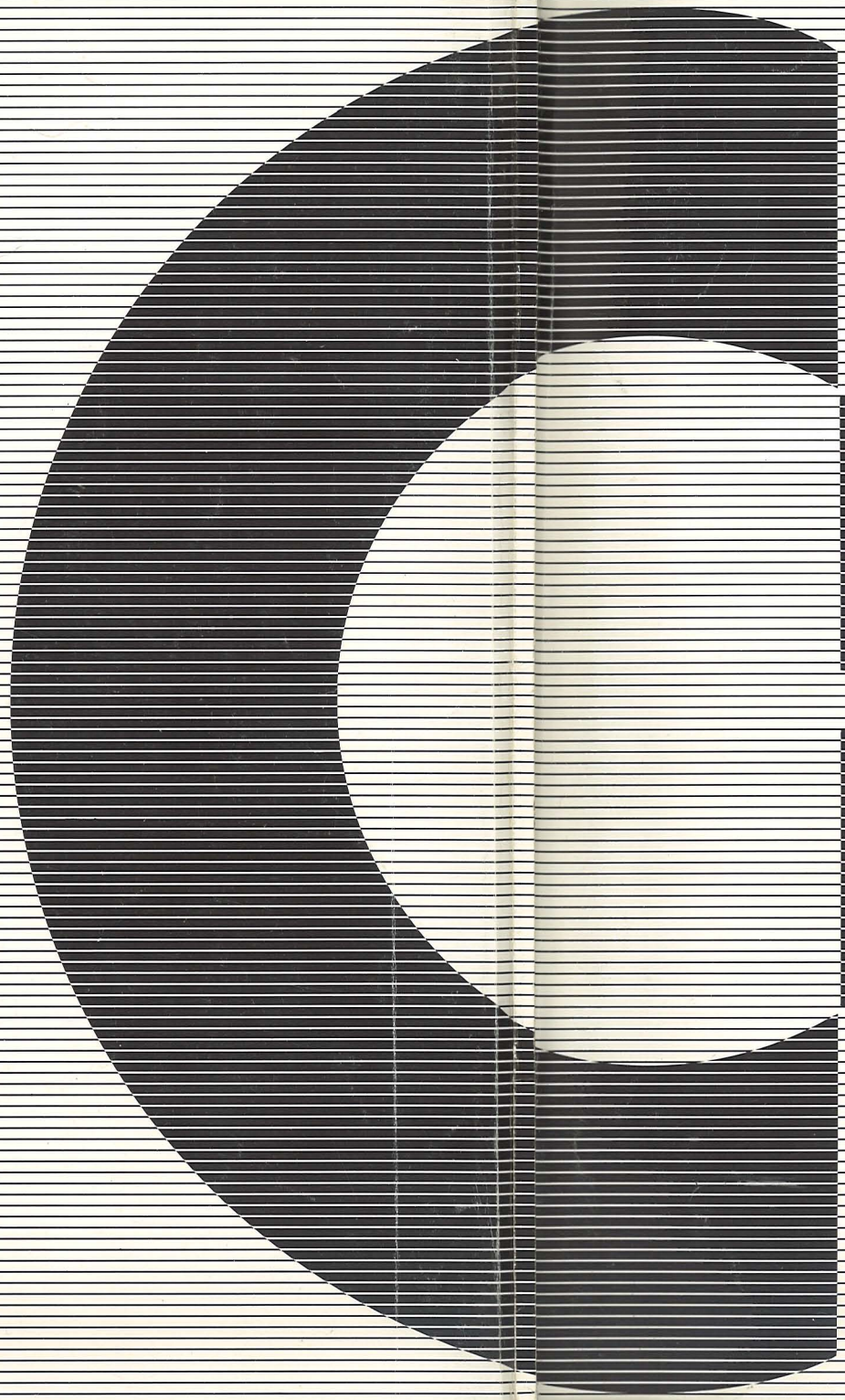


COMMODORE

FLOPPY DISK VC 1541

Bedienungshandbuch

Commodore

Commodore Büromaschinen GmbH
Lyoner Straße 38
D-6000 Frankfurt/M. 71

Commodore AG
Aeschenvorstadt 57
CH-4010 Basel

Commodore Büromaschinen GmbH
Kinskygasse 40–44
A-1232 Wien

P/N 1540031-03

RESCUE
ON
FRACTALUS

By...

1985

FLOPPY DISK VC 1541

Bedienungshandbuch

Commodore

INHALTSVERZEICHNIS

It's so easy

ABBILDUNGEN

TABELLEN

KAPITEL 1 : E I N L E I T U N G

ALLGEMEINE INFORMATION

Da die Floppy VC-1540 und die Floppy VC-1541 nahezu identisch sind, verwendet dieses Handbuch nur die Bezeichnung Floppy Disk VC-1541. Auch können der VC-20 oder der C-64 an die Floppy Disk VC-1541 angeschlossen werden. In diesem Handbuch wird als Rechner nur der C-64 angesprochen, gleiches gilt jeweils auch für den VC-20.

Durch den Erwerb der VC-1541 Single-Floppy haben Sie die Leistungsfähigkeit Ihres VC-Systems wesentlich erhöht. Um dieses System voll ausnutzen zu können, sollten Sie die Programmiersprache BASIC hinreichend beherrschen und mit der Bedienung der C-64-Zentraleinheit vertraut sein.

GERÄTEBESCHREIBUNG

Die VC-1541 ist ein intelligentes Einzeldiskettenlaufwerk, das zum Abspeichern großer Datenmengen geeignet ist. Es besteht zur Hauptsache aus einem elektronisch geregelten Antrieb, Schreib-Lesekopf mit Steuerung und der Elektronik, die die Schreib- und Leseroutinen überwacht. Das Laufwerk wird an den seriellen Bus des C-64 angeschlossen, der auch von Commodore-Druckern benutzt wird. Da es sich bei der VC-1541 um ein "intelligentes " Peripheriegerät handelt, wird beim Betrieb der Floppy kein Arbeitsspeicher in der C-64-Zentraleinheit belegt.

F r o n t p l a t t e

Auf der Frontplatte des Geräts befindet sich ein Schlitz, in den die Diskette eingeführt wird. Durch das Herunterdrücken einer am Schlitz angebrachten Klappe oder eines Knebels wird die Diskette an die Laufwerksachse angekoppelt.

Weiterhin sind zwei LEDS (LED = Light-Emitting-Diode = Leuchtdiode) auf der Frontplatte angebracht. Die grüne LED auf der linken Seite zeigt an, daß das Gerät eingeschaltet ist. Die rote LED unter dem Diskettenschlitz leuchtet kontinuierlich, wenn die Floppy aktiv ist und blinkt bei Fehlfunktion.

R ü c k s e i t e

Auf der Rückseite der Floppy befinden sich zwei parallel geschaltete, serielle Ein/Ausgänge. Weiterhin ist dort der Netzschalter und die Sicherung angebracht.

D i e D i s k e t t e

Als Diskette (auch: Floppy Disk o.ä.) wird die Standard 5-1/4-Zoll Diskette verwendet. Es müssen Disketten verwendet werden, die das SOFT-SEKTOR-FORMAT aufweisen.

Aufstellungsort

Das Diskettenlaufwerk sollte auf einer flachen, vibrationsfreien Unterlage aufgestellt werden. Wichtig ist weiterhin eine staubfreie Umgebung, da eine Ansammlung von Staubpartikeln die Funktion des Geräts empfindlich stören kann.

