 69 03    ADC #$03    plus 3
F47A 85 31    STA $31     als Pufferadresse Hi setzen
F47C D0 05    BNE $F483   unbedingt; 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 klein
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   weiter zum Ende der Jobschleife

F48D 86 3F    STX $3F     Puffernummer 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—Code in den normalen
                          Binärcode.
                          Der GCR-codierte Header steht dabei ab
                          $24; das Ergebnis der Umwandlung steht
                          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 Header
F497 A5 30    LDA $30
F499 48       PHA
```

```
F49A A5 31     LDA $31     Pufferzeiger retten
F49C 48        PHA
F49D A9 24     LDA #$24
F49F 85 30     STA $30     Zeiger auf $0024 (Headerbeginn)
F4A1 A9 00     LDA #$00    setzen
F4A3 85 31     STA $31
F4A5 A9 00     LDA #$00    Zeiger auf erstes GCR-Byte
F4A7 85 34     STA $34     für Decodierung setzen
F4A9 20 E6 F7  JSR $F7E6   3 Bytes decodieren
F4AC A5 55     LDA $55
F4AE 85 18     STA $18     Tracknummer
F4B0 A5 54     LDA $54
F4B2 85 19     STA $19     Sektornummer
F4B4 A5 53     LDA $53
F4B6 85 1A     STA $1A     Prüfsumme
F4B8 20 E6 F7  JSR $F7E6   restliche Bytes decodieren
F4BB A5 52     LDA $52
F4BD 85 17     STA $17     ID 2
F4BF A5 53     LDA $53
F4C1 85 16     STA $16     ID 1
F4C3 68        PLA         Pufferzeiger Hi zurückholen
F4C4 85 31     STA $31     und abspeichern
F4C6 68        PLA         Pufferzeiger Lo zurückholen
F4C7 85 30     STA $30     und speichern
F4C9 60        RTS         Ende
-------------------------------------------------------------
F4CA                       Prüft auf Jobcode zum Lesen; wenn ja,
                           wird der verlangte Header gesucht und
                           der Block gelesen und decodiert.
F4CA C9 00     CMP #$00    Jobcode für Block lesen?
F4CC F0 03     BEQ $F4D1   verzweige, wenn ja
F4CE 4C 6E F5  JMP $F56E   Jobcode weiter prüfen
F4D1 20 0A F5  JSR $F50A   gewünschten Blockheader suchen
F4D4 50 FE     BVC $F4D4   Byte einlösen
F4D6 B8        CLV
F4D7 AD 01 1C  LDA $1C01
F4DA 91 30     STA ($30),Y und in zugehörigen Puffer schreiben
F4DC C8        INY
F4DD D0 F5     BNE $F4D4   256 Bytes lesen
F4DF A0 BA     LDY #$BA
F4E1 50 FE     BVC $F4E1   Byte einlesen
F4E3 B8        CLV
F4E4 AD 01 1C  LDA $1C01
```

```
F4E7 99 00 01  STA $0100,Y Rest des Datenblocks einlesen
F4EA C8        INY         70 Bytes
F4EB D0 F4     BNE $F4E1
F4ED 20 E0 F8  JSR $F8E0   GCR-Bytes nach Binär umwandeln
F4F0 A5 38     LDA $38     erstes Byte des Datenblocks
F4F2 C5 47     CMP $47     gleich $07; Blockkenzeichen ?
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
F505 A9 01     LDA #$01    Nummer für '00, OK'
F507 4C 69 F9  JMP $F969   Rückmeldung ausgeben; Ende
-------------------------------------------------------------
F50A                       Sucht nach einem bestimmten Blockheader
                           und wartet dann das SYNC-Signal des
                           nachfolgenden Datenblocks ab.
F50A 20 10 F5  JSR $F510   Blockheader suchen
F50D 4C 56 F5  JMP $F556   Datenblock einlesen; Ende
-------------------------------------------------------------
F510                       Sucht nach einem Blockheader, dessen
                           Parameter vorher gesetzt wurden. Die
                           Parameter sind entsprechend der
                           Speicherstellen $12 bis 13 zu setzen;
                           im aktuellen Puffer muß der Linker
                           stehen.
F510 A5 3D     LDA $3D     Drivenummer für 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
F51E B1 32     LDA ($32),Y Tracknummer aus Puffer holen
F520 85 18     STA $18     und übernehmen
F522 C8        INY
F523 B1 32     LDA ($32),Y Sektornummer aus Puffer holen
F525 85 19     STA $19     und übernehmen
F527 A9 00     LDA #$00
```

```
F529 45 16     EOR $16     Prüfsumme über den zusammenge-
F52B 45 17     EOR $17     stellten Blockheader
F52D 45 18     EOR $18     berechnen
F52F 45 19     EOR $19
F531 85 1A     STA $1A     und ebenfalls übernehmen
F533 20 34 F9  JSR $F934   Blockheader in GCR umwandeln
F536 A2 5A     LDX #$5A    90 Leseversuche maximal
F538 20 56 F5  JSR $F556   SYNC-Signal abwarten
F53B A0 00     LDY #$00
F53D 50 FE     BVC $F53D   Byte einlesen
F53F B8        CLV
F540 AD 01 1C  LDA $1C01
F543 D9 24 00  CMP $0024,Y mit Blockheader vergleichen
F546 D0 06     BNE $F54E   nächsten Versuch, wenn ungleich
F548 C8        INY
F549 C0 08     CPY #$08    8 Bytes vergleichen
F54B D0 F0     BNE $F53D
F54D 60        RTS         Ende; alles ok
F54E CA        DEX         Zähler für Leseversuche minus 1
F54F D0 E7     BNE $F538   ggf. noch ein Versuch
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    20 ms als maximale Suchzeit
F558 8D 05 18  STA $1805   Timer starten
F55B A9 03     LDA #$03    Nummer der Fehlermeldung
F55D 2C 05 18  BIT $1805   Timer schon abgelaufen'?
F560 10 F1     BPL $F553   '2l, READ ERROR', wenn ja
F562 2C 00 1C  BIT $1C00   SYNC-Signal erkannt ?
F565 30 F6     BMI $F55D   weitermachen, wenn nein
F567 AD 01 1C  LDA $1C01   Port wieder freimachen
F56A B8        CLV         Flag löschen
F56B A0 00     LDY #$00
F56D 60        RTS         Ende
-------------------------------------------------------------
F56E                       prüft auf Jobcode für Schreiben und
                           schreibt ggf. einen Block aus dem
                           aktuellen Puffer auf 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 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 Bit gesetzt
F581 A9 08     LDA #$08    Nummer der Fehlermeldung
F583 4C 69 F9  JMP $F969   '26, WRITE PROTECT ON' ausgeben
F586 20 8F F7  JSR $F78F   Pufferinhalt in GCR umwandeln
F589 20 10 F5  JSR $F510   Blockheader suchen
F58C A2 08     LDX #$09    9 Bytes
F58E 50 FE     BVC $F58E   überlesen, um hinter den Header
F590 B8        CLV         zu kommen
F591 CA        DEX
F592 D0 FA     BNE $F58E
F594 A9 FF     LDA #$FF    Schreib-/Lesekopf auf Schreiben
F596 8D 03 1C  STA $1C03   Port umschalten
F599 AD 0C 1C  LDA $1C0C
F59C 29 1F     AND #$1F    PCR auf Schreibbetrieb umschalten
F59E 09 C0     ORA #$C0
F5A0 8D 0C 1C  STA $1C0C
F5A3 A9 FF     LDA #$FF    $FF; für SYNC-Markierung
F5A5 A2 05     LDX #$05
F5A7 8D 01 1C  STA $1C01   SYNC-Markierung für Datenblock
F5AA B8        CLV         auf Diskette schreiben
F5AB 50 FE     BVC $F5AB
F5AD B8        CLV
F5AE CA        DEX
F5AF D0 FA     BNE $F5AB
F5B1 A0 BB     LDY #$BB
F5B3 B9 00 01  LDA $0100,Y 70 Bytes aus dem Ausweichpuffer
F5B6 50 FE     BVC $F5B6   auf Diskette schreiben
F5B8 B8        CLV
F5B9 8D 01 1C  STA $1C01
F5BC C8        INY
F5BD D0 F4     BNE $F5B3
F5BF B1 30     LDA ($30),Y
F5C1 50 FE     BVC $F5C1   Rest des Datenblocks aus dem
F5C3 B8        CLV         Puffer auf Diskette schreiben
F5C4 8D 01 1C  STA $1C01
F5C7 C8        INY
F5C8 D0 F5     BNE $F5BF
F5CA 50 FE     BVC $F5CA   Ende des Schreibens abwarten
F5CC AD 0C 1C  LDA $1C0C
F5CF 09 E0     ORA #$E0    PCR wieder auf Lesebetrieb
F5D1 8D 0C 1C  STA $1C0C
```

```
F5D4 A9 00     LDA #$00    Kopf wieder auf Lesen; Port
F5D6 8D 03 1C  STA $1C03   auf Eingang umschalten
F5D9 20 F2 F5  JSR $F5F2   Puffer wieder in Binär 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 in Jobspeicher
F5E6 4C B1 F3  JMP $F3B1   Verify ausführen
-----------------------------------------------------------------
F5E9                       Bildet die Prüfsumme über den
                           Datenblock und übergibt diese in A.
F5E9 A9 00     LDA #$00
F5EB A8        TAY
F5EC 51 30     EOR ($30),Y Pufferinhalt verknüpfen
F5EE C8        INY
F5EF D0 FB     BNE $F5EC
F5F1 60        RTS
-----------------------------------------------------------------
F5F2                       Routine wandelt den Inhalt des
                           Ausweichpuffers und den Inhalt des
                           aktuellen Datenpuffers aus dem GCR—Code
                           wieder in den Binärcode um und schreibt
                           diese Werte dann in den aktuellen
                           Puffer zurück.
F5F2 A9 00     LDA #$00
F5F4 85 2E     STA $2E
F5F6 85 30     STA $30
F5F8 85 4F     STA $4F
F5FA A5 31     LDA $31
F5FC 85 4E     STA $4E
F5FE A9 01     LDA #$01    zuerst den Ausweichpuffer
F600 85 31     STA $31     decodieren
F602 85 2F     STA $2F
F604 A9 BB     LDA #$BB
F606 85 34     STA $34
F608 85 36     STA $36
F60A 20 E6 F7  JSR $F7E6
F60D A5 52     LDA $52
F60F 85 38     STA $38
F611 A4 36     LDY $36
F613 A5 53     LDA $53
F615 91 2E     STA ($2E),Y
F617 C8        INY
F618 A5 54     LDA $54
```

```
F61A 91 2E    STA ($2E),Y
F61C C8       INY
F61D A5 55    LDA $55
F61F 91 2E    STA ($2E),Y
F621 C8       INY
F622 84 36    STY $36
F624 20 E6 F7 JSR $F7E6
F627 A4 36    LDY $36
F629 A5 52    LDA $52
F62B 91 2E    STA ($2E),Y
F62D C8       INY
F62E A5 53    LDA $53
F630 91 2E    STA ($2E),Y
F632 C8       INY
F633 F0 0E    BEQ $F643
F635 A5 54    LDA $54
F637 91 2E    STA ($2E),Y
F639 C8       INY
F63A A5 55    LDA $55
F63C 91 2E    STA ($2E),Y
F63E C8       INY
F63F 84 36    STY $36
F641 D0 E1    BNE $F624
F643 A5 54    LDA $54
F645 91 30    STA ($30),Y
F647 C8       INY
F648 A5 55    LDA $55
F64A 91 30    STA ($30),Y
F64C C8       INY
F64D 84 36    STY $36
F64F 20 E6 F7 JSR $F7E6
F652 A4 36    LDY $36
F654 A5 52    LDA $52
F656 91 30    STA ($30),Y
F658 C8       INY
F659 A5 53    LDA $53
F65B 91 30    STA ($30),Y
F65D C8       INY
F65E A5 54    LDA $54
F660 91 30    STA ($30),Y
F662 C8       INY
F663 A5 55    LDA $55
F665 91 30    STA ($30),Y
F667 C8       INY
```

```
F668 84 36     STY $36
F66A C0 BB     CPY #$BB
F66C 90 E1     BCC $F64F
F66E A9 45     LDA #$45
F670 85 2E     STA $2E
F672 A5 31     LDA $31
F674 85 2F     STA $2F
F676 A0 BA     LDY #$BA
F678 B1 30     LDA ($30),Y
F67A 91 2E     STA ($2E),Y
F67C 88        DEY
F67D D0 F9     BNE $F678
F67F B1 30     LDA ($30),Y
F681 91 2E     STA ($2E),Y
F683 A2 BB     LDX #$BB
F685 BD 00 01  LDA $0100,X
F688 91 30     STA ($30),Y
F68A C8        INY
F68B E8        INX
F68C D0 F7     BNE $F685
F68E 86 50     STX $50
F690 60        RTS
```

---

```
F691                      Prüft auf den Jobcode für Verify und
                          vergleicht ggf. die Daten im aktuellen
                          Puffer nach GCR—Codierung direkt mit
                          den Daten auf 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 untersuchen
F698 20 E9 F5  JSR $F5E9  Prüfsumme über Datenblock berechnen
F69B 85 3A     STA $3A    und abspeichern
F69D 20 8F F7  JSR $F78F  Datenblock in GCR umwandeln
F6A0 20 0A F5  JSR $F50A  Datenblockbeginn suchen
F6A3 A0 BB     LDY #$BB
F6A5 B9 00 01  LDA $0100,Y 70 Bytes aus Ausweichpuffer
F6A8 50 FE     BVC $F6A8
F6AA B8        CLV
F6AB 4D 01 1C  EOR $1C01  mit Bytes auf Diskette vergleichen
F6AE D0 15     BNE $F6C5  verzweige, wenn ungleich
F6B0 C8        INY        nächstes Byte
F6B1 D0 F2     BNE $F6A5
F6B3 B1 30     LDA ($30),Y Bytes aus Datenpuffer
F6B5 50 FE     BVC $F6B5
```

```
F6B7 B8       CLV
F6B8 4D 01 1C EOR $1C01   mit Bytes auf Diskette vergleichen
F6BB D0 08    BNE $F6C5   verzweige, wenn ungleich
F6BD C8       INY         nächstes Byte
F6BE C0 FD    CPY #$FD
F6C0 D0 F1    BNE $F6B3
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
-------------------------------------------------------------
F6CA                      Sucht nach einem Blockheader.
F6CA 20 10 F5 JSR $F510   Blockheader auf Dislette suchen
F6CD 4C 18 F4 JMP $F418   Ende; alles ok
-------------------------------------------------------------
F6D0                      Wandelt 4 Werte in den Speicher-
                          steilen $52-55 in 5 GCR-codierte Bytes
                          um und schreibt diese in den Puffer.
                          Pufferzeiger dabei in $34.
F6D0 A9 00    LDA #$00
F6D2 85 57    STA $57
F6D4 85 5A    STA $5A
F6D6 A4 34    LDY $34
F6D8 A5 52    LDA $52
F6DA 29 F0    AND #$F0
F6DC 4A       LSR
F6DD 4A       LSR
F6DE 4A       LSR
F6DF 4A       LSR
F6E0 AA       TAX
F6E1 BD 7F F7 LDA $F77F,X
F6E4 0A       ASL
F6E5 0A       ASL
F6E6 0A       ASL
F6E7 85 56    STA $56
F6E9 A5 52    LDA $52
F6EB 29 0F    AND #$0F
F6ED AA       TAX
F6EE BD 7F F7 LDA $F77F,X
F6F1 6A       ROR
F6F2 66 57    ROR $57
F6F4 6A       ROR
F6F5 66 57    ROR $57
F6F7 29 07    AND #$07
```

```
F6F9 05 56     ORA $56
F6FB 91 30     STA ($30),Y
F6FD C8        INY
F6FE A5 53     LDA $53
F700 29 F0     AND #$F0
F702 4A        LSR
F703 4A        LSR
F704 4A        LSR
F705 4A        LSR
F706 AA        TAX
F707 BD 7F F7  LDA $F77F,X
F70A 0A        ASL
F70B 05 57     ORA $57
F70D 85 57     STA $57
F70F A5 53     LDA $53
F711 29 0F     AND #$0F
F713 AA        TAX
F714 BD 7F F7  LDA $F77F,X
F717 2A        ROL
F718 2A        ROL
F719 2A        ROL
F71A 2A        ROL
F71B 85 58     STA $58
F71D 2A        ROL
F71E 29 01     AND #$01
F720 05 57     ORA $57
F722 91 30     STA ($30),Y
F724 C8        INY
F725 A5 54     LDA $54
F727 29 F0     AND #$F0
F729 4A        LSR
F72A 4A        LSR
F72B 4A        LSR
F72C 4A        LSR
F72D AA        TAX
F72E BD 7F F7  LDA $F77F,X
F731 18        CLC
F732 6A        ROR
F733 05 58     ORA $58
F735 91 30     STA ($30),Y
F737 C8        INY
F738 6A        ROR
F739 29 80     AND #$80
F73B 85 59     STA $59
```