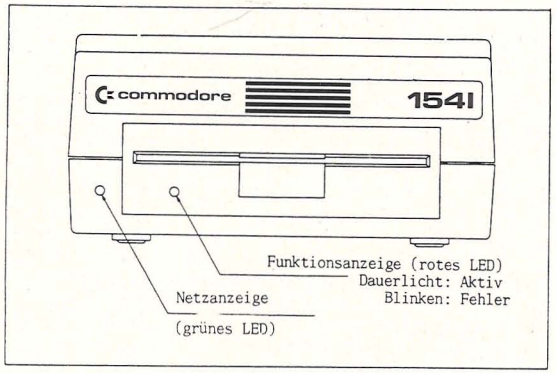


Abb 1. FRONTPLATTE

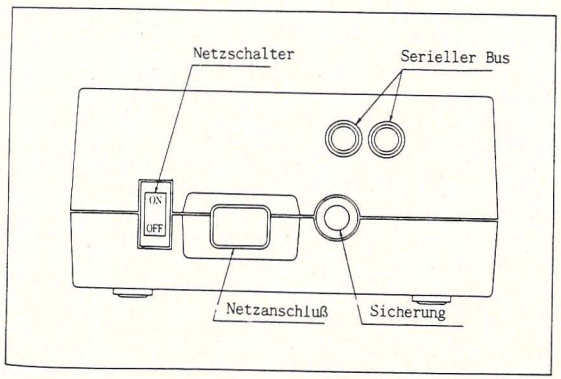


Abb.2 RÜCKSEITE

TABELLE 1. Technische Daten der VC-1541 Single Floppy

SPEICHERUNG	
Totale Speicherkapazität	174848 Bytes pro Diskette
Sequentielle Files	168656 Bytes pro Diskette
Relative Files	167132 Bytes pro Diskette
	65535 Records pro File
Einträge in die Directory	144 pro Diskette
Sektoren pro Spur	17 bis 21
Bytes pro Sektor	256
Spuren	35
Blöcke	683 (664 Blöcke frei)
IC's:	
6502	Mikroprozessor
6522 (2)	I/O, Timer
Puffer : 2114 (4)	2K RAM
ABMESSUNGEN	
Höhe	97 mm
Breite	200 mm
Tiefe	374 mm
STROMVERSORGUNG	
Spannung	100,120,220 oder 240 V
Frequenz	50 oder 60 Hz
Leistungsaufnahme	25 Watt
SPEICHERMEDIUM	
Disketten	Standard 5-1/4", einfache Schreibdichte

Behandlung der Disketten

Gehen Sie mit den Disketten vorsichtig um. Wenn Sie die nachstehend aufgeführten Regeln befolgen, schonen Sie Ihre Disketten und können sich jederzeit auf Ihre gespeicherten Daten verlassen :

1. Stecken Sie die Diskette nach der Entnahme aus dem Laufwerk sofort in die Papierhülle.

2. Bringen Sie die Diskette nicht in die Nähe starker magnetischer Felder, da hierdurch die Daten zerstört werden können.

3. Lassen Sie nie eine Diskette auf der Floppy oder auf Ihrem Computer liegen.

4. Benützen Sie zum Ausfüllen des Etiketts auf der Diskettenschutzhülle keinen Bleistift oder Kugelschreiber, sondern einen Filzschreiber.

5. Setzen Sie die Disketten nie hohen Temperaturen oder der Sonne aus.

6. Berühren Sie keinesfalls die Diskettenoberfläche und versuchen Sie diese auch nicht zu reinigen. Schon der Verlust kleinster Oberflächenpartikel kann zum Verlust von Daten führen.

7. Vergewissern Sie sich beim Ein- und Ausschalten der Floppy, daß sich keine Diskette im Laufwerk befindet.

Das Auspacken der VC-1541

Überprüfen Sie beim Auspacken des Geräts, ob die folgenden Teile in der Verpackung enthalten sind.

1. VC-1541 Single Floppy
2. TEST/DEMO-Diskette
3. Garantiekarte
4. Verbindungskabel
5. Handbuch

Sollten Sie ein Teil vermissen, so verständigen Sie bitte unverzüglich Ihren COMMODORE-Fachhändler.

Anpassung an den VC-20

Die Floppy 1541 befindet sich nach dem Einschalten im C-64 Modus. Da der VC-20 mit einer anderen Lesegeschwindigkeit arbeitet, muß die Floppy 1541 an den VC-20 angepasst werden. Dazu sind folgende Befehle einzugeben:

```
OPEN 15,8,15,"UI-" : CLOSE 15
```

Um wieder in den C-64 Modus zu gelangen, benutzen Sie folgende Befehle :

```
OPEN 15,8,15,"UI+" : CLOSE 15
```


KAPITEL 2 : INBETRIEBNAHME

Bevor Sie mit der Floppy zu arbeiten anfangen, sollten Sie sich vergewissern, daß sich diese in einem einwandfreien Zustand befindet. Dazu muß nach dem Initialisieren der Funktionstest mit der beigefügten TEST/DEMO-Diskette durchgeführt werden.

Anschluß der Floppy an den C-64

Das Kabel, mit dem die Floppy an den C-64 angeschlossen wird, ist dem Diskettenlaufwerk beigelegt.

BEMERKUNG: Werden an den seriellen Bus des C-64 durch "Verkettung" mehrere Geräte angeschlossen, so sollte die Floppy das erste Gerät hinter dem Computer sein.

Befolgen Sie beim Anschluß der Floppy an den Computer bitte Schritt für Schritt die folgenden Anordnungen :

1. SCHRITT : Schalten Sie Ihren Computer (und gegebenenfalls Ihre Modulbox) AUS.
2. SCHRITT : Stellen Sie das Diskettenlaufwerk möglichst nahe am Computer auf. Schließen Sie die Floppy NOCH NICHT an das Netz an.
3. SCHRITT : Verbinden Sie den seriellen Ausgang am C-64 und einen Ein/Ausgang der VC-1541 durch das beiliegende Kabel.
4. SCHRITT : Schließen Sie Ihre Floppy an das Netz an, aber schalten Sie NOCH NICHT ein.

Einschalttest

1. SCHRITT : Öffnen Sie die Klappe des Diskettenlaufwerks und vergewissern Sie sich daß keine Diskette darin ist.
2. SCHRITT : Schalten Sie die Modulbox EIN (falls vorhanden).
3. SCHRITT : Schalten Sie den C-64 EIN.
4. SCHRITT : Schalten Sie die VC-1541 EIN. Zuerst werden beide LEDs auf der Frontplatte aufleuchten. Nach etwa 1 sec sollte die rote LED an der Verschlußklappe ausgehen, während die grüne LED weiterhin leuchtet und damit anzeigt, daß die Floppy betriebsbereit ist. Sollte das rote LED blinken, so schalten Sie bitte das Gerät aus, warten etwa eine Minute und schalten wieder ein. Wenn sich auch daraufhin nicht der Normalzustand (rote LED: AUS ; grüne LED: AN) einstellt, wenden Sie sich bitte an Ihren COMMODORE-Händler.

BEMERKUNG: Um auszuschließen, daß die Fehlfunktion der Floppy auf eine Wechselwirkung mit anderen Geräten zurückzuführen ist, die ebenfalls an den seriellen Bus angeschlossen sind (z.B. ein Drukker), schließen Sie bitte diese Geräte während des Tests nicht an.

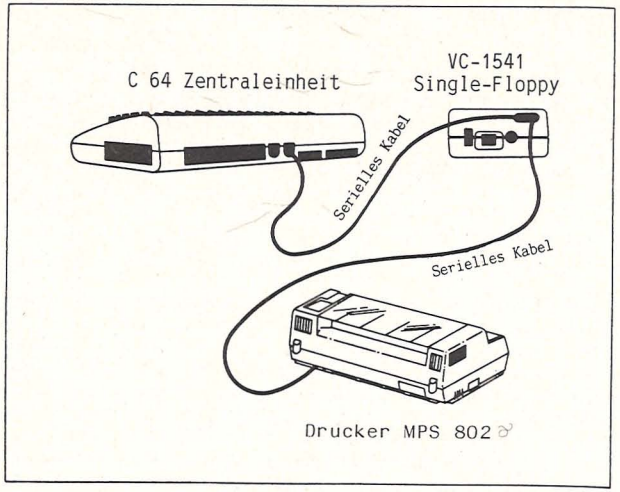


Abb.3 : Anschlußschema der Floppy an den C 64 und den Drucker MPS 802

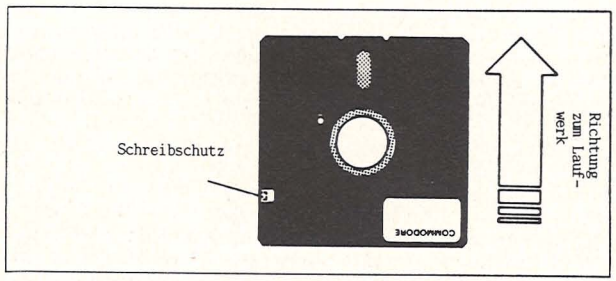


Abb. 4 : Orientierung der Diskette zum Laufwerk beim Einlegen

NIE VERGESSEN:
Zuerst C 64 und VC 1541 verbinden, dann Netzstecker in die Steckdose stecken!

Das Einlegen der Diskette

VORSICHT : SCHALTEN SIE DAS DISKETTENLAUFWERK NIE EIN (ODER AUS), WENN SICH EINE DISKETTE DARIN BEFINDET.

1. SCHRITT : Halten Sie die Diskette so, daß sich der Schreibschutz (siehe Abb.4) auf der linken Seite befindet und schieben Sie sie in dieser Orientierung in den Schlitz des Laufwerks.
2. SCHRITT : Schieben Sie solange weiter, bis Sie einen deutlichen Widerstand spüren und die Diskette einen festen Sitz hat.
3. SCHRITT : Drücken Sie nun die Klappe an der Laufwerksöffnung nach unten, bis sie mit einem hörbaren Klicken einrastet. Die Diskette ist nun fest an das Laufwerk angekoppelt.
4. SCHRITT : Um die Diskette wieder zu entnehmen, drücken Sie die Klappe leicht nach hinten (Richtung Laufwerk) und schieben sie gleichzeitig nach oben. Die Diskette wird dann automatisch aus dem Laufwerk herausgeschoben, sodaß Sie sie leicht entnehmen können.

Funktionstest

Wenn sich beim Einschalttest keine Fehler gezeigt haben, so können Sie jetzt mit dem Funktionstest weitermachen. Die Befehle, die in diesem Abschnitt benutzt werden, werden später detailliert erklärt. Alle Befehle müssen durch ein RETURN (Drücken der RETURN-TASTE) abgeschlossen werden.

BEMERKUNG: Die Befehle müssen genau in der Form eingetippt werden, in der sie unten aufgeführt sind. Ein "Space" (= Leertaste) zuviel oder zuwenig kann dazu führen, daß ein Befehl nicht interpretiert werden kann. Wenn durch Blinken der roten LED ein Fehler angezeigt wird, tippen Sie den letzten Befehl noch einmal ein. Wenn die LED daraufhin erlischt, haben Sie den Fehler erfolgreich korrigiert und können im Test fortfahren.

1. SCHRITT : Schieben Sie die DEMO-Diskette wie oben beschrieben in das Laufwerk ein.
2. SCHRITT : Tippen Sie ein : LOAD"PERFORMANCE TEST",8 (dann RETURN -Taste)

Auf dem Bildschirm erscheint daraufhin:

```
SEARCHING FOR PERFORMANCE TEST
LOADING
READY.
```

3. SCHRITT : Tippen Sie ein : RUN (dann RETURN-Taste)

Auf dem Bildschirm erscheint daraufhin:

```
--------------------
  PERFORMANCE TEST
--------------------

 INSERT SCRATCH

 DISKETTE IN DRIVE

PRESS RETURN
         WHEN READY

 DISK NEW COMMAND

WAIT ABOUT 80 SECONDS
```

Die Floppy fordert Sie auf, eine andere Diskette in das Laufwerk einzuschieben. Benutzen Sie eine neue Diskette, bzw. eine, die keine für Sie wertvollen Informationen enthält, da sie durch das Funktionstest-Programm "reformatiert" wird. Alle Daten, die auf einer Diskette gespeichert sind, gehen durch einen solchen Prozeß verloren. Das Funktionstest-Programm gibt der Diskette den Namen "TEST DISK". Sie kann mit Programmen oder Daten beschrieben werden.

Sie legen also die neue Diskette in das Laufwerk ein und drücken die RETURN-Taste. Als erstes wird die Diskette "formatiert". Eine fehlerfreie Durchführung der Formatierung wird angezeigt durch:

```
                          0 OK 0 0
DRIVE PASS
       MECANICAL TEST
```

Der Test wird nun automatisch weiter durchgeführt und der fehlerfreie Verlauf wird angezeigt durch:

```
OPEN WRITE FILE
                          0 OK 0 0

WRITING DATA
                          0 OK 0 0

CLOSE WRITE DATA
                          0 OK 0 0

OPEN READ FILE
                          0 OK 0 0

READING DATA
                          0 OK 0 0

SCRATCH FILE
                          1 FILE SCRATCHED 1 0

WRITE TRACK 35
                          0 OK 0 0
```



```
WRITE TRACK 1
                          0 OK 0 0

READ TRACK 35
                          0 OK 0 0

READ TRACK 1
                          0 OK 0 0

UNIT HAS PASSED
   PERFORMANCE TEST!

PULL DISKETTE FROM

 DRIVE BEFORE TURNING

 POWER OFF

READY.
```

4. SCHRITT : Entnehmen Sie nun die Diskette und stecken Sie sie zurück in die Staubschutzhülle.
5. SCHRITT : Wenn in der letzten Phase des Tests irgendwelche Probleme aufgetreten sind, so gehen Sie zurück zum 1.Schritt und wiederholen den gesamten Vorgang. Sollten mehrere Versuche nicht zum Erfolg führen, so setzen Sie sich bitte mit Ihrem COMMODORE-Händler in Verbindung.

KAPITEL 3 : ARBEITEN MIT DEM DISKETTENLAUFWERK

Die Steuerung der Floppy erfolgt über den C-64 in folgender Weise:

* BASIC-Kommandos von der Tastatur des C-64
* BASIC-Kommandos in Programmen
* Spezielle Disketten-Kommandos

Alle drei Methoden sollen in diesem Kapitel beschrieben werden.

Zum besseren Verständnis dieses Kapitels sollten Sie über folgende Punkte Bescheid wissen:

1. Bedienung der C-64-Zentraleinheit
2. Elementares Programmieren in BASIC
3. Öffnen und Schließen von Files

Als erstes sollen zwei Begriffe erläutert werden, die wichtig für das Verständnis der Methode sind, mit der Hilfe von Commodore-Diskettenlaufwerken Daten zu verwalten. Es handelt sich hierbei um die "Block- Availability-Map" (BAM = Verzeichnis der verfügbaren Blöcke) und das "Disk Operating System" (DOS = Betriebssystem).

Die Block Availability Map (BAM)

Die BAM enthält ein Verzeichnis des freien und belegten Speicherplatzes der Diskette. Wenn das System Information auf einer Diskette speichern will, so wird automatisch die BAM vom DOS abgefragt, ob und wieviel Speicherplatz noch frei ist. Im Falle einer Speicherung wird die BAM anhand der neuen Werte korrigiert, d.h. ,der von den neuen Daten benötigte Speicherplatz wird als belegt gekennzeichnet. Wird jedoch bei der Abfrage festgestellt, daß die zur Speicherung vorgesehene Datenmenge zu groß ist, so wird eine Fehlermeldung ausgegeben.

Die BAM wird beim Formatieren einer Diskette erstellt und beim Initialisieren der Diskette in den DOS-Speicher geschrieben. Die BAM ist auf der Diskette auf Spur 18, Sektor 0 gespeichert und belegt einen Speicherplatz von 128 Bytes. Alle die BAM betreffenden Änderungen werden zuerst im DOS-Speicher (in der Floppy) registriert und beim SAVEen (Abspeichern) eines Programms bzw. beim CLOSEn (Schließen) eines SEQentiellen Files auf die Disketten-BAM geschrieben.