```
F73D A5 54     LDA $54
F73F 29 0F     AND #$0F
F741 AA        TAX
F742 BD 7F F7  LDA $F77F,X
F745 0A        ASL
F746 0A        ASL
F747 29 7C     AND #$7C
F749 05 59     ORA $59
F74B 85 59     STA $59
F74D A5 55     LDA $55
F74F 29 F0     AND #$F0
F751 4A        LSR
F752 4A        LSR
F753 4A        LSR
F754 4A        LSR
F755 AA        TAX
F756 BD 7F F7  LDA $F77F,X
F759 6A        ROR
F75A 66 5A     ROR $5A
F75C 6A        ROR
F75D 66 5A     ROR $5A
F75F 6A        ROR
F760 66 5A     ROR $5A
F762 29 03     AND #$03
F764 05 59     ORA $59
F766 91 30     STA ($30),Y
F768 C8        INY
F769 D0 04     BNE $F76F
F76B A5 2F     LDA $2F
F76D 85 31     STA $31
F76F A5 55     LDA $55
F771 29 0F     AND #$0F
F773 AA        TAX
F774 BD 7F F7  LDA $F77F,X
F777 05 5A     ORA $5A
F779 91 30     STA ($30),Y
F77B C8        INY
F77C 84 34     STY $34
F77E 60        RTS
```

---

```
F77F                    Bytes für die Umwandlung von
                        Binärwerten in GCR-codierte Werte.
F77F 0A 0B 12 13 0E 0F 16 17
F787 09 19 1A 1B 0D 1D 1E 15
```

```
-------------------------------------------------------------------
F78F                        Wandelt den gesamten aktiven Puffer von
                            Binärwerten in GCR-Codes um. Der
                            Überschuß an Bytes wird dabei im
                            Ausweichpuffer von $01BB-01FF abgelegt.
F78F A9 00     LDA #$00
F791 85 30     STA $30
F793 85 2E     STA $2E
F795 85 36     STA $36
F797 A9 BB     LDA #$BB
F799 85 34     STA $34
F79B 85 50     STA $50
F79D A5 31     LDA $31
F79F 85 2F     STA $2F
F7A1 A9 01     LDA #$01
F7A3 85 31     STA $31
F7A5 A5 47     LDA $47
F7A7 85 52     STA $52
F7A9 A4 36     LDY $36
F7AB B1 2E     LDA ($2E),Y
F7AD 85 53     STA $53
F7AF C8        INY
F7B0 B1 2E     LDA ($2E),Y
F7B2 85 54     STA $54
F7B4 C8        INY
F7B5 B1 2E     LDA ($2E),Y
F7B7 85 55     STA $55
F7B9 C8        INY
F7BA 84 36     STY $36
F7BC 20 D0 F6  JSR $F6D0
F7BF A4 36     LDY $36
F7C1 B1 2E     LDA ($2E),Y
F7C3 85 52     STA $52
F7C5 C8        INY
F7C6 F0 11     BEQ $F7D9
F7C8 B1 2E     LDA ($2E),Y
F7CA 85 53     STA $53
F7CC C8        INY
F7CD B1 2E     LDA ($2E),Y
F7CF 85 54     STA $54
F7D1 C8        INY
F7D2 B1 2E     LDA ($2E),Y
F7D4 85 55     STA $55
```

```
F7D6 C8       INY
F7D7 D0 E1    BNE $F7BA
F7D9 A5 3A    LDA $3A
F7DB 85 53    STA $53
F7DD A9 00    LDA #$00
F7DF 85 54    STA $54
F7E1 85 55    STA $55
F7E3 4C D0 F6 JMP $F6D0
```

---

F7E6 — Wandelt 5 GCR-codierte Werte aus Puffer in 4 Binärwerte um und speichert diese dann nach $56—59[4]. Pufferzeiger in $34.

```
F7E6 A4 34    LDY $34
F7E8 B1 30    LDA ($30),Y
F7EA 29 F8    AND #$F8
F7EC 4A       LSR
F7ED 4A       LSR
F7EE 4A       LSR
F7EF 85 56    STA $56
F7F1 B1 30    LDA ($30),Y
F7F3 29 07    AND #$07
F7F5 0A       ASL
F7F6 0A       ASL
F7F7 85 57    STA $57
F7F9 C8       INY
F7FA D0 06    BNE $F802
F7FC A5 4E    LDA $4E
F7FE 85 31    STA $31
F800 A4 4F    LDY $4F
F802 B1 30    LDA ($30),Y
F804 29 C0    AND #$C0
F806 2A       ROL
F807 2A       ROL
F808 2A       ROL
F809 05 57    ORA $57
F80B 85 57    STA $57
F80D B1 30    LDA ($30),Y
F80F 29 3E    AND #$3E
F811 4A       LSR
F812 85 58    STA $58
F814 B1 30    LDA ($30),Y
F816 29 01    AND #$01
F818 0A       ASL
```

---

[4] @ST: Die Daten werden **nicht** nach $56-$59, sondern vielmehr nach $52-$55 gespeichert.

```
F819 0A       ASL
F81A 0A       ASL
F81B 0A       ASL
F81C 85 59    STA $59
F81E C8       INY
F81F B1 30    LDA ($30),Y
F821 29 F0    AND #$F0
F823 4A       LSR
F824 4A       LSR
F825 4A       LSR
F826 4A       LSR
F827 05 59    ORA $59
F829 85 59    STA $59
F82B B1 30    LDA ($30),Y
F82D 29 0F    AND #$0F
F82F 0A       ASL
F830 85 5A    STA $5A
F832 C8       INY
F833 B1 30    LDA ($30),Y
F835 29 80    AND #$80
F837 18       CLC
F838 2A       ROL
F839 2A       ROL
F83A 29 01    AND #$01
F83C 05 5A    ORA $5A
F83E 85 5A    STA $5A
F840 B1 30    LDA ($30),Y
F842 29 7C    AND #$7C
F844 4A       LSR
F845 4A       LSR
F846 85 5B    STA $5B
F848 B1 30    LDA ($30),Y
F84A 29 03    AND #$03
F84C 0A       ASL
F84D 0A       ASL
F84E 0A       ASL
F84F 85 5C    STA $5C
F851 C8       INY
F852 D0 06    BNE $F85A
F854 A5 4E    LDA $4E
F856 85 31    STA $31
F858 A4 4F    LDY $4F
F85A B1 30    LDA ($30),Y
F85C 29 E0    AND #$E0
```

```
F85E 2A       ROL
F85F 2A       ROL
F860 2A       ROL
F861 2A       ROL
F862 05 5C    ORA $5C
F864 85 5C    STA $5C
F866 B1 30    LDA ($30),Y
F868 29 1F    AND #$1F
F86A 85 5D    STA $5D
F86C C8       INY
F86D 84 34    STY $34
F86F A6 56    LDX $56
F871 BD A0 F8 LDA $F8A0,X
F874 A6 57    LDX $57
F876 1D C0 F8 ORA $F8C0,X
F879 85 52    STA $52
F87B A6 58    LDX $58
F87D BD A0 F8 LDA $F8A0,X
F880 A6 59    LDX $59
F882 1D C0 F8 ORA $F8C0,X
F885 85 53    STA $53
F887 A6 5A    LDX $5A
F889 BD A0 F8 LDA $F8A0,X
F88C A6 5B    LDX $5B
F88E 1D C0 F8 ORA $F8C0,X
F891 85 54    STA $54
F893 A6 5C    LDX $5C
F895 BD A0 F8 LDA $F8A0,X
F898 A6 5D    LDX $5D
F89A 1D C0 F8 ORA $F8C0,X
F89D 85 55    STA $55
F89F 60       RTS
```

---

```
F8A0                    Bytes für die Umwandlung von GCR—
                        codierten Werten in Binärwerte.
F8A0 FF FF FF FF FF FF FF FF
F8A8 FF 80 00 10 FF C0 40 50
F8B0 FF FF 20 30 FF F0 60 70
F8B8 FF 90 A0 B0 FF D0 E0 FF
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-Werte aus dem
                            Ausweichpuffer $01BB-01FF in binäre
                            Werte um und legt diese im aktuellen
                            Puffer ab.
F8E0 A9 00     LDA #$00
F8E2 85 34     STA $34      Pufferzeiger löschen
F8E4 85 2E     STA $2E
F8E6 85 36     STA $36      Pufferadresse nach $2E/2F
F8E8 A9 01     LDA #$01
F8EA 85 4E     STA $4E
F8EC A9 BA     LDA #$BA
F8EE 85 4F     STA $4F
F8F0 A5 31     LDA $31
F8F2 85 2F     STA $2F
F8F4 20 E6 F7  JSR $F7E6
F8F7 A5 52     LDA $52
F8F9 85 38     STA $38
F8FB A4 36     LDY $36
F8FD A5 53     LDA $53
F8FF 91 2E     STA ($2E),Y
F901 C8        INY
F902 A5 54     LDA $54
F904 91 2E     STA ($2E),Y
F906 C8        INY
F907 A5 55     LDA $55
F909 91 2E     STA ($2E),Y
F90B C8        INY
F90C 84 36     STY $36
F90E 20 E6 F7  JSR $F7E6
F911 A4 36     LDY $36
F913 A5 52     LDA $52
F915 91 2E     STA ($2E),Y
F917 C8        INY
F918 F0 11     BEQ $F92B
F91A A5 53     LDA $53
F91C 91 2E     STA ($2E),Y
F91E C8        INY
F91F A5 54     LDA $54
F921 91 2E     STA ($2E),Y
F923 C8        INY
F924 A5 55     LDA $55
F926 91 2E     STA ($2E),Y
F928 C8        INY
F929 D0 E1     BNE $F90C
```

```
F92B A5 53     LDA $53
F92D 85 3A     STA $3A
F92F A5 2F     LDA $2F
F931 85 31     STA $31
F933 60        RTS
-----------------------------------------------------------------
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
F936 85 2F     STA $2F     Pufferadresse nach $2E/2F
F938 A9 00     LDA #$00
F93A 85 31     STA $31
F93C A9 24     LDA #$24
F93E 85 34     STA $34     Pufferzeiger auf #$24
F940 A5 39     LDA $39     Konstante $08; Kennzeichen für
F942 85 52     STA $52     Blockheader
F944 A5 1A     LDA $1A     Prüfsumme des Blockheaders
F946 85 53     STA $53
F948 A5 19     LDA $19     Sektornummer des Blocks
F94A 85 54     STA $54
F94C A5 18     LDA $18     Tracknummer des Blocks
F94E 85 55     STA $55
F950 20 D0 F6  JSR $F6D0   4 Bytes in 5 GCR-Bytes umwandeln
F953 A5 17     LDA $17     ID 2 des Blockheaders
F955 85 52     STA $52
F957 A5 16     LDA $16     ID 1 des Blockheaders
F959 85 53     STA $53
F95B A9 00     LDA #$00    keine übrigen Werte mehr; Zwischen-
F95D 85 54     STA $54     speicher löschen
F95F 85 55     STA $55
F961 20 D0 F6  JSR $F6D0   4 Bytes in 5 GCR-Bytes umwandeln
F964 A5 2F     LDA $2F     Pufferzeiger zurückholen
F966 85 31     STA $31
F968 60        RTS
-----------------------------------------------------------------
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   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 1.5 Sekunden
                           gewartet, bis das Laufwerk für den
                           Betrieb freigegeben wird, da der Motor
                           eine Hochlaufzeit benötigt.
                           Die Betriebsbereitschaft des Laufwerkes
                           wird durch Bit 7 in $20 signalisiert
                           (Bit 7 dann 0).
F97E A9 A0     LDA #$A0    Flag für Motor im Anlaufen aber
F980 85 20     STA $20     noch nicht auf Endgeschwindigkeit
F982 AD 00 1C  LDA $1C00
F985 09 04     ORA #$04    Motor einschalten
F987 8D 00 1C  STA $1C00
F98A A9 3C     LDA #$3C    Zähler für Anlaufphase des Motors
F98C 85 48     STA $48     auf 1.5 Sekunden (60 IRQ's) setzen
F98E 60        RTS         Ende
-------------------------------------------------------------
F98F                       Laufwerksmotor ausschalten, nachdem ein
                           Zähler für die Ausschaltverzögerung in
                           $48 eine 6.4 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
F995 85 20     STA $20
F997 A9 FF     LDA #$FF    255 IRQ's entspricht einer Verzö-
F999 85 48     STA $48     gerung von 6.4 Sekunden
F99B 60        RTS         Ende
-------------------------------------------------------------
F99C                       Diese Routine ist die DC Kontroll—
                           routine des DOS. Sie steuert Laufwerks—
                           und Steppermotor und bildet den
                           jeweiligen IRQ—Abschluß eines
                           Durchlaufs. Der Aufruf dieser Routine
                           erfolgt alle 10 ms.
```

```
F99C AD 07 1C  LDA $1C07   Timer neu setzen; IRQ-Status
F99F 8D 05 1C  STA $1C05   löschen
F9A2 AD 00 1C  LDA $1C00   Testet durch Abfrage der Schreib—
F9A5 29 10     AND #$10    schutzlichtschranke auf Disketten-
F9A7 C5 1E     CMP $1E     wechsel
F9A9 85 1E     STA $1E     neuen Status merken
F9AB F0 04     BEQ $F9B1   verzweige, wenn kein Wechsel
F9AD A9 01     LDA #$01    Flag für Diskettenwechsel
F9AF 85 1C     STA $1C     setzen
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' setzen
F9BC 8D FE 02  STA $02FE
F9BF F0 0A     BEQ $F9CB   unbedingter Sprung

F9C1 85 4A     STA $4A     Kopf steht auf Halbspur; also muß
F9C3 A9 02     LDA #$02    Kopf einen 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
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     Verzögerung für Drivemotor
F9DB D0 1D     BNE $F9FA   verzweige, wenn noch nicht zu Ende
F9DD 98        TYA         Flags für Drivestatus
F9DE 10 04     BPL $F9E4   verzweige, wenn Drive bereit
F9E0 29 7F     AND #$7F    Flag für Drive nicht bereit löschen
F9E2 85 20     STA $20     und übernehmen
F9E4 29 10     AND #$10    Motor in Ausschaltphase?
F9E6 F0 12     BEQ $F9FA   verzweige, wenn nein
F9E8 AD 00 1C  LDA $1C00
F9EB 29 FB     AND #$FB    Drivemotor ausschalten
F9ED 8D 00 1C  STA $1C00
F9F0 A9 FF     LDA #$FF    Flag für aktives Drive
F9F2 85 3E     STA $3E     löschen
F9F4 A9 00     LDA #$00
F9F6 85 20     STA $20     Flags für Drivestatus löschen
```

```
F9F8 F0 DC     BEQ $F9D6   unbedingter Sprung; IRQ Ende

F9FA 98        TYA         Flags für Drivestatus
F9FB 29 40     AND #$40    soll Steppermotor aktiviert werden7
F9FD D0 03     BNE $FA02   verzweige, wenn ja
F9FF 4C BE FA  JMP $FABE   zurück aus IRQ—Programm
FA02 6C 62 00  JMP ($0062) zur aktuellen Steppermotorroutine:
                           $FA3B — kurzer Steppermodus
                           $FA4E — Beenden des Steppermodus
                           $FA7B - Anfahren des Steppermotors
                           $FA97 — schneller Steppermodus
                           $FAA5 - Abbremsen des Motors
-------------------------------------------------------------
FA05                       Steppermotorsteuerung
FA05 A5 4A     LDA $4A     Anzahl der Stepperschritte
FA07 10 05     BPL $FA0E   verzweige, wenn Bewegung nach außen
FA09 49 FF     EOR #$FF    Komplement bilden für Absolutwert
FA0B 18        CLC
FA0C 69 01     ADC #$01
FA0E C5 64     CMP $64     Distanz größer $C8 (200)?
FA10 B0 0A     BCS $FA1C   schneller Steppermodus, wenn ja
FA12 A9 3B     LDA #$3B
FA14 85 62     STA $62     Adresse auf $FA3B für langsamen
FA16 A9 FA     LDA #$FA    Steppermodus
FA18 85 63     STA $63
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 Brems-
FA20 85 61     STA $61     schritte (jeweils 4) abziehen
FA22 A5 5E     LDA $5E     Zahl der Anfahrschritte
FA24 85 60     STA $60     setzen
FA26 A9 7B     LDA #$7B
FA28 85 62     STA $62     Adresse auf $FA7B für schnellen
FA2A A9 FA     LDA #$FA    Steppermodus
FA2C 85 63     STA $63
FA2E A5 4A     LDA $4A     Anzahl der zu fahrenden Schritte
FA30 10 31     BPL $FA63   verzweige, wenn Bewegung nach außen
FA32 E6 4A     INC $4A     Schritte minus 1 (Komplement!)
FA34 AE 00 1C  LDX $1C00   Kontrollport
FA37 CA        DEX         minus 1
FA38 4C 69 FA  JMP $FA69   weiter
-------------------------------------------------------------
FA3B                       langsamer Steppermodus für kurze Weg
                           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
FA41 85 62     STA $62     Adresse auf $FA4E für Beenden der
FA43 A9 FA     LDA #$FA    Kopfpositionierung
FA45 85 63     STA $63
FA47 A9 05     LDA #$05    5 Schritte 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 den
                           Steppermotor.
FA4E C6 60     DEC $60     Zähler zum Abbremsen 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; Steppermotor nicht
FA56 85 20     STA $20     mehr benötigt
FA58 A9 05     LDA #$05
FA5A 85 62     STA $62     Adresse auf $FA05 zur Rückkehr
FA5C A9 FA     LDA #$FA    aus den Stepperroutinen
FA5E 85 63     STA $63
FA60 4C BE FA  JMP $FABE   zurück aus IRQ-Programm
-------------------------------------------------------------
FA63                       Serviceroutine für Steppermotor;
                           beendet unter anderem alle Steppermodi
                           und schaltet den Motor ab.
FA63 C6 4A     DEC $4A     Schrittzähler vermindern
FA65 AE 00 1C  LDX $1C00   Steuerport
FA68 E8        INX         Kopfbewegung durch Konstruktion
FA69 8A        TXA         von bestimmten Bitmustern steuern
                           ---------------------------------------
                           Die Kombinationen in der Folge
                           00/01/10/11/00 bewegen den Kopf nach
                           innen; die Folge
                           00/11/10/01/00 sorgt für eine Bewegung
                           des Kopfes nach außen.
                           ---------------------------------------
FA6A 29 03     AND #$03    Bits 0 und 1 isolieren
FA6C 85 4B     STA $4B     und deren Status merken
FA6E AD 00 1C  LDA $1C00
FA71 29 FC     AND #$FC    bisherige Bits 0 und 1 löschen
FA73 05 4B     ORA $4B     und neuen Status setzen
FA75 8D 00 1C  STA $1C00
FA78 4C BE FA  JMP $FABE   zurück aus IRQ—Programm
```