Das Disk Operating System (DOS)

Das DOS verwaltet den Austausch von Informationen zwischen den Kontrollorganen der Floppy und dem Computer.

So wählt das DOS z.B. die Kanäle bei Ein/Ausgabeoperationen, sucht auf den entsprecheneden Befehl hin die Directory (das "Inhaltsverzeichnis" der Diskette), löscht und kopiert Files.

Floppy Systembefehle

Die folgenden Befehle werden beim File-"Handling" und der Verwaltung der Diskette verwendet

	BEFEHL	FUNKTION
Disketten-Ebene	NEW	Formatiert die Diskette
	INITIALIZE	Initialisiert die Diskette
	LOAD"$",8	Liest die Directory
	VALIDATE	Rekonstruiert die BAM
	COPY	Kopiert Files (ermöglicht auch das Verknüpfen von Files)
File-Ebene	RENAME	Umbenennung eines Files
	SCRATCH	Löschen eines Files

NEW

Bevor Sie mit einer neuen Diskette arbeiten können, müssen Sie sie erst mit NEW formatieren. Dazu wird die Struktur, in der die Daten auf die Diskette gespeichert werden, sowie der Name und der ID-Code, mit dem die Diskette von der Floppy identifiziert wird, auf die Diskette geschrieben. NEW organisiert die Diskette in Spuren (Tracks) und Sektoren und legt die BAM an. Der Befehl NEW kann durch N abgekürzt werden.

BEMERKUNG: Alle in diesem Kapitel behandelten Befehle werden der Floppy unter Verwendung eines PRINT#-Befehls übermittelt. Vor der Ausgabe dieses Befehls muß mit OPEN1,8,15 ein File (in diesem Fall das File Nr.1) eröffnet werden. Sie können dann mit

```
PRINT#1,"Befehlsstring"
```

den Systembefehl an die Floppy übermitteln (1: logische Filenummer, die innerhalb der Grenzen 1 bis 127 frei gewählt werden kann, 8: Geräteadresse der Floppy, 15: Sekundäradresse). Sollte beim Öffnen des Files die Fehlermeldung ? FILE OPEN ERROR erscheinen, so wurde vorher das File Nummer 1 nicht ordnungsgemäß geschlossen, was durch die Eingabe CLOSE 1 nachgeholt werden muß.

Der NEW-Befehl hat das Format: PRINT#lfn,"N:dn,xx"

lfn : Logische Filenummer
dn : Name der Diskette (max. 16 Zeichen)
xx : Ein zwei Zeichen langes Identifizierungsmerkmal (ID)

Der NEW-Befehl (mit ID) wird bei neuen Disketten oder solchen, die reformatiert werden sollen, angewendet. Bei bereits formatierten Disketten kann der NEW-Befehl ohne ID angewendet werden. Hierbei wird nur die Directory gelöscht und in der BAM alle Blocks als nicht belegt gekennzeichnet. Der NEW-Befehl ohne ID braucht erheblich weniger Zeit als der mit Angabe der ID.

1. BEISPIEL : OPEN15,8,15
PRINT#15,"N:TESTDISK,88"

Diese Befehle öffnen mit der Sekundäradresse 15 den Befehls- und Fehlerkanal zum Diskettenlaufwerk. Eine Diskette wird formatiert, und mit dem Namen TESTDISK und der Identifikation 88 versehen. Man kann die beiden Befehle auch in einem zusammenfassen :

2. BEISPIEL : OPEN15,8,15,"N:TESTDISK,88"

Soll beim Reformatieren einer Diskette nur der Name, nicht aber die ID geändert werden, so läßt man die ID einfach weg:

3. BEISPIEL : OPEN15,8,15,"N:NEUE TESTDISK"

Die Diskette bekommt den Namen NEUE TESTDISK zugeordnet und die Directory und die BAM werden gelöscht. Dieses abgekürzte Verfahren funktioniert jedoch nur bei einer bereits formatierten Diskette.

ACHTUNG: WIRD DAS FORMATIEREN EINER DISKETTE ABGEBROCHEN, WEIL EINE SCHLECHTE DISKETTE VERWENDET ODER DIE LAUFWERKSKLAPPE GEÖFFNET WURDE (ROTE LED BLINKT), SO MUSS DIE 1541 VOR DEM NÄCHSTEN FORMATIEREN AUS- UND WIEDER EINGESCHALTET WERDEN.
SOLL DAS FORMATIEREN UNTER PROGRAMM-KONTROLLE ERFOLGEN, SO IST VOR DEM FORMATIER-KOMMANDO FOLGENDE BEFEHLSZEILE EINZUFÜGEN:

OPEN1,8,15:PRINT#1,"M-W"CHR$(81)CHR$(0)CHR$(1)CHR$(255):CLOSE1

I N I T I A L I Z E

Bei der Initialisierung wird die Disketten-BAM in den Speicher des Diskettenlaufwerks geschrieben. Eine fehlerhafte Initialisierung führt zu der Fehlermeldung DISK ID MISMATCH ERROR und unter Umständen zu Datenverlust.

Die Floppy initialisiert die neu eingelegte Diskette selbsttätig, wenn sie sich in der ID von der vorhergehenden Diskette unterscheidet, da die Disketten an Hand der ID von der Floppy identifiziert werden. Wenn alle Disketten eine unterschiedliche ID besitzen, brauchen Sie also nie "von Hand" zu initialisieren.
Der Befehl INITIALIZE kann durch I abgekürzt werden.

Das Format des Initialisierungbefehls ist :

PRINT#lfn,"INITIALIZE"

lfn : logische Filenummer

BEISPIEL : OPEN15,8,15
PRINT15,"I"

Sie können beide Befehle zu einem zusammenfassen:

OPEN15,8,15,"I"

D i e D i r e c t o r y

Die Directory einer Diskette enthält folgende Informationen:

1. Diskettennamen
2. Disketten ID
3. Nr. der DOS Version
4. Belegte Blocks
5. Filenamen
6. Filetyp
7. Freie Blocks

Das Laden der Directory in den Rechner geschieht durch die Eingabe von

LOAD"$",8

mit nachfolgender (RETURN) -Taste. Beachten Sie bitte, daß hierdurch ein im Rechnerspeicher befindliches Programm zerstört wird. Der Inhalt der Directory wird nach Eingabe des Befehls

LIST

mit nachfolgender (RETURN) -Taste auf dem Bildschirm sichtbar gemacht.

Es ist sinnvoll, die Diskette zusammen mit der ausgedruckten Directory aufzubewahren. Die Ausgabe der Directory auf einen Drucker geschieht am bequemsten mit der untenstehenden Befehlsfolge:

LOAD"$",8	Die Directory wird in den Arbeitsspeicher des Rechners geladen
OPEN4,4	Ein File mit der Filenummer 4 wird eröffnet und der Gerätenummer 4 zugeordnet (in den meisten Fällen besitzt der Drucker diese Gerätenummer)
CMD4	Alle Daten, die normalerweise auf den Bildschirm ausgegeben werden, werden nun auf das Peripheriegerät umgelenkt, das der Filenummer 4 zugeordnet ist (in unserem Fall also auf den Drucker)
LIST	Die Directory wird ausgedruckt
PRINT#4	Der CMD-Befehl wird aufgehoben
CLOSE4	Das File 4 wird geschlossen

V A L I D A T E

Bei der Ausführung des VALIDATE-Befehls werden die Inhalte aller auf einer Diskette gespeicherten Files Block für Block gelesen. Blocks, die als belegt gekennzeichnet, aber nicht mit einem Filenamen verknüpft sind, werden für das Abspeichern neuer Daten freigegeben. Weiterhin werden Files, die nicht ordnungsgemäß geschlossen wurden, aus der Directory gelöscht.

Wird der VALIDATE-Befehl ordnungsgemäß durchgeführt, so wird eine neue BAM erzeugt und auf der Diskette abgespeichert. Bei Anzeige eines READ ERROR wird der Vorgang abgebrochen und die Diskette bleibt unverändert. Wird in einem solchen Fall mit der Diskette weitergearbeitet, so muß sie neu initialisiert werden.