```
------------------------------------------------------------------
FA7B                        Steppermotor anfahren;
FA7B 38       SEC
FA7C AD 07 1C LDA $1C07     Timer durch Abziehen des Faktors
FA7F E5 5F    SBC $5F       für das Anfahren des Motors
FA81 8D 05 1C STA $1C05     setzen
FA84 C6 60    DEC $60       Anzahl der Anfahrschritte minus 1
FA86 D0 0C    BNE $FA94     verzweige, wenn noch nicht Null
FA88 A5 5E    LDA $5E       Zähler schon für das Abbremsen
FA8A 85 60    STA $60       setzen
FA8C A9 97    LDA #$97
FA8E 85 62    STA $62       Adresse auf $FA97 für den schnellen
FA90 A9 FA    LDA #$FA      Steppermodus
FA92 85 63    STA $63
FA94 4C 2E FA JMP $FA2E     direkt zur Steppermotorsteuerung
------------------------------------------------------------------
FA97                        Routine zur Steuerung des schnellen
                            Steppermodus (Laufmodus).
FA97 C6 61    DEC $61       Schrittzähler minus 1
FA99 D0 F9    BNE $FA94     weitermachen, wenn noch nicht Null
FA9B A9 A5    LDA #$A5
FA9D 85 62    STA $62       Adresse auf $FAA5 zum Abbremsen
FA9F A9 FA    LDA #$FA      des Steppermotors
FAA1 85 63    STA $63
FAA3 D0 EF    BNE $FA94     unbedingter Sprung
------------------------------------------------------------------
FAA5                        Routine zum Abbremsen des
                            Steppermotors.
FAA5 AD 07 1C LDA $1C07
FAA8 18       CLC           Timer durch Addition des Abbrems-
FAA9 65 5F    ADC $5F       wertes wieder auf Normalmodus
FAAB 8D 05 1C STA $1C05     setzen
FAAE C6 60    DEC $60       Zähler für Bremsen vermindern
FAB0 D0 E2    BNE $FA94     weitermachen, wenn noch nicht Null
FAB2 A9 4E    LDA #$4E
FAB4 85 62    STA $62       Adresse auf $FA4E zur Beendigung
FAB6 A9 FA    LDA #$FA      des Steppermodus
FAB8 85 63    STA $63
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 $1C0C     herstellen
FAC6 60       RTS           zum Hauptinterruptprogramm ($FE7C)
```

```
------------------------------------------------------------------
FAC7                        Jobroutine zum Formatieren einer
                            Diskette. Diese Routine wird durch die
                            Formatierungsroutine ab $C8C6
                            initialisiert und über einen Vektor im
                            RAM bei $0600 (JMP $FAC7) angesprungen.
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 6 und 5 zum Bereitmachen des
FACF 95 20     STA $20,X    Laufwerks setzen
FAD1 A9 01     LDA #$01     Track 1 für Beginn an Jobschleife
FAD3 95 22     STA $22,X    übergeben und Track bzw. Flag für
FAD5 85 51     STA $51      laufende Formatierung setzen
FAD7 A9 A4     LDA #$A4     46 Tracks nach außen fahren (ent-
FAD9 85 4A     STA $4A      spricht -92 Schritten); BUMP
FADB AD 00 1C  LDA $1C00    Schrittphase für Kopfbewegung auf
FADE 29 FC     AND #$FC     00 (Bit 0 und 1) setzen
FAE0 8D 00 1C  STA $1C00
FAE3 A9 0A     LDA #$0A     10 Fehlversuche
FAE5 8D 20 06  STA $0620    erlauben; Zähler setzen
FAE8 A9 A0     LDA #$A0     erster Schätzwert für halbe Kapazi-
FAEA 8D 21 06  STA $0621    tät eines Tracks auf 4000 setzen
FAED A9 0F     LDA #$0F     ($0621/0622 = $0FA0)
FAEF 8D 22 06  STA $0622
FAF2 4C 9C F9  JMP $F99C    zur Jobschleife; Kopf positionieren

FAF5 A0 00     LDY #$00
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    zur Jobschleife; Kopf positionieren

FB00 AD 00 1C  LDA $1C00    Kontrollport 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    ($062l/$0622) mal $FF schreiben
FB12 A9 55     LDA #$55     'Leerbyte'
FB14 8D 01 1C  STA $1C01
FB17 20 C3 FD  JSR $FDC3    ($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
FB22 0D 0B 18  ORA $180B   Timer 1 auf 'free running mode'
FB25 8D 0B 18  STA $180B   setzen
FB28 A9 62     LDA #$62    98 Zyklen entsprechen ca. 0.1 ms
FB2A 8D 06 18  STA $1806   als Timerwert setzen
FB2D A9 00     LDA #$00
FB2F 8D 07 18  STA $1807   Timer Hi auf 0
FB32 8D 05 18  STA $1805   Timer starten
FB35 A0 00     LDY #$00    Zähler zurücksetzen
FB37 A2 00     LDX #$00
FB39 2C 00 1C  BIT $1C00   auf Beginn der SYNC-Zone warten
FB3C 30 FB     BMI $FB39
FB3E 2C 00 1C  BIT $1C00   auf Ende der SYNC-Zone warten
FB41 10 FB     BPL $FB3E
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   Bereiches
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 erhöhen
FB52 D0 EF     BNE $FB43
FB54 C8        INY
FB55 D0 EC     BNE $FB43
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-Bereiches merken
FB60 A2 00     LDX #$00    Zähler wieder zurücksetzen
FB62 A0 00     LDY #$00
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   Bereiches
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 erhöhen
FB73 D0 EF     BNE $FB64
FB75 C8        INY
FB76 D0 EC     BNE $FB64
FB78 A9 02     LDA #$02    Nummer der Fehlermeldung
FB7A 4C D3 FD  JMP $FDD3   '20, READ ERROR' ausgeben
FB7D 38        SEC
FB7E 8A        TXA         Differenz der beiden Meßwerte
```

```
FB7F E5 71     SBC $71
FB81 AA        TAX         bilden
FB82 85 70     STA $70
FB84 98        TYA
FB85 E5 72     SBC $72     und nach $71/72 abspeichern
FB87 A8        TAY
FB88 85 71     STA $71
FB8A 10 0B     BPL $FB97   verzweige, wenn Differenz größer 0
FB8C 49 FF     EOR #$FF
FB8E A8        TAY
FB8F 8A        TXA
FB90 49 FF     EOR #$FF    invertieren und Absolutwert
FB92 AA        TAX
FB93 E8        INX         bilden
FB94 D0 01     BNE $FB97
FB96 C8        INY
FB97 98        TYA
FB98 D0 04     BNE $FB9E   Differenz kleiner 4 ?
FB9A E0 04     CPX #$04
FB9C 90 18     BCC $FBB6   verzweige, wenn ja
FB9E 06 70     ASL $70
FBA0 26 71     ROL $71     Wert verdoppeln und zum Wert für
FBA2 18        CLC         die Länge der Bereiche ($0621/0622)
FBA3 A5 70     LDA $70     addieren
FBA5 6D 21 06  ADC $0621
FBA8 8D 21 06  STA $0621
FBAB A5 71     LDA $71
FBAD 6D 22 06  ADC $0622
FBB0 8D 22 06  STA $0622
FBB3 4C 0C FB  JMP $FB0C   Messung mit neuen Werten beginnen
FBB6 A2 00     LDX #$00    Zähler wieder zurücksetzen
FBB8 A0 00     LDY #$00
FBBA B8        CLV
FBBB AD 00 1C  LDA $1C00   liegt SYNC noch an?
FBBE 10 0E     BPL $FBCE   verzweige, wenn ja
FBC0 50 F9     BVC $FBBB   auf BYTE READY warten
FBC2 B8        CLV
FBC3 E8        INX         Zähler erhöhen
FBC4 D0 F5     BNE $FBBB
FBC6 C8        INY
FBC7 D0 F2     BNE $FBBB
FBC9 A9 03     LDA #$03    Nummer der Fehlermeldung
FBCB 4C D3 FD  JMP $FDD3   '21, READ ERROR' ausgeben

FBCE 8A        TXA
```

```
FBCF 0A       ASL         mal 2
FBD0 8D 25 06 STA $0625   und in $0624/0625 abspeichern
FBD3 98       TYA
FBD4 2A       ROL
FBD5 8D 24 06 STA $0624
FBD8 A9 BF    LDA #$BF
FBDA 2D 0B 18 AND $180B   Timer 1 anhalten
FBDD 8D 0B 18 STA $180B
FBE0 A9 66    LDA #$66    Gesamtanzahl aller Bytes berechnen,
FBE2 8D 26 06 STA $0626   die auf diesen Track passen und
FBE5 A6 43    LDX $43     somit geschrieben werden müssen
FBE7 A0 00    LDY #$00
FBE9 98       TYA
FBEA 18       CLC
FBEB 6D 26 06 ADC $0626
FBEE 90 01    BCC $FBF1
FBF0 C8       INY
FBF1 C8       INY
FBF2 CA       DEX
FBF3 D0 F5    BNE $FBEA
FBF5 49 FF    EOR #$FF
FBF7 38       SEC         Berechnung der Bytes in den Block—
FBF8 69 00    ADC #$00    zwischenräumen
FBFA 18       CLC
FBFB 6D 25 06 ADC $0625
FBFE B0 03    BCS $FC03
FC00 CE 24 06 DEC $0624
FC03 AA       TAX
FC04 98       TYA
FC05 49 FF    EOR #$FF
FC07 38       SEC
FC08 69 00    ADC #$00
FC0A 18       CLC
FC0B 6D 24 06 ADC $0624   Spurkapazität überschritten?
FC0E 10 05    BPL $FC15   verzweige, wenn nein
FC10 A9 04    LDA #$04    Nummer der Fehlermeldung
FC12 4C D3 FD JMP $FDD3   '22, READ ERROR' ausgeben
FC15 A8       TAY
FC16 8A       TXA
FC17 A2 00    LDX #$00    Gesamtanzahl der Bytes in den
FC19 38       SEC         Zwischenräumen geteilt durch die
FC1A E5 43    SBC $43     Anzahl der Sektoren ergibt die
FC1C B0 03    BCS $FC21   Anzahl der Bytes pro Zwischenraum.
FC1E 88       DEY
```

```
FC1F 30 03     BMI $FC24
FC21 E8        INX
FC22 D0 F5     BNE $FC19
FC24 8E 26 06  STX $0626   Zahl der Bytes merken
FC27 E0 04     CPX #$04    Anzahl kleiner als 4?
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
FC31 65 43     ADC $43     Rest der Division plus Anzahl der
FC33 8D 27 06  STA $0627   Sektoren merken
FC36 A9 00     LDA #$00
FC38 8D 28 06  STA $0628   Zähler für die Sektoren pro Track
FC3B A0 00     LDY #$00    Zeiger in Puffer
FC3D A6 3D     LDX $3D     Drivenummer für Job
FC3F A5 39     LDA $39     Kennzeichen $08 für Blockheader
FC41 99 00 03  STA $0300,Y in Puffer schreiben
FC44 C8        INY
FC45 C8        INY         1 Byte für Prüfsumme freilassen
FC46 AD 28 06  LDA $0628   Nummer des entsprechenden Sektors
FC49 99 00 03  STA $0300,Y in Puffer schreiben
FC4C C8        INY
FC4D A5 51     LDA $51     aktuelle Tracknummer
FC4F 99 00 03  STA $0300,Y in Puffer schreiben
FC52 C8        INY
FC53 B5 13     LDA $13,X   ID 2
FC55 99 00 03  STA $0300,Y in Puffer schreiben
FC58 C8        INY
FC59 B5 12     LDA $12,X   ID 1
FC5B 99 00 03  STA $0300,Y in Puffer schreiben
FC5E C8        INY
FC5F A9 0F     LDA #$0F    15 als Lückenwert
FC61 99 00 03  STA $0300,Y
FC64 C8        INY         zweimal in Puffer schreiben
FC65 99 00 03  STA $0300,Y
FC68 C8        INY
FC69 A9 00     LDA #$00
FC6B 59 FA 02  EOR $02FA,Y Prüfsumme für Blockheader berechnen
FC6E 59 FB 02  EOR $02FB,Y
FC71 59 FC 02  EOR $02FC,Y
FC74 59 FD 02  EOR $02FD,Y
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
```

```
FC80 C5 43     CMP $43     schon maximale Nummer erreicht?
FC82 90 BB     BCC $FC3F   nächster Header, wenn nein
FC84 98        TYA         Endeposition im Puffer
FC85 48        PHA         merken
FC86 E8        INX         00 + 1 (unnötig; eigentlich NOP)
FC87 8A        TXA
FC88 9D 00 05  STA $0500,X als 'dummy'—Wert in Puffer
FC8B E8        INX
FC8C D0 FA     BNE $FC88
FC8E A9 03     LDA #$03    Pufferadresse auf 40300
FC90 85 31     STA $31
FC92 20 30 FE  JSR $FE30   Pufferinhalt nach GCR umwandeln
FC95 68        PLA         Endeposition im Puffer zurückholen
FC96 A8        TAY
FC97 88        DEY         minus 1; Puffer Inhalt ab $0300,Y
FC98 20 E5 FD  JSR $FDE5   nach $0344,Y verschieben; GCR-Bytes
FC9B 20 F5 FD  JSR $FDF5   in frei gewordenen Bereich schieben
FC9E A9 05     LDA #$05
FCA0 85 31     STA $31     Pufferadresse auf $0500
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
FCAC 85 32     STA $32     Zeiger auf Blockheader setzen
FCAE 20 0E FE  JSR $FE0E   Track löschen; Schreibbetrieb!
FCB1 A9 FF     LDA #$FF
FCB3 8D 01 1C  STA $1C01   SYNC-Markierung auf Diskette
FCB6 A2 05     LDX #$05    schreiben
FCB8 50 FE     BVC $FCB8
FCBA B8        CLV
FCBB CA        DEX
FCBC D0 FA     BNE $FCB8
FCBE A2 0A     LDX #$0A
FCC0 A4 32     LDY $32     Blockheader aus Puffer auf Diskette
FCC2 50 FE     BVC $FCC2   schreiben (10 GCR-Bytes)
FCC4 B8        CLV
FCC5 B9 00 03  LDA $0300,Y
FCC8 8D 01 1C  STA $1C01
FCCB C8        INY
FCCC CA        DEX
FCCD D0 F3     BNE $FCC2
FCCF A2 09     LDX #$09    9 'Leerbytes' schreiben
FCD1 50 FE     BVC $FCD1   Lücke hinter dem Blockheader
FCD3 B8        CLV
```

```
FCD4 A9 55     LDA #$55
FCD6 8D 01 1C  STA $1C01
FCD9 CA        DEX
FCDA D0 F5     BNE $FCD1
FCDC A9 FF     LDA #$FF
FCDE A2 05     LDX #$05    SYNC-Markierung für Datenblock
FCE0 50 FE     BVC $FCE0   schreiben
FCE2 B8        CLV
FCE3 8D 01 1C  STA $1C01
FCE6 CA        DEX
FCE7 D0 F7     BNE $FCE0
FCE9 A2 BB     LDX #$BB
FCEB 50 FE     BVC $FCEB   ersten Teil des Datenblocks aus
FCED B8        CLV         dem Ausweichpuffer auf Diskette
FCEE BD 00 01  LDA $0100,X schreiben
FCF1 8D 01 1C  STA $1C01
FCF4 E8        INX
FCF5 D0 F4     BNE $FCEB
FCF7 A0 00     LDY #$00
FCF9 50 FE     BVC $FCF9   die restlichen 256 Bytes des
FCFB B8        CLV         Datenblocks ebenfalls auf Diskette
FCFC B1 30     LDA ($30),Y schreiben
FCFE 8D 01 1C  STA $1C01
FD01 C8        INY
FD02 D0 F5     BNE $FCF9
FD04 A9 55     LDA #$55
FD06 AE 26 06  LDX $0626   variable Lücke zwischen zwei
FD09 50 FE     BVC $FD09   Sektoren schreiben
FD0B B8        CLV
FD0C 8D 01 1C  STA $1C01
FD0F CA        DEX
FD10 D0 F7     BNE $FD09
FD12 A5 32     LDA $32     Zeiger auf Blockheader in Puffer
FD14 18        CLC
FD15 69 0A     ADC #$0A    auf nächsten Blockheader
FD17 85 32     STA $32
FD19 CE 28 06  DEC $0628   nächste Sektornummer
FD1C D0 93     BNE $FCB1   nächsten Sektor, nenn ungleich 0
FD1E 50 FE     BVC $FD1E
FD20 B8        CLV
FD21 50 FE     BVC $FD21   2 Bytes abwarten
FD23 B8        CLV
FD24 20 00 FE  JSR $FE00   auf Lesen umschalten
FD27 A9 C8     LDA #$C8    200 Leseversuche
```

```
FD29 8D 23 06  STA $0623   setzen
FD2C A9 00     LDA #$00
FD2E 85 30     STA $30
FD30 A9 03     LDA #$03    Pufferadresse auf $0300
FD32 85 31     STA $31
FD34 A5 43     LDA $43     Anzahl der Sektoren pro Track
FD36 8D 28 06  STA $0628   übernehmen
FD39 20 56 F5  JSR $F556   SYNC-Signal abwarten
FD3C A2 0A     LDX #$0A    10 Bytes für Blockheader
FD3E A0 00     LDY #$00
FD40 50 FE     BVC $FD40   einlesen
FD42 B8        CLV
FD43 AD 01 1C  LDA $1C01   und mit Pufferinhalt vergleichen
FD46 D1 30     CMP ($30),Y verzweige, wenn ungleich
FD48 D0 0E     BNE $FD58
FD4A C8        INY
FD4B CA        DEX         nächstes Byte vergleichen
FD4C D0 F2     BNE $FD40
FD4E 18        CLC         Pufferadresse
FD4F A5 30     LDA $30     um 10 erhöhen; zeigt auf nächsten
FD51 69 0A     ADC #$0A    Blockheader
FD53 85 30     STA $30     weitermachen; alles ok
FD55 4C 62 FD  JMP $FD62   Zähler für Leseversuche minus 1
FD58 CE 23 06  DEC $0623   weiter versuchen, wenn ungleich 0
FD5B D0 CF     BNE $FD2C   Nummer der Fehlermeldung
FD5D A9 06     LDA #$06    '24, READ ERROR' ausgeben
FD5F 4C D3 FD  JMP $FDD3   SYNC des Datenblocks abwarten
FD62 20 56 F5  JSR $F556
FD65 A0 BB     LDY #$BB    Bytes des Datenblocks aus Aus—
FD67 50 FE     BVC $FD67   weichpuffer mit Disketteninhalt
FD69 B8        CLV
FD6A AD 01 1C  LDA $1C01   vergleichen
FD6D D9 00 01  CMP $0100,Y erneuter Versuch, wenn ungleich
FD70 D0 E6     BNE $FD58
FD72 C8        INY
FD73 D0 F2     BNE $FD67
FD75 A2 FC     LDX #$FC    Bytes aus Datenpuffer mit Bytes
FD77 50 FE     BVC $FD77   auf Diskette vergleichen
FD79 B8        CLV
FD7A AD 01 1C  LDA $1C01
FD7D D9 00 05  CMP $0500,Y erneuter Versuch, wenn ungleich
FD80 D0 D6     BNE $FD58
FD82 C8        INY
FD83 CA        DEX
FD84 D0 F1     BNE $FD77
```

```
FD86 CE 28 06  DEC $0628   Anzahl der Sektoren minus 1
FD89 D0 AE     BNE $FD39   weitermachen, wenn ungleich 0
FD8B E6 51     INC $51     Tracknummer plus 1
FD8D A5 51     LDA $51
FD8F C9 24     CMP #$24    schon maximale Tracknummer (36)?
FD91 B0 03     BCS $FD96   Ende, wenn ja
FD93 4C 9C F9  JMP $F99C   in Jobschleife für nächsten Track
FD96 A9 FF     LDA #$FF
FD98 85 51     STA $51     Flags für Formatierung beendet
FD9A A9 00     LDA #$00
FD9C 85 50     STA $50
FD9E A9 01     LDA #$01    Nummer der Rückmeldung
FDA0 4C 69 F9  JMP $F969   '00, OK' ausgeben; Ende
-------------------------------------------------------------
FDA3                       Spur löschen durch Beschreiben mit
                           10240 $FF-Bytes.
FDA3 AD 0C 1C  LDA $1C0C
FDA6 29 1F     AND #$1F    PCR auf Schreibbetrieb umschalten
FDA8 09 C0     ORA #$C0
FDAA 8D 0C 1C  STA $1C0C
FDAD A9 FF     LDA #$FF    Port für Schreib-/Lesekopf
FDAF 8D 03 1C  STA $1C03   auf Ausgang
FDB2 8D 01 1C  STA $1C01   $FF zum Schreib—/Lesekopf
FDB5 A2 28     LDX #$28
FDB7 A0 00     LDY #$00
FDB9 50 FE     BVC $FDB9   40*256 mal schreiben
FDBB B8        CLV
FDBC 88        DEY
FDBD D0 FA     BNE $FDB9
FDBF CA        DEX
FDC0 D0 F7     BNE $FDB9
FDC2 60        RTS         Ende; Schreibmodus an
-------------------------------------------------------------
FDC3                       Eine in $0621/0622 gegebene Anzahl von
                           BYTE READY Signalen abwarten.
FDC3 AE 21 06  LDX $0621
FDC6 AC 22 06  LDY $0622
FDC9 50 FE     BVC $FDC9   BYTE READY abwarten
FDCB B8        CLV
FDCC CA        DEX
FDCD D0 FA     BNE $FDC9
FDCF 88        DEY
FDD0 10 F7     BPL $FDC9
FDD2 60        RTS
```

```
------------------------------------------------------------------
FDD3                        Fehlerausgang beim Formatieren.
FDD3 CE 20 06  DEC $0620    Versuchezähler minus 1
FDD6 F0 03     BEQ $FDDB    Fehler, wenn Null
FDD8 4C 9C F9  JMP $F99C    weitermachen
FDDB A0 FF     LDY #$FF     Flag für Ende der Formatierung
FDDD 84 51     STY $51      setzen
FDDF C8        INY
FDE0 84 50     STY $50      Track für Formatierung löschen
FDE2 4C 69 F9  JMP $F969    Ende; Fehlernummer in A
------------------------------------------------------------------
FDE5                        Schiebt die ersten 69 ($45) Bytes in
                            Puffer 0 um 69 ($45) nach oben, um am
                            Anfang Platz zu bekommen.
FDE5 B9 00 03  LDA $0300,Y
FDE8 99 45 03  STA $0345,Y
FDEB 88        DEY
FDEC D0 F7     BNE $FDE5
FDEE AD 00 03  LDA $0300
FDF1 8D 45 03  STA $0345
FDF4 60        RTS
------------------------------------------------------------------
FDF5                        Holt den Inhalt des Ausweichpuffers in
                            den aktuellen Puffer.
FDF5 A0 44     LDY #$44
FDF7 B9 BB 01  LDA $01BB,Y  Byte aus Ausweichpuffer
FDFA 91 30     STA ($30),Y  in Datenpuffer schreiben
FDFC 88        DEY
FDFD 10 F8     BPL $FDF7
FDFF 60        RTS
------------------------------------------------------------------
FE00                        Schaltet das PCR des DC vom Schreib— in
                            den Lesemodus um.
FE00 AD 0C 1C  LDA $1C0C
FE03 09 E0     ORA #$E0
FE05 8D 0C 1C  STA $1C0C
FE08 A9 00     LDA #$00     Port für Schreib—/Lesekopf
FE0A 8D 03 1C  STA $1C03    auf Eingang
FE0D 60        RTS
------------------------------------------------------------------
FE0E                        Überschreibt eine Spur mit 10240 $55
                            Bytes.
FE0E AD 0C 1C  LDA $1C0C
FE11 29 1F     AND #$1F     PCR auf Schreibbetrieb umschalten
```

```
FE13 09 C0     ORA #$C0
FE15 8D 0C 1C  STA $1C0C
FE18 A9 FF     LDA #$FF    Fort für Schreib-/Lesebetrieb
FE1A 8D 03 1C  STA $1C03   auf Ausgang
FE1D A9 55     LDA #$55
FE1F 8D 01 1C  STA $1C01   Byte zum Tonkopf
FE22 A2 28     LDX #$28
FE24 A0 00     LDY #$00
FE26 50 FE     BVC $FE26   und 40*256 mal schreiben
FE28 B8        CLV
FE29 88        DEY
FE2A D0 FA     BNE $FE26
FE2C CA        DEX
FE2D D0 F7     BNE $FE26
FE2F 60        RTS
-----------------------------------------------------------------
FE30                       Wandelt die Bytes des Blockheaders der
                           Binärform in die GCR—Codierung um und
                           schreibt diese Werte in den
                           Ausweichspeicher ($01BB-$01FF)
FE30 A9 00     LDA #$00
FE32 85 30     STA $30
FE34 85 2E     STA $2E
FE36 85 36     STA $36
FE38 A9 BB     LDA #$BB
FE3A 85 34     STA $34
FE3C A5 31     LDA $31     Pufferadresse Hi
FE3E 85 2F     STA $2F     übernehmen
FE40 A9 01     LDA #$01
FE42 85 31     STA $31
FE44 A4 36     LDY $36
FE46 B1 2E     LDA ($2E),Y
FE48 85 52     STA $52
FE4A C8        INY
FE4B B1 2E     LDA ($2E),Y
FE4D 85 53     STA $53
FE4F C8        INY
FE50 B1 2E     LDA ($2E),Y
FE52 85 54     STA $54
FE54 C8        INY
FE55 B1 2E     LDA ($2E),Y
FE57 85 55     STA $55
FE59 C8        INY
FE5A F0 08     BEQ $FE64
```

```
FE5C 84 36     STY $36
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 umwandeln
-------------------------------------------------------------
FE67                       System IRQ-Routine; wird direkt über
                           den Hardwarevektor angesprungen.
FE67 48        PHA
FE68 8A        TXA
FE69 48        PHA
FE6A 98        TYA         Prozessorregister retten
FE6B 48        PHA
FE6C AD 0D 18  LDA $180D   ATN—Signal vom seriellen Bus?
FE6F 29 02     AND #$02
FE71 F0 03     BEQ $FE76   verzweige, wenn nein
FE73 20 53 E8  JSR $E853   ATN-Flags setzen; RTS
FE76 AD 0D 1C  LDA $1C0D   Interrupt durch Timer 1?
FE79 0A        ASL
FE7A 10 03     BPL $FE7F   verzweige, wenn nein
FE7C 20 B0 F2  JSR $F2B0   IRQ-Routine für Disk-Controller
FE7F 68        PLA
FE80 A8        TAY
FE81 68        PLA         Prozessorregister zurückholen
FE82 AA        TAX
FE83 68        PLA
FE84 40        RTI
-------------------------------------------------------------
FE85 12                    Spurnummer des Directory (18)
FE86 04                    Anzahl der Bytes/Spur in der BAM
FE87 04                    BAM steht in 18,0 ab Position 4
FE88 90                    Anfang des Diskettennamens in Block
                           18,0 bei Position 144
-------------------------------------------------------------
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
FE91 43                    C = COPY Befehl
FE92 52                    R = RENAME Befehl
```

```
FE93 53                  S = SCRATCH Befehl
FE94 4E                  N = NEW oder HEADER Befehl
-------------------------------------------------------------
Lo Byte  Hi Byte         Adressen der Befehle
FE95 84  FEA1 ED         V = VALIDATE
FE96 05  FEA2 D0         I = INITIALIZE
FE97 C1  FEA3 C8         D = DUPLICATE (n.v.)
FE98 F8  FEA4 CA         M = MEMORY
FE99 1B  FEA5 CC         B = BLOCK
FE9A 5C  FEA6 CB         U = USER
FE9B 07  FEA7 E2         P = POSITION
FE9C A3  FEA8 E7         & = & Befehl
FE9D F0  FEA9 C8         C = COPY
FE9E 88  FEAA CA         R = RENAME
FE9F 23  FEAB C8         S = SCRATCH
FEA0 0D  FEAC EE         N = NEW
-------------------------------------------------------------
                 Bitmuster für Befehle
FEAD 51   %01010001      Kopieren einer Diskette
FEAE DD   %11011101      Umbenennen eines Files
FEAF 1C   %00011100      Löschen eines Files
FEB0 9E   %10011110      Formstieren einer Diskette
FEB1 1C   %00011100      Laden eines Files
-------------------------------------------------------------
                 Bytes der Betriebsarten
FEB2 52                  R = READ; Lesen eines Files
FEB3 57                  W = WRITE; Schreiben eines Files
FEB4 41                  A = APPEND; Anhängen von Daten
FEB5 4D                  M = MODIFY; Lesen eines
                         offengebliebenen Files
-------------------------------------------------------------
                 Kürzel der Filetypen
FEB6 44                  D = DELETED File
FEB7 53                  S = SEQUENTIAL File
FEB8 50                  P = PROGRAM File
FEB9 55                  U = USER File
FEBA 4C                  L = RELATIVE File
-------------------------------------------------------------
                 Bezeichnung der Filetypen im Directory
FEBB 44  FEC0 45  FEC5 4C     DEL
FEBC 53  FEC1 45  FEC6 51     SEQ
FEBD 50  FEC2 52  FEC7 47     PRG
FEBE 55  FEC3 53  FEC8 52     USR
FEBF 52  FEC4 45  FEC9 4C     REL
```

```
-----------------------------------------------------------------
FECA 08                  LED Maskenbyte für Laufwerk 0
FECB 00                  LED Maskenbyte für Laufwerk 1(n.v.)
-----------------------------------------------------------------
          Wert für Fehlermeldungen bei Bit Kommandos
FECC 00
FECD 3F
FECE 7F
FECF BF
FED0 FF
-----------------------------------------------------------------
              Anzahl der Sektoren pro Spur
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 41                  Formatkennzeichen der DOS—Version
FED6 04                  Anzahl der Zonen auf Diskette mit
                         unterschiedlichen Sektorzahlen pro Spur
-----------------------------------------------------------------
          Spurnummern bei denen ein Zonenwechsel 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)
-----------------------------------------------------------------
          Steuerbytes für die Kopfdejustierung bei Leseproblemen
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
-----------------------------------------------------------------
                  Hi Bytes der Adressen der Puffer
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 FD                  Prüfsumme für ROM Version
```

```
-------------------------------------------------------------
FEE7 6C 65 00  JMP ($0065) Sprungbefehl vom NMI
-------------------------------------------------------------
FEEA 8D 00 1C  STA $1C00   LED einschalten
FEED 8D 02 1C  STA $1C02   Port A auf Ausgabe
FEF0 4C 7D EA  JMP $EA7D   Zurück zur RESET-Routine
-------------------------------------------------------------
FEF3                       Verzögerung für seriellen Bus
FEF3 8A        TXA
FEF4 A2 05     LDX #$05
FEF6 CA        DEX
FEF7 D0 FD     BNE $FEF6
FEF9 AA        TAX
FEFA 60        RTS
-------------------------------------------------------------
FEFB                       Unbenutzter Programmrest
FEFB 20 AE E9  JSR $E9AE
FEFE 4C 9C E9  JMP $E99C
-------------------------------------------------------------
FF01                       Einsprung vom UI Befehl
FF01 AD 02 02  LDA $0202
FF04 C9 2D     CMP #$2D
FF06 F0 05     BEQ $FF0D
FF08 38        SEC
FF09 E9 2B     SBC #$2B
FF0B D0 DA     BNE $FEE7
FF0D 85 23     STA $23
FF0F 60        RTS
-------------------------------------------------------------
FF10                       Dieser Bereich ist bei Floppies mit den
                           ROMs der Serie -03 leer. Die neuen ROMs
                           -05 enthalten folgende Programmteile:
FF10 8E 03 18  STX $1803
FF13 A9 02     LDA #$02    RESET für seriellen Bus
FF15 8D 00 18  STA $1800
FF18 A9 1A     LDA #$1A
FF1A 8D 02 18  STA $1802
FF1D 4C A7 EA  JMP $EAA7   zurück zur RESET-Routine
-------------------------------------------------------------
FF20 AD 00 18  LDA $1800   Warten auf DATA IN Hi
FF23 29 01     AND #$01
FF25 D0 F9     BNE $FF20
FF27 A9 01     LDA #$01    Timer für Bus zurücksetzen
```

```
FF29 8D 05 18  STA $1805
FF2C 4C DF E9  JMP $E9DF   zurück zur Busbedienung
-----------------------------------------------------------------
FF2F AA ...                Leerbereich; wird nicht benutzt
FFE5 ... AA
-----------------------------------------------------------------
              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
FFF0 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      $EAA0
FFFE 67 FE                 IRQ-Vektor                      $FE67
-----------------------------------------------------------------
```

# Anhang III
# Liste des gesamten Befehlssatzes mit Kurzbeschreibung

# Anhang III: Liste des Befehlssatzes der 1541

## *Der "$"-Befehl*

dient dem Laden eines Directory von der Diskette. Er wird zusammen mit einem LOAD-Befehl zur Floppy geschickt, wobei die Syntax folgendermaßen lautet: LOAD"$0",8 oder LOAD"$1",8. Hinter dem '$'-Zeichen wird also die Drivenummer angegeben. Sie kann beim Betrieb der 1541 entfallen, da hier nur ein Laufwerk zur Verfügung steht.

## *Der N-Befehl*

dient dem Formatieren einer neuen Diskette. Will man auf einer schon einmal formatierten Diskette nur sämtliche Programme löschen, so läßt man die ID und das Komma im Befehlsstring einfach weg.

"N:diskname,id"

diskname - Name der neuen Diskette (max. 16 Zeichen)
Id - Diskettenidentifikation (besteht aus 2 Zeichen)

## *Der V-Befehl*

dient dem 'Aufräumen' einer Diskette. Hierbei werden alle Blöcke auf der Diskette, die keinem Filenamen im Directory zugeordnet sind, in der BAM wieder freigegeben.

## *Der S-Befehl*

dient dem Löschen eines Files im Directory. Wird als Filename ein '*' eingegeben, so werden alle Files auf der Diskette geloscht. Steht nach einem oder mehreren Buchstaben ein '*', werden alle Files geloscht, die mit der angegebenen Zeichenfolge beginnen.

"S:filename"

filename  - Name des/der zu löschenden Files

## Der I-Befehl

Mit dem I-Befehl kann eine Diskette 'von Hand" initialisiert werden. Dieser Befehl ist nur beim Direktzugriff auf die Diskette sinnvoll, da er sonst automatisch beim Diskettenwechsel erfolgt.

## Der R-Befehl

dient dem Umbenennen eines Files. Seine Syntax lautet:

"R:neuer filename=alter filename"

filename  - Name des entsprechenden Files auf Diskette

## Der C-Befehl

Dieser Befehl ist normalerweise nur auf Diskettenstationen mit zwei Laufwerken sinnvoll. Er gestattet jedoch bei der 1541 das Duplizieren einer Datei, zum Beispiel um eine Sicherheitskopie anzufertigen.

"C:neuer filename=alter filename"

filename  - Name des entsprechenden Files auf Diskette

## Der D-Befehl

dient normalerweise der Anfertigung eines Backups auf Doppellaufwerken. Er ist bei der 1541 nicht implementiert.

## Der P-Befehl

dient dem Positionieren auf einen bestimmten Datensatz (Record) in einer relativen Datei.