Der VALIDATE-Befehl hat das Format:

```
PRINT#1fn,"VALIDATE"
```

Bemerkung: VALIDATE kann durch V abgekürzt werden.

```
BEISPIEL:  OPEN1,8,15
           PRINT#1,"V"
```

Sie können die beiden Befehle zu einem zusammenfassen und verkürzt schreiben:

```
OPEN1,8,15,"V"
```

C O P Y

Mit Hilfe des COPY-Befehls können Sie ein File (unter einem beliebigen Namen) innerhalb einer Diskette kopieren. Sie können den Befehl auch benutzen, um aus mehreren sequentiellen Datenfiles (maximale Anzahl: 4) ein neues zu erzeugen. Der COPY-Befehl kann durch C abgekürzt werden.
Das Format des COPY-Befehls lautet beim Kopieren eines Files:

```
PRINT#1fn,"C:nfn=afn"
```

Möchte man mehrere Files zu einem zusammenfassen, so benutzt man das Format

```
PRINT#1fn,"C:nfn=afn1,afn2,..."

lfn  : logische Filenummer
nfn  : neuer Filename
afn  : alter Filename
afn1, afn2,... : alte Filenamen
```

BEISPIEL : PRINT#1,"C:NEUFILE=ALTFILE"

BEISPIEL : PRINT#1,"C:SUMMENFILE=FILE1,FILE2,FILE3"

Hat man vor, mehrere Files zu einem zusammenzufassen, so muß man kurze Filenamen wählen, da ein Befehlsstring maximal 40 Zeichen lang sein darf.

R E N A M E

Der RENAME-Befehl belegt ein bereits existierendes File mit einem neuen Namen. RENAME kann im Befehlsstring durch R abgekürzt werden. Der Befehl hat das Format:

```
PRINT#1fn,"R:nfn=afn"

lfn  : logische Filenummer
nfn : neuer Filename
afn : alter Filename
```

Bemerkung: Ein File, das mit dem RENAME-Befehl umbenannt wird, muß vorher mit dem CLOSE-Befehl geschlossen werden.

S C R A T C H

Mit dem SCRATCH-Befehl lassen sich Files löschen. Man kann einzelne, einige, oder alle Files einer Diskette in einem Arbeitsgang löschen. Die Abkürzung von SCRATCH ist S.

Der Befehl hat das Format:

```
PRINT#1fn,"S:fn1,fn2,..."

lfn : logische Filenummer
fn1,fn2,... : Namen der zu löschenden Files
```

BEISPIEL: PRINT#1,"S:FILE1,FILE2,FILE3"

Die 3 Files: FILE1, FILE2 und FILE3 werden gelöscht. Sie können dasselbe erreichen, wenn Sie eingeben:

BEISPIEL: PRINT#1,"S:FILE*"

Wenn Sie alle Files einer Diskette löschen wollen, so müssen Sie folgenden Befehlsstring verwenden:

BEISPIEL: PRINT#1,"S:*"

Bemerkung: Näheres über die Verwendung des Zeichens "*", des sogenannten "Jokers" erfahren Sie im Kapitel 8.

4.KAPITEL : DAS ABSPEICHERN UND LADEN VON PROGRAMMEN UND DATEN

Folgende BASIC-Befehle sind wichtig für den Austausch von Programmen und Daten zwischen dem C-64 und der Floppy:

```
OPENlfn,8,sa,"fn,ft,mode"      VERIFY"fn",8
CLOSElfn                       PRINT#lfn
LOAD"fn",8                     GET#lfn
SAVE"fn",8                     INPUT#lfn
```

```
lfn : logische Filenummer ( Zahl zwischen 1 und 255) (*)
fn  : Filename (maximal 16 Zeichen lang)
8   : Geräteadresse der Floppy  (**)
sa  : Sekundäradresse
ft  : Filetyp ( SEQ, USR oder PRG )
mode: READ (Abk.:R) Lesen ; WRITE (Abk.:W) Schreiben
      APPEND (Abk.:A) Anhängen
```

(*) : Bei der Anwendung des PRINT#-Befehls ist darauf zu achten, ob die logische Filenummer größer oder kleiner als 128 ist. Siehe dazu den Abschnitt über den PRINT#-Befehl.

(**): Die Geräteadresse der Floppy wurde werksseitig auf 8 eingestellt, sie kann von Ihnen oder durch Ihren COMMODORE-Vertragshändler in 9, 10 oder 11 umgeändert werden (siehe auch Kapitel 10).

S A V E

Anwendung: Abspeichern (Saven) eines Programms vom Arbeitsspeicher des Rechners auf die Diskette.

BEISPIEL : SAVE"TESTPROGRAMM",8

Das im Arbeitsspeicher des Rechners stehende Programm wird unter dem Namen TESTPROGRAMM auf die Diskette geschrieben.

Haben Sie ein Programm abgeändert und wollen das alte Programm unter Beibehaltung des Namens überschreiben, so müssen Sie vor den Filenamen das Zeichen "@" (engl. "at sign" aber auch "Klammeraffe" genannt) setzen.

BEISPIEL : SAVE"@:TESTPROGRAMM",8

V E R I F Y

Anwendung: Das im Arbeitsspeicher des Rechners stehende Programm wird Byte für Byte mit dem im VERIFY-Befehl angeführten Programmfile verglichen. Jedesmal nach dem Abspeichern eines Programms sollte ein VERIFY durchgeführt werden.

BEISPIEL : VERIFY"TESTPROGRAMM",8

Geben Sie statt des Filenamens das Zeichen "*" ein, so wird das zuletzt abgespeicherte Programm überprüft.

LOAD

Mit dem LOAD-Befehl kann man Programme von der Diskette in den Rechnerspeicher laden.
Der LOAD-Befehl hat das Format:

```
LOAD"fn",8,1

fn : Filename
8  : Geräteadresse
1  : Parameter für die Ladeadresse (wahlfreie Angabe)
```

BEISPIELE : LOAD"TESTPROGRAMM",8
ein Basic-Programm wird geladen

LOAD"M-PROGRAMM",8,1
ein Maschinen-Program wird an die gespeicherte Adresse geladen

OPEN

Sie haben im letzten Kapitel den OPEN-Befehl schon im Zusammenhang mit anderen Anweisungen kennengelernt, die insbesondere der Behandlung von Programmfiles dienen. Im vorliegenden Abschnitt geht es in der Hauptsache um Datenfiles, d.h. um das Schreiben von Daten auf Diskette und das Wiedereinlesen in den Rechnerspeicher.

Der OPEN-Befehl stellt den Zusammenhang zwischen einer logischen Filenummer und einer Geräteadresse her. Weiterhin werden Sekundäradressen spezifiziert und Filenamen, Filetyp und Modus (Schreiben oder Lesen) festgelegt.
Der OPEN-Befehl hat das Format:

```
OPENlfn,8,sa,"fn,ft,modus"

lfn : Logische Filenummer
8   : Geräteadresse
sa  : Sekundäradresse (eine Zahl zwischen 2 und 14)  (*)
fn  : Filename
ft  : Filetyp, z.B. SEQ bzw.S (Sequentiell)
mode: READ bzw. R (lesen) ; WRITE bzw. W (schreiben)
      APPEND bzw. A (anhängen von Daten)
```

(*) Die Sekundäradresse 15 öffnet den Befehls- und Fehlerkanal. Das führt dazu, daß der String, der einem solchen OPEN-Befehl folgt, bzw. dem zugehörigen PRINT#-Befehl, als Befehl interpretiert wird. Die Sekundäradressen 0 und 1 sind für das Laden (LOAD) und Speichern (SAVE) von Programmen bestimmt.