```
"P"CHR$(Kanal#) CHR$(NrLow) CHR$(NrHigh) CHR$(Stelle)

Kanal#    - Kanalnummer der Datei (2-14)
NrLow     - niederwertiges Byte der Recordnummer
NrHigh    - höherwertiges Byte der Recordnummer
Stelle    - Nummer des Bytes innerhalb des Records
```

## Der '#'-Befehl

dient dem Eröffnen eines Direktzugriffskanals, um einzelne Blöcke auf der Diskette zu lesen oder zu beschreiben (siehe Kapitel 4.2.1).

```
OPEN File#, Geräte#, Kanal#, "#"

File#     - logische Filenummer (0-127)
Geräte#   - Gerätenummer der Floppy (8)
Kanal#    - Nummer für Direktzugriffskanal (2-14)
```

## Der BLOCK-READ-(U1)-Befehl

dient dem Lesen eines Blocks in den reservierten Pufferspeicher der Floppy (siehe Kapitel 4.2.2).

```
"U1" Kanal# Laufwerk# Track# Sektor#

Kanal#    - Nummer des Direktzugriffskanals
Laufwerk# - Drivenummer (bei der 1541 immer 0)
Track#    - Tracknummer des Blocks
Sektor#   - Sektornummer des Blocks
```

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

dient dem Positionieren auf ein bestimmtes Byte des reservierten Pufferspeichers (siehe Kapitel 4.2.3).

"B-P" Kanal# Position#

Kanal# - Nummer des Direktzugriffskanals
Position# - Nummer des Bytes, auf das positioniert werden soll

## Der BLOCK-WRITE-(U2)-Befehl

dient dem Schreiben eines Blocks aus dem reservierten Pufferspeicher direkt auf Diskette (siehe Kapitel 4.2.4).

"U2" Kanal# Laufwerk# Track# Sektor#

Kanal# - Nummer des Direktzugriffskanals
Laufwerk# - Drivenummer (bei der 1541 immer 0)
Track# - Tracknummer des Blocks
Sektor# - Sektornummer des Blocks

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

dient dem Belegen eines soeben neu beschriebenen Blocks, damit beim nächsten Schreibzugriff auf Diskette kein überschreiben stattfinden kann (siehe Kapitel 4.2.5).

"B-A" Laufwerk# Track# Sektor#

Laufwerk# - Drivenummer (bei der 1541 immer 0)
Track# - Tracknummer des Blocks
Sektor# - Sektornummer des Blocks

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

dient dem Freigeben eines bereits belegten Blocks in der BAM (siehe Kapitel 4.2.6).

"B-F" Laufwerke Track# Sektorf

Laufwerk# - Drivenummer (bei der 1541 immer 0)
Track# - Tracknummer des Blocks
Sektor# - Sektornummer des Blocks

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

dient dem Lesen von Speicherinhalten der Floppy (siehe Kapitel 4.2.7).

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

ADL - niederwertiges Byte der Speicheradresse
ADH - höherwertiges Byte der Speicheradresse
Anzahl - Anzahl der Bytes, die gelesen werden sollen

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

dient dem Schreiben von Bytes in den Speicher der Floppy, wobei auch mehrere Bytes hintereinander geschrieben werden können (siehe Kapitel 4.2.8).

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

ADL - niederwertiges Byte der Speicheradresse
ADH - höherwertiges Byte der Speicheradresse
Anzahl - Anzahl der Bytes, die gelesen werden sollen
DATA - Datenbyte, das geschrieben werden soll

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

dient der Ausführung eines Programms *im* Speicher der Floppy und zwar sowohl im ROM als auch im RAM (siehe Kapitel 4.2.9).

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

ADL       - niederwertiges Byte der Speicheradresse
ADH       - höherwertiges Byte der Speicheradresse
```

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

dient dem Starten eines Programms in einem Block, der zuvor von der Diskette geladen wird (siehe Kapitel 4.2.10).

```
"B-E" Kanal# Laufwerk# Track# Sektor#

Kanal#    - Nummer des Direktzugriffskanals
Laufwerk# - Drivenummer (bei der 1541 immer 0)
Track#    - Tracknummer des Blocks
Sektor#   - Sektornummer des Blocks
```

## Die USER-(U)-Befehle

dienen dem Programmstart bei bestimmten, oft verwendeten Stellen, die deshalb als Vektoren fest gespeichert sind (siehe Kapitel 4.3).

## Der '&'-Befehl

dient, ähnlich dem B-E-Befehl, dem Ausfuhren eines Programms im Pufferspeicher der Floppy, nachdem das Programm vorher als '&'-File von der Diskette gelesen wurde (siehe Kapitel 4.4).

# Anhang IV
# Liste der Fehlermeldungen des DOS 2.6

# Anhang IV: Liste der Fehlermeldungen des DOS 2.6

00 OK (alles in Ordnung);

Diese Meldung wird vom DOS immer dann geschickt, wenn ein Befehl einwandfrei und fehlerlos ausgeführt werden konnte.

01 FILES SCRATCHED (keine Fehlermeldung):

Diese Meldung wird nach jedem SCRATCH-Befehl ausgegeben und gibt in einer weiteren Nummer die Anzahl der geSCRATCHten Files an. Es handelt sich hier um eine Kontrollmeldung.

20 READ ERROR (Blockheader nicht gefunden):

Der Header eines Datenblocks konnte vom Disk-Controller nicht ausfindig gemacht werden. Das kann bei der Angabe einer ungültigen Sektornummer oder bei einem zerstörten Blockheader passieren.

21 READ ERROR (SYNC-Markierung nicht rechtzeitig gefunden):

Es konnte innerhalb der Toleranzzeit keine SYNC-Markierung auf dem gewünschten Track gefunden werden. Dieser Fehler tritt bei einer defekten Diskette oder einem dejustierten Tonkopf bevorzugt auf. Auch eine nicht formatierte Diskette kann für diesen Fehler verantwortlich sein.

22 READ ERROR (Datenblock nicht gefunden):

Bei dieser Meldung konnte der Datenblock hinter dein Header nicht identifiziert werden. Hier handelt es sich zumeist um eine Fehlformatierung des entsprechenden Blocks, wobei das Datenblockkennzeichen nicht mit dem Wert in Speicherstelle $38 der Zeropage übereinstimmt.

23 READ ERROR (Prüfsummenfehler im Datenblock):

Hier stimmt die Prüfsumme über den Datenblock nicht mit den gelesenen Werten überein. Der Block konnte zwar in den DOS-Puffer gelesen werden; es besteht jedoch die Gefahr einer defekten Diskette. Auch Erdungsprobleme der Floppy können durch diese Fehlermeldung angezeigt werden.

24 READ ERROR (Fehler bei der GCR-Recodierung);

Beim Recodieren des Datenblocks sind ungültige Werte aufgetreten, die die Leseelektronik nicht einwandfrei verarbeiten kann (mehr als 9 '1'-Bits oder mehr als 2 "0"-Bits). Auch diese Fehlermeldung kann Erdungsprobleme anzeigen.

25 WRITE ERROR (Fehler beim Verifizieren):

Bei der Vergleichskontrolle eines eben geschriebenen Blocks mit dem Pufferinhalt wurde ein Fehler entdeckt. Diese Meldung ist mit großer Wahrscheinlichkeit auf eine defekte Diskette zurückzuführen.

26 WRITE PROTECT ON:

Hier wurde versucht, auf eine Diskette zu schreiben, die eine Schreibschutzplakette trägt.

27 READ ERROR (Prüfsummenfehler im Blockheader):

Es wurde bei der Überprüfung der Headerprüfsumme ein Fehler entdeckt. Hier handelt es sich sehr wahrscheinlich um eine defekte Diskette oder um Erdungsprobleme.

28 WRITE ERROR (zu langer Block):

Hier wurde nach dem Schreiben eines Datenblocks in einer festgelegten Zeit nicht die SYNC-Markierung des nächsten Blockheaders gefunden. Diese Meldung zeigt eine fehlformatierte Diskette (Datenblock hat folgende SYNC-Markierung überschrieben) oder einen Hardware-Defekt an.

29 DISK ID MISMATCH:

Es wurde auf eine Diskette zugegriffen, die vorher nicht initalisiert worden ist. Bei dieser Fehlermeldung kann es sich aber auch um die Folge eines zerstörten Blockheaders handeln.

30 SYNTAX ERROR (allgemeiner Syntaxfehler):

Hier konnte ein gesendetes Kommando nicht interpretiert werden, weil entweder die Parameterübergabe falsch war oder das gegebene Kommando nicht exisitiert.

31 SYNTAX ERROR (ungültiger Befehl):

Das DOS war hier nicht in der Lage, einen Befehl zu erkennen. Das kann unter Umständen auf Leerzeichen vor dem Kommandowort hindeuten (Befehl muß immer an erster Position stehen).

32 SYNTAX ERROR (Befehlszeile zu lang):

Hier wurde eine Zeile mit einer Länge von mehr als 58 Zeichen zur Floppy geschickt.

33 SYNTAX ERROR (unerlaubte Verwendung eines 'Jokers'):

Es wurde ein Joker in einem Filenamen verwendet, obwohl das für den gewünschten Befehl nicht zulässig ist.

34 SYNTAX ERROR (Filename nicht gefunden):

Das DOS konnte in der Eingabezeile keinen Filenamen finden. Hierfür ist oft das Fehlen eines Doppelpunkts ':' verantwortlich.

39 SYNTAX ERROR (ungültiger Befehl):

Dieser Fehler kann auftreten, wenn ein Befehl, der über den Kommandokanal geschickt worden ist, vom DOS nicht interpretiert werden konnte.

50 RECORD NOT PRESENT:

Diese Fehlermeldung zeigt an, daß in einer relativen Datei über den letzten Record hinaus positioniert worden ist. Ist der nächste Dateizugriff ein Schreibzugriff, kann diese Meldung ignoriert werden, da eine Erweiterung automatisch stattfindet.

Eine weitere Funktion dieser Meldung zeigt sich bei den &-Files. Hier wird auf eine verkehrte Prüf summe hingewiesen.

51 OVERFLOW IN RECORD:

Hier wurde der Versuch unternommen, in einen Datensatz mehr als die zulässige Zeichenanzahl hineinzuschreiben.

Bei der Behandlung von &-Files bedeutet diese Meldung eine falsche Angabe über die Anzahl der Bytes im nächsten Abschnitt.

52 FILE TOO LARGE:

Eine Positionierung innerhalb einer relativen Datei zu deren Erweiterung ist nicht mehr möglich, da die Diskette voll ist.

60 WRITE FILE OPEN:

Diese Fehlermeldung erscheint, wenn ein Lesezugriff auf ein nicht ordnungsgemäß geschlossenes File stattfindet.

61 FILE NOT OPEN:

Es wird auf ein File zugegriffen, das vorher vom DOS nicht geöffnet worden war.

62 FILE NOT FOUND:

Das zum Lesen angeforderte File ist auf dem aktuellen Laufwerk nicht vorhanden.

63 FILE EXISTS:

Es soll ein File zum Schreiben geöffnet werden, das bereits auf Diskette existiert.

64 FILE TYPE MISMATCH:

Der angeforderte Filetyp stimmt nicht mit dem Filetyp im Directory überein.

65 NO BLOCK:

Diese Fehlermeldung zeigt an, daß ein Block mit dem B-A-Befehl belegt werden sollte, der bereits belegt ist. Die Track- und Sektornummer geben den nächsten freien Block an und sind 0, wenn alle höheren Blöcke bereits belegt sind.

66 ILLEGAL TRACK OR SECTOR:

Hier wurde versucht, auf einen ungültigen Track oder einen ungültigen Sektor zuzugreifen.

67 ILLEGAL TRACK OR SECTOR:

Hier zeigte der Linker eines Datenblocks auf eine ungültige Spur- oder Sektornummer.

70 NO CHANNEL:

Hier sind entweder alle Kanäle belegt, oder es wurde versucht, einen schon belegten Kanal zu reservieren.

71 DIR ERROR;

Hier stimmt das Verhältnis zwischen der Angabe der gesamten freien Blöcke und der Summe der freien Blöcke jedes Tracks in der BAM nicht. Dieser Fehler zeigt eine zerstörte BAM an.

72 DISK FULL:

Diese Fehlermeldung erscheint entweder, wenn alle Blöcke der Diskette belegt sind, oder wenn das Directory voll ist (144 Einträge bei der 1541).

73 CBM DOS V2.6 1541 (Einschalt- oder Fehlermeldung):

Es wurde versucht, mit einer Diskette anderen Formats zu arbeiten, oder die Floppy wurde soeben neu eingeschaltet.

74 DRIVE NOT READY:

Es wurde versucht, auf ein Laufwerk zuzugreifen, in dem sich keine Diskette befindet.

# Anhang V
# Programmlistings

```
10 rem steuerzeichen des commodore 64
20 rem tabelle dient der erleichterung
30 rem beim eintippen der listings.
40 rem
50 rem 'c=' ist die commodore-taste auf
60 rem      der tastatur links unten
70 rem
80 rem
90 rem
100 "{blk}" = ctrl + 1
110 "{wht}" = ctrl + 2
120 "{red}" = ctrl + 3
130 "{cyn}" = ctrl + 4
140 "{pur}" = ctrl + 5
150 "{grn}" = ctrl + 6
160 "{blu}" = ctrl + 7
170 "{yel}" = ctrl + 8
180 "{rvon}" = ctrl + 9
190 "{rvof}" = ctrl + 0
200 "{orng}" = c=   + 1
210 "{brn}" = c=   + 2
220 "{lred}" = c=   + 3
230 "{gry1}" = c=   + 4
240 "{gry2}" = c=   + 5
250 "{lgrn}" = c=   + 6
260 "{lblu}" = c=   + 7
270 "{gry3}" = c=   + 8
280 "{home}" = home
290 "{clr}" = clr/home
300 "{down}" = crsr down
310 "{up}" = crsr up
320 "{rght}" = crsr right
330 "{left}" = crsr left
340 "{$03}" = stop
350 "{F1}" = f1
360 "{F3}" = f3
370 "{F5}" = f5
380 "{F7}" = f7
390 "{F2}" = f2
400 "{F4}" = f4
410 "{F6}" = f6
420 "{F8}" = f8
430 "{del}" = del
```

```
10 rem verwaltung eines rezeptbuches
20 rem     als beispiel fuer die
30 rem   sequentielle und relative
40 rem       datenspeicherung
50 rem
60 rem (w) 1985 by karsten schramm
70 rem
80 rem
90 rem
100 print"{clr}{grn}";chr$(14):poke53280,0:poke53281,0
110 print:print"{clr}      **** Rezeptbuch 64 ****"
120 print:print:print"Hauptmenue:"
130 print"{CBM-Y}{CBM-Y}{CBM-Y}{CBM-Y}{CBM-Y}{CBM-Y}{CBM-Y}{CBM-Y}{CBM-Y}{CBM-
Y} "
140 print:print:print
145 print"{rvon} 1 {rvof}  Datei neu anlegen    ":print
150 print"{rvon} 2 {rvof}  Neues Rezept eingeben":print
160 print"{rvon} 3 {rvof}  Rezept einlesen      ":print
170 print"{rvon} 4 {rvof}  Rezept loeschen      ":print
180 print"{rvon} E {rvof}  Programmende         ":print
190 geta$:ifa$="e"then60000
200 ifa$<"1"ora$>"4"then190
210 onval(a$)gosub 1000,2000,3000,4000
220 goto110
230 :
1000 rem
1010 rem datei neu anlegen
1020 rem
1030 print"{clr}Neues Anlegen der Datei:":print:print
1040 print"ACHTUNG! Alle bisherigen Rezepte werden":print
1050 print"geloescht!":print:print
1060 print"Sind Sie sicher?"
1070 geta$:ifa$<>"j"anda$<>"n"then1070
1080 ifa$="j"then1100
1090 return
1100 print"{clr}Neues Anlegen der Datei:":print:print
1110 z=100
1120 rem
1130 print:input"Nummer der Datei";a$
1140 n=val(a$)
1150 ifn<0orn>1000then1130
1155 gosub20000
1160 open1,8,2,"inhalt"+str$(n)+",l,"+chr$(41)
```

```
1170 open2,8,15
1180 print#2,"p"+chr$(2)chr$(z)chr$(0)chr$(1)
1190 print#1,chr$(255)
1200 get#2,a$:ifst<>64then1200
1210 close1:close2
1220 return
2000 rem
2010 rem neues rezept eingeben
2020 rem
2030 print"{clr}Eingabe eines neuen Rezepts:":print:print
2040 print"Name des Rezeps (max. 16 Zeichen):":print
2050 inputr$:iflen(r$)<1orlen(r$)>16then2030
2060 print:input"Nummer der Datei zum Ablegen";n$
2070 n=val(n$):ifn<1orn>1000then2060
2080 print:print:print"Alles richtig?"
2090 geta$:ifa$<>"j"anda$<>"n"then2090
2100 ifa$="n"then2030
2102 gosub10000
2110 open1,8,2,r$+",s,w"
2120 print"{clr}Geben Sie jetzt das Rezept ein!":print
2130 print"Bei jedem Zeilenende 'RETURN' druecken!":print
2135 print"'|' heisst ENDE!":print
2140 print"{CBM-T}{CBM-T}{CBM-T}{CBM-T}{CBM-T}{CBM-T}{CBM-T}{CBM-T}{CBM-
T}{CBM-T}{CBM-T}{CBM-T}{CBM-T}{CBM-T}{CBM-T}{CBM-T}{CBM-T}{CBM-T}{CBM-T}{CBM-
T}{CBM-T}{CBM-T}{CBM-T}{CBM-T}{CBM-T}{CBM-T}{CBM-T}{CBM-T}{CBM-T}{CBM-T}{CBM-
T}{CBM-T}{CBM-T}{CBM-T}{CBM-T}{CBM-T}{CBM-T}{CBM-T}{CBM-T}{CBM-T}";:print
2150 inputa$:ifa$="|"then2180
2160 print#1,a$
2170 goto2150
2180 close1
2190 open1,8,2,"inhalt"+str$(n)+",l,"+chr$(41)
2200 open2,8,15
2210 print#2,"p"+chr$(2)chr$(p)chr$(0)chr$(1)
2220 print#1,r$
2230 close1:close2:return
3000 rem
3010 rem rezept einlesen
3020 rem
3030 print"{clr}Einlesen eines Rezepts:":print:print
3040 input"Dateinummer";n$
3050 n=val(n$):ifn<1orn>1000then3030
3060 print:print:print"Eingabe von {rvon}Z{rvof}ahl oder {rvon}N{rvof}ame ?"
3070 geta$:ifa$<>"z"anda$<>"n"then3070
3080 print:print:ifa$="n"then3180
3090 input"Nummer des Rezepts";x
3100 ifx<1orx>100then3090
```

```
3110 open1,8,2,"inhalt"+str$(n)+",l,"+chr$(41)
3120 open2,8,15
3130 print#2,"p"+chr$(2)+chr$(x)+chr$(0)+chr$(1)
3140 get#1,a$:ifa$<>chr$(255)then3150
3142 print:print"Diese Nummer ist nicht belegt!"
3144 print:print"Mit 'R' zum Menue":a$=""
3146 geta$:ifa$<>"r"then3146
3148 close1:close2:return
3150 print#2,"p"+chr$(2)+chr$(x)+chr$(0)+chr$(1)
3160 input#1,a$
3170 close1:close2:goto3200
3180 print:print:input"Name des Rezepts";a$
3190 iflen(a$)<1then3180
3200 open1,8,2,a$+",s,r"
3202 open2,8,15
3204 get#2,m$:ifm$<>"0"thenclose1:close2:print"File not found":goto3144
3210 print"{clr}Zum Weiterlesen Taste druecken!":print:print"Rezept:
";a$:print
3220 rem einlesen
3230 ifst=64then3270
3240 get#1,x$:printx$;
3250 geta$:ifa$=""then3250
3260 goto3230
3270 print:print:print"Mit 'R' zurueck zum Menue"
3280 geta$:ifa$<>"r"then3280
3290 close1:close2:return
4000 rem
4010 rem rezept loeschen
4020 rem
4030 print"{clr}Loeschen eines Rezepts":print:print
4040 input"Dateinummer";n
4050 ifn<1orn>1000then4040
4060 print:print:input"Name des Rezepts";r$
4070 iflen(r$)<1thenreturn
4080 print:print"Sind Sie sicher?"
4090 geta$:ifa$<>"j"anda$<>"n"then4090
4100 ifa$="j"then4120
4110 return
4120 open1,8,15,"s:"+r$:close1
4130 open1,8,2,"inhalt"+str$(n)+",l,"+chr$(41)
4140 open2,8,15
4150 forx=1to100
4160 print#2,"p"+chr$(2)+chr$(x)+chr$(0)+chr$(1)
```

```
4170 get#1,x$:ifa$=chr$(255)then4200
4180 print#2,"p"+chr$(2)+chr$(x)+chr$(0)+chr$(1)
4190 input#1,x$:ifx$=r$then4220
4200 nextx
4210 print:print:print"Datei existiert nicht !!!":goto3144
4220 print#2,"p"+chr$(2)+chr$(x)+chr$(0)+chr$(1)
4230 print#1,chr$(255)+chr$(255)+chr$(255)
4240 close1:close2
4250 return
10000 rem
10010 rem auf volle datei pruefen
10020 rem
10030 open1,8,2,"inhalt"+str$(n)+",l,"+chr$(41)
10040 open2,8,15
10050 forx=1to100
10060 print#2,"p"+chr$(2)chr$(x)chr$(0)chr$(1)
10070 get#1,a$:ifa$=chr$(255)then10130
10080 nextx
10090 print"{clr}Datei voll; keine Eingabe mehr moeglich"
10100 print:print"Mit 'R' zurueck ins Menue"
10110 geta$:ifa$<>"r"then10110
10120 close1:close2:run
10130 p=x:close1:close2:return
20000 rem
20010 rem datei loeschen
20020 rem
20030 open1,8,2,"inhalt"+str$(n)+",l,"+chr$(41)
20040 open2,8,15
20045 print#2,"p"+chr$(2)+chr$(100)+chr$(0)+chr$(1):print#1,chr$(255)
20050 forx=1to100
20060 print#2,"p"+chr$(2)+chr$(x)+chr$(0)+chr$(1)
20070 get#1,a$:ifa$=chr$(255)then20110
20080 print#2,"p"+chr$(2)+chr$(x)+chr$(0)+chr$(1)
20090 input#1,a$:printa$
20100 print#2,"s:"+a$
20110 nextx
20120 print#2,"s:inhalt"+str$(n)
20125 close1:close2
20130 return
60000 rem
60010 rem programmende
60020 rem
60030 print"{clr}Programmende:":print:print
```

```
60040 print"Sind Sie sicher?"
60050 geta$:ifa$<>"j"anda$<>"n"then60050
60060 ifa$="n"thenreturn
60070 print:print"Auf Wiedersehen !!!"
60080 sys64738

READY.
```

```
0 rem anzeige eines directory mit
1 rem einer unterroutine
2 rem ohne basic-programmverlust
3 rem
4 rem (w) 1985 by karsten schramm
5 rem
1000 open1,8,0,"$":get#1,a$:get#1,a$
1010 get#1,d$:get#1,a$:printval(d$);
1020 forx=0to27:get#1,a$:printa$;:next:print
1030 ifst=64then1080
1040 get#1,a$:get#1,a$
1050 get#1,zl$:get#1,zh$:printasc(zl$+chr$(0))+256*asc(zh$+chr$(0));
1060 forx=0to27:get#1,a$:printa$;:next:print
1070 goto1030
1080 close1

READY.
```

```
10 rem ************************
20 rem *                      *
30 rem *  disk-format-system  *
40 rem *                      *
50 rem *   (w) 1985 by koss   *
60 rem *                      *
70 rem ************************
80 data5657,5638,6947,7770,8264,7062,8578,6111,3989,3215,9192,10797
90 data8104,8232,8308,3524,3180,5204,4577
100 data0,14,8,10,0,158,32,50,48,54,52,32,32,0,0,0,162,64,160,8,134,2,132,3
110 data162,0,160,192,134,4,132,5,160,0,162,5,177,2,145,4,200,208,249,230,3
120 data230,5,202,208,242,120,169,242,141,50,3,169,195,141,51,3,88,96,234,234
130 data165,10,201,36,144,7,169,18,133,67,76,19,5,32,75,242,133,67,169,0,133
140 data27,160,0,162,0,165,57,153,0,3,200,200,165,27,153,0,3,200,165,10,153
150 data0,3,200,165,19,153,0,3,200,165,18,153,0,3,200,169,15,153,0,3,200,153
160 data0,3,200,169,0,89,250,2,89,251,2,89,252,2,89,253,2,153,249,2,230,27
170 data165,27,197,67,144,190,169,3,133,49,152,72,138,157,0,7,232,208,250,32
180 data48,254,104,168,136,32,229,253,32,245,253,169,7,133,49,32,233,245,133
190 data58,32,143,247,169,0,133,50,32,14,254,169,255,141,1,28,162,5,80,254
200 data184,202,208,250,162,10,164,50,80,254,184,185,0,3,141,1,28,200,202,208
210 data243,162,9,80,254,184,169,85,141,1,28,202,208,245,169,255,162,5,80,254
220 data184,141,1,28,202,208,247,162,187,80,254,184,189,0,1,141,1,28,232,208
230 data244,160,0,80,254,184,177,48,141,1,28,200,208,245,169,85,162,8,80,254
240 data184,141,1,28,202,208,247,165,50,24,105,10,133,50,198,27,208,149,80
250 data254,184,80,254,184,32,0,254,169,200,133,31,169,0,133,48,169,3,133,49
```

```
260 data165,67,133,27,32,86,245,162,10,160,0,80,254,184,173,1,28,209,48,208
270 data14,200,202,208,242,24,165,48,105,10,133,48,76,53,6,198,31,208,209,169
280 data6,76,211,253,32,86,245,160,187,80,254,184,173,1,28,217,0,1,208,231
290 data200,208,242,162,252,80,254,184,173,1,28,217,0,7,208,215,200,202,208
300 data241,198,27,208,176,76,158,253,160,0,185,224,6,153,0,2,200,204,223,6
310 data144,244,173,223,6,141,116,2,173,222,6,141,123,2,169,0,133,127,32,0
320 data193,172,123,2,185,0,2,133,18,185,1,2,133,19,32,7,211,169,26,141,5,28
330 data169,192,133,0,165,0,48,252,174,220,6,134,10,169,224,133,2,165,2,48
340 data252,201,2,176,12,232,236,221,6,144,236,32,64,238,96,234,234,162,2,76
350 data10,230,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
360 data0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
370 data162,0,32,135,194,160,0,32,207,255,201,13,240,8,153,224,193,200,192
380 data16,144,241,169,44,153,224,193,200,140,222,193,162,71,32,135,194,162
390 data0,32,207,255,201,13,240,9,153,224,193,200,232,224,2,144,240,140,223
400 data193,162,83,32,135,194,32,207,255,133,250,32,207,255,133,251,169,0,133
410 data208,162,98,32,135,194,32,207,255,133,252,32,207,255,133,253,169,0,133
420 data208,165,250,166,251,32,4,196,141,220,193,165,252,166,253,32,4,196,141
430 data221,193,238,221,193,234,234,234,234,234,234,234,234,234,234,234,234
440 data234,234,76,147,194,189,77,195,240,6,32,210,255,232,208,245,96,169,13
450 data32,210,255,169,13,32,210,255,169,0,162,192,133,167,134,168,169,0,162
460 data5,133,169,134,170,169,8,32,177,255,169,111,32,147,255,169,77,32,168
470 data255,169,45,32,168,255,169,87,32,168,255,160,0,165,169,32,168,255,165
```

```
480 data170,32,168,255,169,30,32,168,255,177,167,32,168,255,200,192,30,144
490 data246,32,174,255,24,165,167,105,30,133,167,144,3,230,168,24,165,169,166
500 data170,105,30,133,169,144,2,230,170,224,7,144,173,201,0,144,169,169,8
510 data32,177,255,169,111,32,147,255,169,77,32,168,255,169,45,32,168,255,169
520 data69,32,168,255,169,96,32,168,255,169,6,32,168,255,32,174,255,169,0,133
530 data144,169,8,32,180,255,169,111,32,150,255,32,165,255,32,210,255,36,144
540 data80,246,32,171,255,76,220,195,0,0,0,0,0,147,32,32,32,32,32,32,32,42
550 data42,42,32,68,73,83,75,45,70,79,82,77,65,84,45,83,89,83,84,69,77,32,42
560 data42,42,13,13,13,32,40,67,41,32,49,57,56,53,32,66,89,32,75,79,83,83,32
570 data32,32,13,13,13,68,73,83,75,78,65,77,69,58,32,0,13,13,68,73,83,75,45
580 data73,68,58,32,0,13,13,70,82,79,77,32,84,82,65,67,75,58,36,0,13,13,84
590 data79,32,84,82,65,67,75,58,36,0,13,13,65,78,79,84,72,69,82,32,70,79,82
600 data77,65,84,32,40,89,47,78,41,32,63,32,13,13,0,0,0,0,0,32,41,196,162,111
610 data32,135,194,32,228,255,240,251,201,89,208,3,76,0,194,96,0,165,183,240
620 data3,76,237,245,32,0,194,169,1,162,0,160,0,24,96,133,2,134,3,165,2,201
630 data65,144,3,24,105,9,41,15,10,10,10,10,133,2,165,3,201,65,144,3,24,105
640 data9,41,15,5,2,133,2,96,169,242,141,50,3,169,195,141,51,3,96,0
1000 rem
1010 rem **** datas initialisieren
1020 rem
1030 restore:print:print:print"datas werden ueberprueft !!!":print:print
1040 clr:dimp(19):dimw(19)
1050 forx=0to18:readp(x):p=p+p(x):next
1060 ifp<>124349thenprint"pruefsummenfehler":print:print:list 80-90
1070 forx=0to18:fory=0to59:reada:w(x)=w(x)+a:nexty
```

```
1080 ifw(x)<>p(x)then1150
1090 nextx
1100 print:print"die datas sind ok und werden":print:print"abgespeichert!"
1110 restore:forx=0to18:reada:next
1120 forx=0to1139:reada:pokex+2048,a:next
1130 poke45,119:poke174,119:poke46,12:poke175,12:clr
1140 print:print"mit 'save' abspeichern!":print:end
1150 rem fehlerbehandlung
1160 print:print"fehler in den datas"x*60" bis"x*60+59" !":z=int(x*600/17.8)
1170 print:print:print"das entspricht in etwa den              {down}zeilen
ab"z
1180 end

READY.
```

```
0 rem nachtraegliches schliessen eines
1 rem files.
2 rem soll ein file geschlossen werden,
3 rem so ist nach dem filenamen ein 'j'
4 rem einzugeben.
5 rem soll beendet werden, so geben sie
6 rem 'e' ein.
7 rem soll zum naechsten filenamen ge-
8 rem gangen werden, so tippen sie 'n'.
9 rem
1000 mm=0
1010 mm=mm+1:dd$="":gosub1120
1020 ifdd$=nn$thenend
1030 printmid$(dd$,4,16):inputaa$
1040 ifaa$="e"thenend
1050 ifaa$="n"then1010
1060 hh$=left$(dd$,1)
1070 hh$=chr$(asc(hh$)or2^7)
1080 dd$=hh$+right$(dd$,29)
1090 gosub1330
1100 goto1010
1110 end
1120 rem
1130 rem holen des eintrags
1140 rem
1150 open15,8,15:open8,8,8,"#"
1160 nn$="":fori=1to30:nn$=nn$+chr$(0):nexti
1170 xx=int((mm-1)/8)
1180 print#15,"u1 8 0 18 0"
1190 forzz=1toxx+1
1200 print#15,"b-p 8 0"
1210 get#8,tt$:tt=asc(tt$+chr$(0))
1220 get#8,ss$:ss=asc(ss$+chr$(0))
1230 iftt=0thendd$=nn$:goto1310
1240 print#15,"u1 8 0";tt;ss
1250 nextzz
1260 pp=mm-(xx*8):pp=(pp-1)*32+2
1270 print#15,"b-p 8";pp
1280 forzz=1to30:get#8,zz$
1290 ifzz$=""thenzz$=chr$(0)
1300 dd$=dd$+zz$:nextzz
1310 close8:close15
1320 return
```

```
1330 rem
1340 rem zurueckschreiben des eintrags
1350 rem
1360 open15,8,15:open8,8,8,"#"
1370 xx=int((mm-1)/8)
1380 print#15,"u1 8 0 18 0"
1390 forzz=1toxx+1
1400 print#15,"b-p 8 0"
1410 get#8,t$:tt=asc(t$+chr$(0))
1420 get#8,s$:ss=asc(s$+chr$(0))
1430 iftt=0then1500
1440 print#15,"u1 8 0";tt;ss
1450 nextzz
1460 pp=mm-(xx*8):pp=(pp-1)*32+2
1470 print#15,"b-p 8";pp
1480 print#8,dd$;
1490 print#15,"u2 8 0";tt;ss
1500 close8:close15
1510 return

READY.
```

```
10 remeddi - diskmonitor/editor
20 rem         von
30 remkarsten schramm (w) 1984
40 rem
50 print"{clr}{blk}":poke53280,14:poke53281,14
60 gosub1340
70 open1,8,15,"i0":open2,8,2,"#"
80 print"{clr} e d d i  -  hauptmenue"
90 he$="byte      dec   hex   bin        asc":poke650,128
100 print" EEEEEEEEEEEEEEEEEEEEE":print
110 print" '`' - disk-status":print
120 print"(f1) - scrolling vorwaerts":print
130 print"(f2) - scrolling rueckwaerts":print
140 print"(f3) - block lesen":print
150 print"(f4) - block schreiben":print
160 print"(f5) - editor einschalten":print
170 print"(f6) - diskette wechseln":print
180 print"(f7) - rueckkehr ins menue":print
190 print"(f8) - programmende"
200 po=1:goto 1170
210 rem eddi an
220 x=0:y=0
230 fory=eto255step16
240 po=2:print"{clr}editor-modus fuer track"t" sektor"s
250 print:printhe$:print
260 forx=ytoy+15:printx:nextx
270 print"{home}{down}{down}{down}":forx=ytoy+15
280 da=peek(50000+x):gosub1010:printx,ou$
290 input"{up}{rght}{rght}{rght}{rght}{rght}{rght}{rght}{rght}";in$
300
ifleft$(in$,1)="~"thenprint"{home}{down}{down}{down}{down}{down}{down}{down}{d
own}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}":goto11
70
310
ifleft$(in$,1)="□"thenprint"{home}{down}{down}{down}{down}{down}{down}{down}{d
own}{down}{down}{down}{down}{down}{down}{down}{down}{down}{down}":
goto350
320 da=val(left$(in$,3)):ifda>255orda<0thenprint"{up}{up}":goto280
330 poke50000+x,da
340 nextx:print
350 print"eingabe ?";
360 geta$:ifa$=""then360
370 ifa$="{F1}"then410
380 ifa$="{F2}"then470
390 ifa$<>" "thennexty
400 po=1:goto1170
```

```
410 print"{home}{down}{down}{down}":printe".....???"
420 geta$:ifa$=""then420
430 ifa$="{F2}"then470
440 ifa$<>"{F1}"then230
450 e=e+16:ife>255thene=0
460 goto410
470 print"{home}{down}{down}{down}":printe".....???"
480 geta$:ifa$=""then480
490 ifa$="{F1}"then410
500 ifa$<>"{F2}"then230
510 e=e-16:ife<0thene=240
520 goto470
530 rem diskettenwechsel
540 print"{clr}bitte neue diskette einlegen"
550 geta$:ifa$=""then550
560 run
570 rem block read
580 po=2:print"{clr}   block lesen":print:print
590 input"track, sektor ";t,s
600 ift<1ort>35then580
610 print#1,"u1 2 0"t;s
620 ifst<>0thenprint:goto1170
630 print#1,"b-p 2 0"
640 sys49152:e=0:x=0:y=0:goto770
650 fory=eto255step16
660 print"{clr}track"t" sektor"s
670 print:printhe$:print
680 forx=ytoy+15:da=peek(50000+x):gosub1010:printx,ou$:nextx
690 goto1170
700 rem block write
710 po=1:print:print:input"{clr}{red}track, sektor";t,s:print"{blk}"
720 print#1,"b-p 2 0"
730 sys49177
740 print#1,"u2 2 0"t;s
750 goto1170
760 rem scroll forward
770 e=x:ife>255thenx=0:e=0
780 print"{clr}track"t" sektor"s
790 print:printhe$:print
800 da=peek(50000+e):gosub1010:printe,ou$
810 x=x+16
820 geta$:ifa$=""then820
830 ifa$="{F1}"then770
840 ifa$="{F2}"thenx=x-16:goto880
```

```
850 ifa$="{F5}"then210
860 goto650
870 rem scroll backward
880 e=x:ife<0thene=240:x=240
890 print"{clr}track"t" sektor"s
900 print:printhe$:print
910 da=peek(50000+e):gosub1010:printe,ou$
920 x=x-16
930 geta$:ifa$=""then930
940 ifa$="{F2}"then880
950 ifa$="{F1}"thenx=x+16:goto770
960 ifa$="{F5}"then210
970 goto650
980 rem bereitstellung des strings
990 rem da/da$ sind ausgabewerte                h$,d$,b$,c$ sind zwischenwerte
1000 rem ou,ou$ sind endergebnisse
1010 ifda>31andda<128orda>159andda<256thenc$=chr$(da):goto1030
1020 c$="."
1030 xx$="000":d$=right$(str$(da),len(str$(da))-1)
1040 d$=left$(xx$,3-len(d$))+d$
1050 xx$="123456789abcdef":h$=""
1060 hh=int(da/16):hl=da-hh*16
1070 ifhhthenh$=h$+mid$(xx$,hh,1):goto1090
1080 h$=h$+"0"
1090 ifhlthenh$=h$+mid$(xx$,hl,1):goto1110
1100 h$=h$+"0"
1110 b$="":forq=7to0step-1
1120 if(daand(2^q))<>0thenb$=b$+"1":next:goto1140
1130 b$=b$+"0":next
1140 ou$=d$+"   "+h$+"    "+b$+"   "+c$
1150 return
1160 end
1170 rem get kommando
1180 print:print"kommando ?  ";
1190 print"{left}{CBM-B}";:forw=1to75:getko$:ifko$<>""then1240
1200 nextw
1210 print"{left}{rvon}{CBM-B}{rvof}";:forw=1to75:getko$:ifko$<>""then1240
1220 nextw
1230 goto1190
1240 ifko$="`"then1300
1250 if asc(ko$)>140orasc(ko$)<133then1190
1260 ko=asc(ko$)-132
1270 on po goto1280,1290,1400
```

```
1280 on ko goto1190,570,210,80,1190,700,530,1400
1290 on ko goto760,570,210,80,870,700,530,1400
1300 print
1310 get#1,a$:printa$;:ifst<>64then1310
1320 goto 1170
1330 end
1340 data160,0,169,8,32,9,237,169,98,32,199,237,32,19,238,153,80,195,200
1350 data208,247,32,239,237,96,160,0,169,8,32,12,237,169,98,32,185,237
1360 data185,80,195,32,221,237,200,208,247,32,254,237,96,0,0
1370 restore:forz=1to51:reada:poke49151+z,a:next
1380 rem get:49152; write:49177
1390 return
1400 print:print:print"{lblu}auf wiedersehen
!!!":print:poke53280,14:poke53281,6
1410 end

READY.
```

```
0 rem dieses programm testet die
1 rem schreibschutzlichtschranke und
2 rem zeigt deren zustand mit der led
3 rem des laufwerks an.
4 rem
5 rem floppy muss nach dem test
6 rem ausgeschaltet werden, da der test
7 rem aus einer endlosschleife besteht!
8 rem
9 rem (w) 1985 by karsten schramm
10 goto80
20 , 0300 ad 00 1c lda $1c00
30 , 0303 29 10    and #$10
40 , 0305 4a       lsr
50 , 0306 8d 00 1c sta $1c00
60 , 0309 4c 00 03 jmp $0300
70 :
80 open1,8,15
90 forx=0to11:reada
100 print#1,"m-w"chr$(x)chr$(3)chr$(1)chr$(a):next
110 print#1,"m-e"chr$(0)chr$(3)
120 data173,0,28,41,16,74,141,0,28,76,0,3

READY.
```

```
0 rem   programm zum erzeugen eines
1 rem         22, read error
2 rem      in beliebigem sektor
3 rem
4 rem  von karsten schramm 09.01.1985
5 rem
6 rem
7 rem  programm wird auch in speicher
8 rem       ab $8000 abgelegt.
9 rem
10 poke56,31:poke52,31:clr:open1,8,15,"i"
20 forx=0to80:reada:poke32768+x,a:next
30 input"track fuer error 22";t
40 input"sektor fuer error 22";s
50 poke32777,t:poke32834,t:poke32781,s
60 restore
70 forx=0to80:reada:print#1,"m-w"chr$(x)chr$(5)chr$(1)chr$(a):nextx
80 print:print:print"programm startet"
90 print#1,"m-e"chr$(64)chr$(5):close1:end
100 data 165,18,133,22,165,19,133,23,169,35,133,24,169,1,133,25,32,39
110 data 245,32,86,245,173,12,28,41,31,9,192,141,12,28,169,255,141,3,28
120 data 169,85,141,1,28,80,254,184,80
130 data 254,184,80,254,184,32,0,254,76
140 data 158,253,234,234,234,234,234,234
150 data 234,234,169,35,133,10,169,224,133,2,165,2,48,252,96,0,0,0

READY.
```

```
0 rem   programm zum erzeugen eines
1 rem         23, read error
2 rem      in beliebigem sektor
3 rem
4 rem  von karsten schramm 09.01.1985
5 rem
6 rem
7 rem  programm wird auch in speicher
8 rem       ab $8000 abgelegt.
9 rem
10 poke56,31:poke52,31:clr:open1,8,15,"i"
20 forx=0to80:reada:poke32768+x,a:next
30 input"track fuer error 23";t
40 input"sektor fuer error 23";s
50 poke32777,t:poke32834,t:poke32781,s
60 restore
70 forx=0to80:reada:print#1,"m-w"chr$(x)chr$(5)chr$(1)chr$(a):nextx
80 print:print:print"programm startet"
90 print#1,"m-e"chr$(64)chr$(5):close1:end
100 data 165,18,133,22,165,19,133,23,169,35,133,24,169,0,133,25,32,39
110 data 245,32,86,245,162,0,202,208,253
120 data 173,12,28,41,31,9,192,141,12,28,169,255,141,3,28,169,85,141,1
130 data 28,80,254,184,80,254,184,80,254,184,32,0,254,76,158,253,234,234
140 data 234,169,35,133,10,169,224,133,2,165,2,48,252,96,0,0,0

READY.
```

```
0 rem block-scanner
1 rem dieses programm tastet die
2 rem reihenfolge von bloecken eines
3 rem bestimmten tracks ab und zeigt
4 rem diese an.
5 rem programm steht ab $9200.
6 rem
7 rem          scanner
8 rem (w) 1985 by karsten schramm
9 rem
10 print"{clr}{blk}{down}       b l o c k  s c a n n e r "
20 print"      EEEEEEEEEEEEEEEEEEEEEEEEE"
30 restore:forx=0to125:reada:poke37376+x,a:next
40 poke56,128:poke52,128:clr
50 poke53280,14:poke53281,14
60 print:print:input"track";t
70 open1,8,15:open2,8,2,"#":print#1,"u1 2 0";t;0:poke37376+106,t
80 ift<36thensa=16
90 ift<31thensa=17
100 ift<25thensa=18
110 ift<18thensa=20
120 print:print"installing program":print
130 forx=0to125:a=peek(37376+x):print#1,"m-w"chr$(x)chr$(3)chr$(1)chr$(a):next
140 input"want to scan ";a$:ifa$<>"y"thenend
150 print#1,"m-e"chr$(105)chr$(3)
160 print#1,"m-r"chr$(0)chr$(4)chr$(21)
170 forx=20to0step-1:get#1,a$:poke32768+x,asc(a$+chr$(0)):next
180 print"{clr}":print
190 print"an stelle 0    steht sektor"peek(32768+sa-1)
200 print"an stelle 1    steht sektor"peek(32768+sa)
210 forx=0tosa-2:a=peek(32768+x)
220 print"an stelle"x+2 tab(14)" steht sektor"a
230 nextx
240 geta$:ifa$=""then240
250 close2:close1:run
260 data165,0,41,2,208,3,76,158,253,169,82,133,36,162,0,32,86,245,80,254,184
270 data173,1,28,197,36,208,243,80,254,184,173,1,28,149,37,232,224,7,208,243
280 data32,151,244,165,25,201,0,208,215,169,21,133,12,162,0,169,82,133,36,32
```

```
290 data86,245,80,254,184,173,1,28,197,36,208,243,80,254,184,173,1,28,149,37
300 data232,224,7,208,243,32,151,244,165,25,166,12,157,0,4,198,12,16,210,76
310 data158,253,234,234,169,1,133,6,133,34,169,226,133,0,165,0,48,252,76,34
320 data235,0,0,0,0

READY.
```

```
0 rem programm zum spezialformatieren
1 rem eines tracks.
2 rem erlaubt abaenderung saemtlicher
3 rem header-parameter und vertauschung
4 rem der reihenfolge der sektoren
5 rem auf diskette
6 rem
7 rem       format one track
8 rem (w) 1985 by karsten schramm
9 rem
10 poke56,128:poke52,128:clr
20 restore:forx=0to127:reada:poke37120+x,a:next
30 poke53280,14:poke53281,14
40 print:print"{clr}{blk}     f o r m a t  o n e  t r a c k"
50 print"     EEEEEEEEEEEEEEEEEEEEEEEEEEEE"
60 print:print:print
70 input"track ";t:print
80 ift<1ort>35thenrun
90 ift<36thensa=17:k=11
100 ift<31thensa=18:k=10
110 ift<25thensa=19:k=9
120 ift<18thensa=21:k=9
130 open1,8,15,"i"
140 print#1,"m-r"chr$(18)chr$(0)chr$(2)
150 get#1,i1$,i2$
160 print"id1 ist jetzt : "asc(i1$+chr$(0)),i1$
170 print"id2 ist jetzt : "asc(i2$+chr$(0)),i2$
180 print:print:input"neue id fuer gewaehlten track (i1,i2) ";i1,i2
190 poke252,t:poke253,0:poke254,i2:poke255,i1:poke2,sa
200 poke37238,k:poke37240,sa:poke37221,sa:poke37217,t
210 sys(9*4096+256)
220 print:input"sollen sektoren vertauscht werden ";a$
230 ifa$<>"j"then330
240 forc=32768to33023:pokec,0:next:print"{clr}"
250 print"{home}track"t" :":print
260 forx=0tosa-1
270 print"stelle"x;:input"sektor ";y
280 forz=0to7:poke32768+x*8+z,peek(36864+y*8+z):nextz
290 nextx
300 input"{down}alles richtig ";a$
310 ifa$<>"j"then250
```

```
320 forc=0to255:poke36864+c,peek(32768+c):next
330 forx=0to50:a=peek(37200+x)
340 print#1,"m-w"chr$(x)chr$(6)chr$(1)chr$(a)
350 nextx
360 print:print"soll formatiert werden?"
365 input"'n' fuer headermanipulation";a$
370 ifa$<>"j"then530
380 print"{clr} formatierung startet bei 0 !!!"
390 forc=0to192:print"{home}{down}{down}
{left}{left}{left}{left}{left}"192-c
400 print#1,"m-w"chr$(c)chr$(3)chr$(1)chr$(peek(36864+c)):next
410 print:print:print
420 print#1,"m-e"chr$(16)chr$(6)
430 get#1,a$:printa$;:ifst<>64then430
440 print"{clr}{down}testen von track"t
450 print:open2,8,2,"#"
460 forx=0tosa-1
470 print#1,"u1 2 0";t;x:printx,
480 get#1,a$:printa$;:ifst<>64then480
490 nextx
500 close2:close1
510 geta$:ifa$=""then510
520 run
530 print"{clr} headermanipulation":print:print
540 print"welcher header (0 -"(sa-1)")";:inputa
550 ifa<0ora>=sathen530
560 print:print:x=36864+a*8
570 print"headercode
"peek(x);:input"{left}{left}{left}{left}{left}{left}";hc:pokex,hc
580 print"id 1
"peek(x+5);:input"{left}{left}{left}{left}{left}{left}";i1:pokex+5,i1
590 print"id 2
"peek(x+4);:input"{left}{left}{left}{left}{left}{left}";i2:pokex+4,i2
600 print"track
"peek(x+3);:input"{left}{left}{left}{left}{left}{left}";tr:pokex+3,tr
610 print"sektor
"peek(x+2);:input"{left}{left}{left}{left}{left}{left}";sk:pokex+2,sk
620 print"paritaet
"peek(x+1);:input"{left}{left}{left}{left}{left}{left}";pr:pokex+1,pr
630 print"luecke 1
"peek(x+6);:input"{left}{left}{left}{left}{left}{left}";fr:pokex+6,fr
640 print"luecke 2
"peek(x+7);:input"{left}{left}{left}{left}{left}{left}";fr:pokex+7,fr
650 print:input"noch ein header";a$:ifa$="j"then530
660 goto220
670 data169,0,133,253,160,0,169,8,153,0,144,200,200,165,253,153,0,144,200,165
680 data252,153,0,144,200,165,254,153,0,144,200,165,255,153,0,144,200,169,15
690 data153,0,144,200,153,0,144,200,169,0,89,250,143,89,251,143,89,252,143
700 data89,253,143,153,249,143,230,253,165,253,197,2,144,190,96,0,0,0,0,0,0
```

```
710 data0,160,176,162,0,169,35,133,81,76,132,252,0,0,0,0,0,169,1,133,12,169
720 data21,133,67,32,7,211,32,0,193,76,218,200,0,0,0,0,0,9,0,21,0,0,0,0,0,0
730 data0

READY.
```