BEISPIELE : OPEN2,8,2,"TEXTFILE,S,W"

OPEN2,8,6,"DATEN,S,A"

OPEN3,8,5,"DATENFILE,U,R"

Wollen Sie den Inhalt eines Files überschreiben, den Filenamen jedoch beibehalten, so müssen Sie den "Klammeraffen" (@) verwenden:

BEISPIEL : OPEN4,8,7,"@:NEUFILE,S,R"
Es kann manchmal von Vorteil sein, den Stringausdruck im OPEN-Befehl bzw. Teile davon durch eine Stringvariable darzustellen.

```
BEISPIELE : BS$="MESSDATEN,S,W"
            OPEN,1,8,14,BS$

            BS$="MITGLIEDER"
            OPEN,3,8,9,BS$+",S,R"
```

CLOSE

Der CLOSE-Befehl schließt Files, die mit OPEN geöffnet wurden. Sein Format ist:

```
            CLOSElfn

            lfn : Logische Filenummer
```

Um Datenverlust zu vermeiden und um stets die größtmögliche Anzahl von Files zur Verfügung zu haben (max. 10 Files dürfen gleichzeitig geöffnet sein), sollte man Files schließen, nachdem sie nicht mehr benötigt werden.

Bemerkung : Eine spezielle Bedeutung hat das Schließen eines mit der Sekundäradresse 15 eröffneten Files. Hierdurch werden nämlich alle Datenkanäle einer Floppy auf einen Schlag geschlossen.

PRINT#

Mit dem PRINT#-Befehl lassen sich Daten (siehe auch den Abschnitt über SEQuentielle Files) oder Systembefehle übermitteln. Der PRINT#-Befehl hat das Format:

```
            PRINT#lfn,"Daten"

            lfn : logische Filenummer  (*)
            Daten : Befehle, Variablen (numerisch oder Strings)
```

(*) Bei Verwendung des PRINT#-Befehls zur Datenübertragung ist zu berücksichtigen, daß bei Verwendung einer logischen Filenummer zwischen 1 und 127 ein "carriage return" (CR) und bei Verwendung einer Filenummer zwischen 128 und 255 ein CR zusammen mit einem "line feed" (LF) gesendet wird. Da bereits ein CR als Beendigung des PRINT#-Befehls interpretiert wird, wird das LF fälschlicherweise in den nächsten Datensatz übernommen.Abhilfe läßt sich dadurch schaffen, daß man den Befehl mit einem Semikolon (;) abschließt, dadurch das CRLF unterdrückt und in Form eines CHR$(13) ein "künstliches" CR hinterherschickt.

BEISPIEL : PRINT#137,"PER ASPERA AD ASTRA";CHR$(13);

INPUT#

Mit dem INPUT#-Befehl läßt sich Information von einem Peripheriegerät, wie z.B. der Floppy in den Speicher des Rechners übermitteln. Er ist nur im Rahmen eines Programms (nicht im Direkt-Modus) verwendbar. Der INPUT#-Befehl hat das Format:

```
            INPUT#lfn,A$  oder   INPUT#lfn,A

            lfn  : logische Filenummer
            A$   : Stringvariable
            A    : numerische Variable
```

Mit dem INPUT#-Befehl können auch mehrere Variable zugleich in den Rechnerspeicher eingelesen werden:

```
            INPUT#lfn,A$,B$,...,A,B,..
```

Strings bzw. numerische Variable, die mit Hilfe dieses Befehls von der Floppy gelesen werden, können nur dann als getrennt erkannt werden, wenn sie auf der Floppy durch "carriage returns" getrennt sind.

```
BEISPIEL : 50 INPUT#2,A
           Liest das nächste Datum (das numerisch sein muß) und
           ordnet es der Variablen A zu.

BEISPIEL : 10 INPUT#7,A$
           Liest das nächste Datum und ordnet es der String-
           variablen A$ zu

BEISPIEL : 40INPUT#3,B,C$
           Liest die nächsten beiden Daten und ordnet sie der nu-
           merischen Variablen B und dem String C$ zu.
```

Der INPUT-Befehl akzeptiert nur Strings, die eine Länge von maximal 88 Zeichen aufweisen. Im Falle längerer Strings muß der GET#-Befehl angewendet werden.

GET#

Der GET#-Befehl dient dazu, Strings Byte für Byte von einem Peripheriegerät wie z.B. der Floppy in den Rechnerspeicher einzulesen. GET# kann nur im Rahmen eines Programms angewendet werden. Der Befehl hat das Format:

```
GET#lfn,A$

GET#lfn,A

lfn : logische Filenummer
A$  : Stringvariable
A   : numerische Variable
```

GET# kann benutzt werden, um Strings, die länger als 88 Zeichen und damit zu lang für den INPUT#-Befehl sind, einzulesen. Der folgende Programmteil ist ein Beispiel für eine Unterroutine, mit der es möglich ist, einen String mit einer Länge von 254 Zeichen in den Rechnerspeicher einzulesen, wo er der Stringvariablen AA$ zugeordnet wird:

```
10 OPEN1,8,5,"FILE,S,R"
20 AA$=""
30 FORI=1to254
40 GET#1,A$:IFA$=""THENA$=CHR$(0)
50 AA$=AA$+A$
60 NEXT
70 CLOSE1
```


KAPITEL 5 : FORTGESCHRITTENES PROGRAMMIEREN MIT DER FLOPPY

D a s C O M M O D O R E D i s k O p e r a t i n g S y s t e m

(D O S)

Das File-System wird durch Kanäle organisiert, die mit Hilfe des OPEN-Statements geöffnet werden. Das DOS reserviert je nach Filetyp ein oder zwei Pufferspeicher für jeden Kanal. Steht kein Pufferbereich mehr zur Verfügung, so wird die Fehlermeldung: NO CHANNEL ausgegeben. Drei der acht Pufferspeicher werden vom DOS von vornherein belegt für die Block Availability Map (BAM), die Befehlskanal Ein/Ausgabe und die "Job-Schlange" des Disk-Controllers, die für eine optimale Abwicklung der verschiedenen Ein/Ausgabeoperationen sorgt. Die im OPEN-Befehl angegebene Sekundäradresse wird vom DOS zur Kennzeichnung der Kanäle verwendet. Die Größe der Sekundäradresse wird jedoch nur als Identifizierungsmerkmal eines bestimmten Kanals verwendet und muß nicht mit der internen Kanalnummer identisch sein. Bei Eingabe des LOAD- bzw. SAVE-Befehls werden als Sekundäradressen automatisch 0 bzw. 1 gewählt. Die verbleibenden Sekundäradressen 2 bis 14 können dazu verwendet werden, bis zu 5 weitere Datenkanäle zu öffnen.

D i e D i r e k t - Z u g r i f f s - B e f e h l e

Bei den nun folgenden Befehlen wird die Organisation der Daten auf der Diskette nicht vom DOS übernommen, sondern kann vom Anwender selbst bestimmt werden. Voraussetzung für eine sachgemäße Anwendung dieser Befehle ist die Kenntnis des Diskettenaufbaus, der am Ende dieses Kapitels detailliert besprochen wird.

Alle nachstehend aufgeführten Befehle werden nach Eröffnung eines Files mit der Sekundäradresse 15 durch einen PRINT#-Befehl übermittelt:

Befehl	Abkürzung	Format
BLOCK-READ	B-R	"B-R:"ch,0,t,s
BLOCK-WRITE	B-W	"B-W:"ch,0,t,s
BLOCK-EXECUTE	B-E	"B-E:"ch,0,t,s
BUFFER-POINTER	B-P	"B-P:"ch,p
BLOCK-ALLOCATE	B-A	"B-A:"0,t,s
BLOCK-FREE	B-F	"B-F:"0,t,s
memory-write (*)	M-W	"M-W"adl/adh/nc/data
memory-read (*)	M-R	"M-R"adl/adh
memory-execute(*)	M-E	"M-E"adl/adh
USER	U	"Ui:param"

(*): Während die drei "memory"-Befehle n u r in der abgekürzten Form verwendet werden dürfen, kann man bei allen anderen Befehlen sowohl die abgekürzte wie auch die in der Spalte "Befehl" angegebene ausgeschriebene Form verwenden.

Die Parameter in den Formatanweisungen haben folgende Bedeutungen:

ch : Kanalnummer im DOS, die identisch mit der Sekundäradresse im OPEN-Befehl ist
O : Nummer des Floppy-Laufwerks
t : Nummer der Spur ("track") auf der Diskette (eine Zahl zwischen 1 und 35)
s : Nummer des Sektors auf der Diskette (je nach Spur eine Zahl zwischen 0 und 20; siehe dazu Tabelle 4)
p : Position des Buffer-Pointers
adl : Low Byte der Adresse
adh : High Byte der Adresse
nc : Anzahl der Zeichen (zwischen 1 und 34)
data : Daten in hexadezimaler Darstellung. Die Daten werden byteweise mit Hilfe der CHR$-Funktion dargestellt, d.h.mit CHR$(160) wird hexadezimal AO übermittelt.
i : Index der User-Sprung-Tabelle
param: Parameter, die mit dem USER-Befehl verknüpft sind.

Die Parameter können wahlweise durch ein Komma (,) oder ein Semikolon (;) getrennt werden. In den nachfolgenden Beispielen wird das Semikolon benutzt.

BLOCK-READ

Dieser Befehl erlaubt direkten Zugriff zu jedem Datenblock auf der Diskette. Zusammen mit anderen Direkt-Zugriffs-Befehlen kann so ein File-System aufgebaut werden, in dem jeder beliebige Block direkt angesprochen wird. Der B-R-Befehl liest einen Zeiger (Pointer), der in der O-Position eines jeden Blocks steht und der auf das letzte zu lesende Zeichen zeigt. Nachdem der Block bis zu dieser Position (durch einen GET#- oder INPUT#-Befehl) gelesen wurde, wird ein "End-Or-Identify" (EOI) an den Rechner gesendet. Hierdurch wird der INPUT#-Befehl abgeschlossen und die Status-Variable auf 64 gesetzt.

BEISPIEL : "B-R:"5;0;18;0

Der Datenblock von Spur 18, Sektor 0 wird in den zu Kanal 5 gehörenden Puffer eingelesen. Eröffnet man nun ein File unter Verwendung der Sekundäradresse 5, so kann man die Daten mit GET# oder INPUT# in den Rechnerspeicher einlesen.

BLOCK-WRITE

Bei der Ausführung dieses Befehls wird der momentane Puffer-Zeiger (Buffer-Pointer) als Zeiger auf das letzte Zeichen interpretiert und in die O-Position des Puffers geschrieben. Daraufhin wird der Pufferinhalt in den spezifizierten Block auf die Diskette geschrieben. Der Puffer-Zeiger zeigt nach Ausführung des Befehls wieder auf die Position 1.

BEISPIEL : "B-W:"7;0;35;10

Der Inhalt des zu Kanal 7 gehörenden Pufferspeichers wird in den Block von Spur 35, Sektor 10 geschrieben.

BLOCK-EXECUT

Dieser Befehl lädt Programm-Routinen von der Diskette in den Floppy-Speicher und führt sie nach Abschluß des Ladens aus. Das Programm muß mit einer RTS-Anweisung (Return-From-Subroutine) abgeschlossen werden.

BEISPIEL : "B-E:"6;0;1;10

Lädt den Datenblock von Spur 1, Sektor 10 in den zu Kanal 6 gehörenden Puffer und führt die in den Daten enthaltenen Anweisungen aus.

BUFFER-POINTER

Mit diesem Befehl lässt sich der Wert des Puffer-Zeigers (Buffer-Pointers) verändern. Auf diese Weise ist es möglich, einen Block in Bereiche zu unterteilen, auf die beliebig zugegriffen werden kann.

BEISPIEL : "B-P:"2,1

Der Zeiger des zu Kanal 2 gehörenden Puffers wird auf den Beginn des Speicherbereichs gesetzt.

BLOCK-ALLOCATE

Mit diesem Befehl können in der BAM (im DOS-Speicher) Blocks als belegt gekennzeichnet werden. Beim Schließen eines Schreib-Files wird dieser Eintrag in die Disketten-BAM übernommen.

BEISPIEL : "B-A:"0;10;0

Der Block auf Spur 10, Sektor 0 soll als belegt gekennzeichnet werden.

Wenn ein Block markiert werden soll, der schon vorher als belegt gekennzeichnet wurde, so wird durch Erhöhen von Spur- und Sektornummer der nächste freie Block aufgesucht. Die Position des so gefundenen Blocks kann zusammen mit der Fehlermeldung: NO BLOCK über den Fehlerkanal 15 ausgelesen werden. Zu diesem Zweck muß bei Anwendung des B-A-Befehls stets ein File mit der Sekundäradresse 15 geöffnet werden. Mit

INPUT#lfn,FN,FM$,FT,FS

können dann Spur (FT) und Sektor (FS) des nächsten frei verfügbaren Blocks ausgelesen werden. Ist kein freier Block mehr verfügbar, so werden FT, FS als 0,0 ausgegeben.

VORSICHT: Ein VALIDATE-Befehl gibt alle Blöcke wieder frei, die nicht zu ordnungsgemäß im Directory eingetragenen Files gehören.

BLOCK-FREE

Dieser Befehl hat den entgegengesetzten Effekt wie der B-A-Befehl. Ein vorher in der BAM als belegt gekennzeichneter Block wird freigegeben.

BEISPIEL : "B-F:"0;9;20

Der Block in Spur 9, Sektor 20 wird als frei gekennzeichnet.

Die BAM-Befehle der 1541 kommen in der Regel nur mit Schreibbefehlen gemeinsam vor. Sollte einmal ein Block-Free- oder Block-Allocate-Befehl ohne nachfolgenden Schreibbefehl auftreten, so ist ein Dummy-Zugriff auf die Diskette auszuführen, z.B.:

```
OPEN2,8,2,"@:dummy,s,r"2
PRINT#2,"x
CLOSE2
```

Die aktualisierte BAM wird immer erst beim Schließen eines Schreib-Kanals auf die Diskette zurückgespeichert.

"MEMORY"-Befehle

Die drei MEMORY-Befehle arbeiten byte-orientiert, sodaß man sie im Zusammenhang mit Maschinenprogrammen verwenden kann. Wenn man im Rahmen eines BASIC-Programms mit den MEMORY-Befehlen arbeiten will, muß man die einzelnen Bytes mit Hilfe der CHR$-Funktion übertragen. Es dürfen nur die Abkürzungen M-R, M-W und M-E benutzt werden; weder das Ausschreiben der Befehlsnamen noch die Verwendung des Doppelpunkts (:) ist zugelassen.

Memory-Write

Der M-W-Befehl erlaubt direkten Zugriff zum DOS-Speicher. Programm-Routinen können geladen und mit dem M-E- oder dem USER-Befehl gestartet werden. Bei jeder Anwendung des Befehl können bis zu 34 Bytes übertragen werden.

BEISPIEL : "M-W"CHR$(00)CHR$(7)CHR$(3)CHR$(32)CHR$(0)CHR$(17)

Drei Bytes werden in die Speicheradressen $0700, $0701, $0702 (hexadezimal) (dezimal 1792, 1793, 1794) geschrieben.

Memory-Read

Mit diesem Befehl kann das Byte gelesen werden, dessen Adresse im Befehlsstring steht. Es können sowohl Variable aus dem DOS, wie auch aus den Puffern ausgelesen werden. Der M-R-Befehl ändert den Inhalt des Fehlerkanals, da er diesen benutzt, um Information an den Rechner zu übermitteln. Der nächste GET#-Befehl, der den Fehlerkanal anspricht (Sekundäradresse 15), liest das Byte aus.

BEISPIEL : "M-R"CHR$(128);CHR$(0)

Liest das Byte, das unter der Adresse $0080 (dezimal: 128) gespeichert ist.

Memory-Execute

Startet die Ausführung einer Subroutine im DOS-Speicher. Die Subroutine muß mit RTS ($60) abgeschlossen werden.