Stichwortverzeichnis

&-File ..... 70ff

A

ADH ..... 64ff
ADL ..... 64ff
ASC ..... 60
ATN IN ..... 80
ATN OUT ..... 80
ATN-Leitung ..... 138, 159
ATN-Signal ..... 85, 87
Abfrage der Lichtschranke unterbinden ..... 152f
Adresse ..... 14, 64ff, 77
Adressbus ..... 137
Anfügen von Daten ..... 25
Anlegen einer Datei ..... 58
Anschlag des Tonkopfes ..... 150, 153
Anzahl der Bytes, im Programm ..... 70
Anzahl, Sektoren pro Track ..... 37
Anzahl, der Blöcke eines Files ..... 47
Anzahl, der Bytes eines Blocks ..... 38f, 51
Anzahl, der Tracks ..... 162
Append ..... 26, 27
Arbeiten mit ungültigen Spuren ..... 133
Arbeitspuffer ..... 78
Arbeitsweise des 6502 Mikroprozessors ..... 83
Arten der Datenspeicherung ..... 36
Aufbau, der BAM ..... 39
Aufbau, der Datenblöcke relativer Files ..... 50
Aufbau, der Side-Sektor-Blöcke ..... 50f
Aufbau, des Directory ..... 43
Aufbau, einer Spur ..... 37
Aufbau, einer neu formatierten Diskette ..... 35
Aufbau, eines S-Fi-les ..... 70ff
Aufbau, eines Blockheaders ..... 102
Aufbau, eines Datenblocks ..... 103
Aufbau, eines Fileeintrags ..... 44ff
Aufbau, eines Sektors ..... 38, 101
Aurbau, von DEL-Files ..... 52

Aufbau, von PRG-Files ..... 48
Aufbau, von REL-Files ..... 49
Aufbau, von SEQ-Files ..... 48
Aufbau, von USR-Files ..... 49
Aufgaben des DOS ..... 82
Aufschrauben der Floppy ..... 92
Aufzeichnung, auf Diskette ..... 114
Aufzeichnung, von Daten ..... 101
Aufzeichnungskapazitäten ..... 41
Ausfuhren von Programmen ..... 91
Ausführung von Befehlen ..... 18, 82
Ausgangszustand ..... 16
Auswertung von Befehlen ..... 82, 83
Autostart-Routine ..... 158
Änderung der Gerätenummer ..... 65, 150

B

BAM ..... 39ff, 62f
BC ..... 79, 85, 87
BUMP ..... 96
BYTE-READY-Leitung ..... 118
Befehl, # ..... 58f
Befehl, & ..... 69ff
Befehl, BLOCK-ALLOCATE ..... 62, 157
Befehl, BLOCK-EXECUTE ..... 67
Befehl, BLOCK-FREE ..... 63
Befehl, BLOCK-READ ..... 59, 157
Befehl BLOCK-WRITE ..... 61, 157
Befehl BUFFER-POINTER ..... 61
Befehl INITIALIZE ..... 97
Befehl MEMORY-EXECÜTE ..... 66
Befehl MEMORY-READ ..... 64
Befehl MEMORY-WRITE ..... 65
Befehl NEW ..... 17
Befehl POSITION ..... 30, 32
Befehl REPLACE ..... 45, 47, 158
Befehl SCRATCH ..... 121
Befehl Ul ..... 59
Befehl U2 ..... 61
Befehl USER ..... 68
Befehl VALIDATE ..... 26, 122
Befehle, für Direktzugriff ..... 57ff

Befehlssatz ..... 57, 91, 381
Befehlsstrings ..... 78
Behandlung der Floppy ..... 153
Belegung, der Bytes im Directory ..... 44
Belegung, der Pufferspeicher ..... 78
Belegung, einer Diskette ..... 36
Benutzerpuffer ..... 78
Berechnung, Gesamtanzahl der Datensätze ..... 52
Berechnung, der Prüfsumme bei &-Files ..... 71
Betriebsart eines Files ..... 25-27
Betriebssystem ..... 13
Bewegung des Tonkopfes ..... 96
Binär-GCR-Konvertierung ..... 108ff
Binärcode ..... 108ff
Bitkombinationen ..... 110
Bitmuster der BAM ..... 40
Blinken der Leuchtdiode ..... 18, 31
Block ..... 19, 39, 58ff
Block Availability Map ..... 39, 41
Block, lesen ..... 59
Block, schreiben ..... 61
Block-Befehle ..... 59ff
Blockbelegungsplan ..... 39
Blockheader ..... 101, 126
Blockheaderkennzeichen ..... 102
Bus, parallel ..... 137
Bus, seriell ..... 14, 137
Busbedienung ..... 82
Busbetrieb ..... 85
Buscontroller ..... 79
Buspriorität ..... 15
Busroutinen ..... 144
Busroutinen, von HYPRA-LOAD ..... 144ff
Byte, höherwertiges ..... 31, 64ff, 77
Byte, niederwertiges ..... 31, 64ff, 77

C

CIA 6526 ..... 80
CLK-Leitung ..... 138
CLOCK IN ..... 80
CLOCK OUT ..... 80
CLOSE ..... 16, 17

CMD ..... 29
Checksumme ..... 102, 103
Continuos Mode ..... 86
Controller ..... 138

D

DATA IN ..... 80
DATA OUT ..... 80
DATA-Leitung ..... 138
DC ..... 81, 105, 107
DEL-File ..... 52
DOS ..... 13, 14, 38, 47, 52, 82
DOS 5.1 ..... 19
DOS-Version ..... 43
Datei, gelöschte ..... 52
Datei, relative ..... 31, 52
Datei, sequentielle ..... 26
Daten ..... 15, 38
Daten, hinzufügen ..... 27
Datenblock ..... 38, 101
Datenblockkennzeichen ..... 103
Datenbus ..... 137
Dateneinheit ..... 18
Datenkanal ..... 15
Datensatz ..... 30, 50
Datensatzlänge ..... 30, 47, 51
Datenspeicherung ..... 23, 25, 28, 29
Datenspeicherung, hardsektoriert ..... 36, 105
Datenspeicherung, relativ ..... 29
Datenspeicherung, sequentiell ..... 25, 28, 48
Datenspeicherung, softsektoriert ..... 36, 105
Datensatze, Anzahl ..... 31
Datenverwaltung ..... 23
Dialog ..... 14, 17, 79
Directory ..... 23, 38, 92
Directorytrack ..... 162
Direktzugriff ..... 57ff
Disk Operating System ..... 13
Diskcontroller ..... 81, 83, 93
Diskcontroller-Modus ..... 85
Diskette ..... 14
Disketten doppelseitig verwenden ..... 149

Diskettenformat ..... 35
Diskettenhülle ..... 128
Diskettenidentifikation ..... 40, 43
Diskettenkapazität ..... 52
Diskettenname ..... 40
Diskettenstation ..... 13
Drehzahlschwankungen ..... 117
Drivemotor ..... 83, 94

E

EOI ..... 19, 139
Eingriffe in die Platine ..... 150
Einschalten von Floppy und Computer ..... 15, 84
Einsprungadressen, für USER-Befehle ..... 69
Eintrag im Directory ..... 44, 47
Elementarmagnete ..... 111, 112
Empfangen ..... 16, 83, 118
Endekennzeichen ..... 48f, 123

F

Fehler, beim Laden ..... 19
Fehler, im DOS 2.6 der 1541 ..... 157
Fehler, im Datenblock ..... 126
Fehler, im Header eines Datenblocks ..... 98, 126
Fehler, in Prüfsumme ..... 19
Fehlerkanal ..... 19
Fehlermeldungen ..... 18, 71f, 78, 97, 125, 389
Fehlerroutine ..... 66
File, offen ..... 46
Filebetriebsart ..... 27
Fileeintrag ..... 44, 47, 51
Filekopierprogramm ..... 28
Filename ..... 23, 25, 45, 58
Filenummer ..... 16, 58
Filetyp ..... 23, 26, 45ff
Filetyp, DEL ..... 46
Filetyp, PRG ..... 24, 46
Filetyp, REL ..... 46
Filetyp, SEQ ..... 46
Filetyp, USR ..... 46

Flag ..... 87
Floppy ..... 13
Floppystation ..... 13, 41
Formatieren einer Diskette ..... 17, 35f, 43, 105
Formatierungsroutine ..... 41
Formatkennzeichen ..... 40
Freigeben eines Datensatzes ..... 30, 32
Frequenz der IRQs ..... 96
Funktionsweise eines Schreib/Lesekopfes ..... 111

G

GCR-Code ..... 107 ff
GCR-Codierung ..... 107ff
GET# ..... 16, 28, 32, 60
Gerätenummer ..... 14, 48, 64f, 139
Geschwindigkeit ..... 69, 91, 143
Group Code Recording ..... 107

H

Hard-Errors 125
Hardware ..... 82, 149
Hardware-Stack ..... 77
Hauptarbeitspuffer ..... 78
HaupCprogramm ..... 84ff, 92
Header, des Directory ..... 40
Header, eines Datenblocks ..... 97

I

ID ..... 43, 97, 102f
IEEE-488-Bus ..... 143
INPUT# ..... 16, 28
IRQ ..... 83
Indexloch ..... 105
Inhalt, einer Diskette ..... 23
Inhalt, eines Blocks ..... 39
Inhalt, eines Blocks im Directory ..... 44
Inhalt, eines Side-Sektor-Blocks ..... 50f
Inhalte von Speicherstellen ..... 65

Inhaltsverzeichnis ..... 23, 26, 39
Initialisierung ..... 97
Interrupt ..... 83ff
Interrupt Disable Bit ..... 84
Interruptprogramm ..... 84ff, 93
Intervalltimer ..... 86

J

Job ..... 87, 94
Jobcode ..... 94
Jobschleife ..... 86f, 96
Jobspeicher ..... 93
Justage des Tonkopfes ..... 94, 162

K

Kabel ..... 137
Kanalnummer ..... 16, 59 ff
Kennzeichnung des Filetyps ..... 45f
Kommando ..... 85
Kommandokanal ..... 15, 30, 64, 97
Kommunikation zwischen Hauptprogramm und DG ..... 93
Kompatibilität 1541-4040 ..... 161f
Kopfe, Anzahl pro Laufwerk ..... 162

L

LED am Laufwerk ..... 81
LIST ..... 24
LISTEN ..... 139, 141
LOAD ..... 15, 24, 27, 48
Laden ..... 48
Laufwerk ..... 14, 81, 149, 150, 162
Laufwerk, intelligentes ..... 14, 18
Laufwerksmechanik ..... 92
Laufwerksmotor ..... 81, 95
Leerinhalt von Blöcken ..... 158
Leerkennzeichen, CHR$(255) ..... 32
Lesen der Daten ..... 83, 115
Lesen eines Blocks ..... 59, 96

Lesen von fehlerhaften Files ..... 124
Lesezugriff ..... 32
Leuchtdiode ..... 27
Lichtschranke ..... 81, 150
Linkbytes ..... 104
Linker ..... 38, 40, 48, 60
List-File ..... 140
Länge, des Diskettennamens ..... 42
Länge, eines Files ..... 23, 45
Löschen eines Files ..... 46
Lücke ..... 103, 104
Lücke nach Datenblockheader ..... 161

M

Magnetisches Feld ..... 112
Magnetisierungswechsel ..... 114
Magnetismus ..... 111
Markensetzen ..... 36
Markieren eines Datensatzes ..... 30
Markierungen ..... 36
Maschinenprogramm, im Puffer starten ..... 66ff, 94
Meldungen ..... 389
Memory-Befehle ..... 59ff
Mikroprozessor ..... 75, 77, 83
Modus, zum Öffnen eines Files ..... 26

N

NMI ..... 83
NMI-Vektor ..... 68
Namen, der Diskette ..... 42
Namen, der Files ..... 23
NEW, kurz ..... 122ff
NEW, lang ..... 122ff
Nibble ..... 108ff
Normalmodus ..... 94
Nummer, des Direktzugriffskanals ..... 58ff
Nummer, des gewählten Puffers ..... 59
Nummer, eines Datensatzes ..... 50
Nummer, eines Side-SekCor-Blocks ..... 51
Nutzung, der Jobschleife ..... 93

Nutzung, des Hauptprogramms ..... 91

O

OPEN ..... 16
Overflow-Flag ..... 118
Öffnen, eines Direktzugriffskanals ..... 57f
Öffnen, eines Files ..... 16

P

PEEK ..... 64
POKE ..... 65
PRG-File ..... 24, 27, 48
PRINT ..... 28
PRINT# ..... 16, 28, 48, 62
Page 1 ..... 77
Peripheriegeräte ..... 14, 16
Platz, auf der Diskette ..... 23
Positionsbestimmung der Magnet Scheibe ..... 36
Priorität am Bus ..... 15, 138
Programm starten ..... 66, 94
Prozessor ..... 83, 162
Prüf summe, über Programmteil ..... 70f
Prüfsummenfehler ..... 72
Puffer ..... 58
Puffer reservieren ..... 58
Pufferspeicher ..... 32, 58ff, 78, 92

R

RAM ..... 75ff
RAM-Aufteilung ..... 76
RAM-Belegung ..... 165
REL-File ..... 45
RESCRATCH ..... 122
RESET-Signal ..... 84
KESET-Vektor ..... 68, 69
RETURN ..... 28
ROM ..... 13, 75ff
ROM-Bereich ..... 82, 177

Recordlänge ..... 45, 47, 51
Relatives File ..... 32, 47, 49, 52
Retten einer Diskette nach der Formatierung ..... 122ff
Retten von physikalisch zerstörten Disketten ..... 127f
Rückmeldungen ..... 18
Ruckmeldungen, des Diskcontrollers ..... 94

S

SAVE ..... 15, 24, 27
SCRATCH-Schutz ..... 46
SEQ-File ..... 25, 49
ST ..... 18
SYNC-Markierung ..... 102, 104, 106, 131, 161
SYNC-Signal ..... 104, 110
SYS ..... 66
Schließen eines Files ..... 16, 26, 32
Schließen eines Files, nachträglich ..... 26
Schnittstellenbausteine ..... 75, 79
Schreiben ..... 26
Schreib/Leseelektronik ..... 110
Schreib/Lesekopf ..... 83, 92, 96, 113
Schreibdichte ..... 115
Schreiben eines Blocks ..... 61, 97
Schreiben von Bits ..... 111
Schreiben von Daten ..... 62, 107
Schreiben von Fehlern ..... 131
Schreiben von Markierungen ..... 107
Schreibschutz ..... 81
Schreibzugriff ..... 32
Schrittmotor ..... 95
Seite 2 ..... 78
Sektor ..... 37, 47, 48, 101
Sektornummer ..... 38, 58ff
Sekundäradresse ..... 16, 48
Semikolon ..... 28
Senden ..... 16
sequentielles File ..... 26, 32, 48
Side-Sektor ..... 50
Side-Sektor-Block ..... 45, 47, 50
Soft-Errors ..... 125
Softwareschutz ..... 131
Speicher ..... 64ff, 162

Speicherorganisation der 1541 ..... 75ff
Speicherstelle ..... 18, 64, 65, 87
Spooling ..... 28, 49, 140
Spule ..... 112
Spur ..... 35, 48
Spurnummer ..... 38
Startadresse, des Programms im Computer ..... 48
Startadresse, von Programmen ..... 66
Statusbits ..... 19
Statusvariable ..... 18
Steppermotor ..... 81, 95
Steuerung, der Lese/Schreibvorgänge ..... 85
Steuerung, des Laufwerkmotors ..... 85, 95
Steuerung, des Steppermotors ..... 85, 95
Subroutinen des DOS ..... 66
Synchronisation ..... 144ff
Synchronisierung ..... 86, 146
Synchronmarkierungen ..... 105
Systemroutinen ..... 91

T

TALK ..... 139, 141
Time-out ..... 19
Timer ..... 117
Timersteuerung ..... 81
Timing, beim Schreiben und Lesen ..... 117
Tonkopf ..... 95, 107, 111, 112, 153
Track ..... 37, 39, 47, 101
Tracknunmer ..... 37, 58ff
Triggerung des Timers ..... 117
Typ eines Files ..... 23, 24

U

U/min ..... 162
Umleiten der Bildschirmausgabe ..... 29, 49
UNLISTEN ..... 139
UNTALK ..... 139
USR-File ..... 28, 49
Unterprogramme ..... 87, 91
User-Befehle ..... 68

Übertragungsrate ..... 162

V

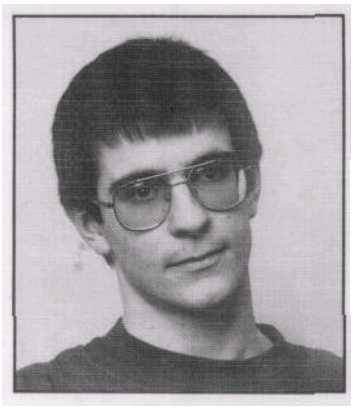
VERIFY ..... 15
VIA 6522 ..... 75, 79
Vergleich der 1541 mit anderen CBM-Floppies ..... 161
Verkettung von Sektoren ..... 38, 48
Verzeichnis der verfügbaren Blöcke ..... 41
Veränderung, der Abstände zwischen Sektoren ..... 133
Veränderung, der Reihenfolge der Sektoren ..... 132
Vorspann, Datenblock ..... 38, 97

W

Wahl eines Puffers ..... 58
Wartung der Floppy ..... 93
Wiederfreigeben eines Files ..... 46
Wiederherstellen gelöschter Files ..... 121
Wiederherstellen zerstörter Disketten ..... 121

Z

Zahlen ..... 16
Zeichenketten ..... 16
Zeiger auf einen Block ..... 41, 44
Zeilenvorschub ..... 28
Zeropage ..... 77, 87f, 95
Zugriff ..... 23, 27
Zugriff, auf einen Datensatz ..... 30
Zurückverfolgen einer Datei ..... 123
Zählweise ..... 37



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 1541

Dieses Buch ist für alle Programmierer konzipiert, die endlich mehr über ihre VC 1541 Floppystation erfahren wollen. Es beginnt bei der grundlegenden Arbeit mit Files und führt Sie im weiteren Verlauf immer tiefer in die Geheimnisse der VC 1541 ein. Es enthält ein komplett dokumentiertes DOS-Listing und ist sowohl für Floppy-Einsteiger als auch für die fortgeschrittenen Maschinensprache-Programmierer geschrieben. Mit diesem Buch lernen Sie die Technik der Aufzeichnung auf Diskette, die Funktionsweise von modernem Softwareschutz sowie von schnellen Kopier- und Ladeprogrammen kennen, damit Sie Ihre VC 1541 effektiv programmieren und manipulieren können.

Das Buch

- beschreibt ausführlich den Vorgang des Formatierens und des Schreibens von Files auf Diskette,
- stellt Fehler im Commodore-Handbuch richtig,
- zeigt die Funktionsweise von schnellen Kopier- und Ladeprogrammen,
- enthält viele fertige Programme,
- erklärt, wie man defekte Disketten trotzdem lesen und beschreiben kann.

Eine Beispieldiskette mit den im Buch enthaltenen Programmlistings ist beim Verlag gesondert erhältlich.

Markt&Technik Verlag Aktiengesellschaft
ISBN 3-89090-0984

DM 49,—
sFr..45,10 / öS 382,20