BEISPIEL : "M-E"CHR$(128);CHR$(49)

Startet die Programmausführung bei $3180 (dezimal :12672)

USER

Mit dieser Befehlsgruppe kann man unter Benutzung einer Sprungtabelle (Tabelle 3) die Verbindung zur Maschinensprache des 6502 herstellen. Der Index n im Befehl Un definiert nach Tabelle 3 die Sprungadresse. Statt der Zahlen 1 bis 9 und dem Doppelpunkt (:) können auch die Buchstaben A bis J als Indizes benutzt werden.

Die Befehle U1 und U2 sind nahezu identisch (*) mit den B-R und B-W-Befehlen und haben deshalb auch das gleiche Format:

"U1:"ch;dr;t;s

"U2:"ch;dr;tr;s

Über die Befehle U1 bis U9 kann vom Anwender frei verfügt werden.

(*) U1 setzt den Pufferzeiger auf 255 und liest stets den gesamten Block in den Speicher ein.
U2 schreibt den Speicherinhalt in einen Block, ohne den Wert des Bytes in Position 0 zu ändern.

Tabelle 2 : Sprungtabelle

1.BEZEICHNUNG	2.BEZEICHNUNG	FUNKTION
U1	UA	Ersatz für BLOCK-READ
U2	UB	Ersatz für BLOCK-WRITE
U3	UC	Sprung nach $0500
U4	UD	Sprung nach $0503
U5	UE	Sprung nach $0506
U6	UF	Sprung nach $0509
U7	UG	Sprung nach $050C
U8	UH	Sprung nach $050F
U9	UI	Sprung nach $FFFA
U:	UJ	Power-up Vektor
UI+		Umschalten in C-64 Modus
UI-		Umschalten in VC-20 Modus

DIE DATENSTRUKTUR AUF DER DISKETTE

Die einzelnen Datenblocks der Diskette können mit Hilfe des Programms: "DISPLAY T&S" (= Anzeige von Spur Sektor), das sich auf der TEST/DEMO-Diskette befindet, gelesen werden. Die Tabelle 4 zeigt Anzahl und Nummerierung der Sektoren (Blocks) auf den verschiedenen Spuren.

Tabelle 3 : Die Belegung der Spuren mit Sektoren

SPUR	SEKTORNUMMER	ANZAHL DER SEKTOREN
1 bis 17	0 bis 20	21
18 bis 24	0 bis 18	19
25 bis 30	0 bis 17	18
31 bis 35	0 bis 16	17

Tabelle 4 : Format der VC-1541-BAM

Gespeichert auf : Spur 18; Sektor 0

BYTE	INHALT	BEDEUTUNG
0,1	18,01	Spur und Sektor des ersten Blocks der Directory
2	65	ASCII-Zeichen "A"; zeigt 1541-Format an
3	0	0-Flag
4-143		Bitmuster der belegten bzw. nicht belegten Blocks (0: belegt; 1: nicht belegt)

Tabelle 5 : Vorspann (Header) der Directory

Gespeichert auf : Spur 18; Sektor 0

BYTE	INHALT	BEDEUTUNG
144-161		Name der Diskette (ergänzt mit"geshifteten Spaces")
162,163		ID-Kennzeichnung der Diskette
164	160	"Geshifteter Space"
165,166	50,65	ASCII-Darstellung von 2A (gibt DOS-Version und Format an)
166,167	160	"Geshiftete Spaces"
177-255	0	Nicht benutzt, aufgefüllt mit Nullen

Bem: Die Positionen 180 bis 191 können mit ASCII-Zeichen belegt sein.

Tabelle 6 : Format der Directory

Gespeichert auf : Spur 18, Sektor 1

BYTE	INHALT
0,1	Spur und Sektor des nächsten Blocks der Directory
2-31	Eintrag des 1. Files
34-63	Eintrag des 2. Files
66-95	Eintrag des 3. Files
98-128	Eintrag des 4. Files
130-159	Eintrag des 5. Files
162-191	Eintrag des 6. Files
194-223	Eintrag des 7. Files
226-255	Eintrag des 8. Files

Tabelle 7 : Format eines Directory-Eintrags

BYTE	INHALT
0	Filetyp (*) oder-verknüpft ("geort") mit $80
1,2	Spur und Sektor des ersten Datenblocks
3-18	Filename (ergänzt mit "geshifteten Spaces")
19,20	Nur bei relativen Files benutzt
21	Nur bei relativen Files benutzt
22-25	Nicht benutzt
26,27	Spur und Sektor des neuen Files beim Überschreiben mit dem (a) ("Klammeraffe")
28,29	Anzahl der Blocks im File (Low Byte, High Byte)

(*) Filetypen : 0 = DELeted
1 = SEQuential
2 = PROGram
3 = USeR
4 = RELative

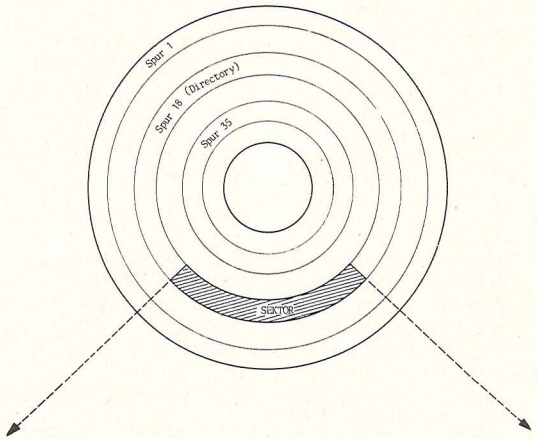
Tabelle 8 : Format eines sequentiellen Files

BYTE	INHALT
0,1	Spur und Sektor des nächsten sequentiellen Datenblocks
2-255	254 Bytes Daten

Tabelle 9 : Format eines Programmfiles

BYTE	INHALT
0,1	Spur und Sektor des nächsten Programmblocks
2-255	254 Bytes Programm im VC-Speicherformat (BASIC-Befehle als "Tokens"). Das Fileende ist durch 3 Nullen markiert.

Abb.5 zeigt den Aufbau eines Sektors auf einer Diskette, die für die VC-1541 formatiert wurde. Jeder Sektor enthält einen Datenblock von 254 Zeichen Daten bzw. Programm. Zwei zusätzliche Bytes bilden einen Zeiger zum nächsten Block; dadurch wird es möglich, mit Files zu arbeiten, die länger als ein Block sind. Dieser Aufbau ist bei PRG, SEQ und USER-Files der gleiche.
Ein Datenblock wird durch die Angabe von Block und Sektor adressiert. Eine VC-1541-Diskette enthält 35 Spuren (Ringe), die von 1 bis 35 durchnummeriert sind. Da die Spuren verschieden lang sind, enthalten sie eine verschiedene Anzahl von Sektoren (Tabelle 4.). Die Gesamtheit der Daten, die in einem Sektor enthalten sind, zeigt die Abb.5.



Zeiger zum Verknüpfen der Blocks in einem File

SYNC	08	ID1	ID2	SPUR	SEKTOR	PRÜF-SUMME	LÜCKE1	SYNC	07	BYTE 0	BYTE 1	254 BYTES DATEN	PRÜF-SUMME	LÜCKE2

Abb.5 : Aufbau eines Diskettensektors.
(Darstellung nicht maßgerecht)

KAPITEL 6 : SEQUENTIELLE FILES

Die Daten, aus denen ein sequentielles File aufgebaut ist, können nur in der Reihenfolge gelesen werden, in der sie geschrieben wurden. Ein sequentielles File kann nicht teilweise abgeändert werden, es besteht lediglich die Möglichkeit (durch Wahl des "mode" A), neue Daten an das Fileende anzuhängen.

Anlegen und Lesen eines sequentiellen Files (Siehe dazu auch die Beschreibung des OPEN- und des PRINT#-Befehls in Kapitel 4)

1. SCHRITT : Eröffnen eines Schreibfiles

```
OPEN2,8,2,"DATEN,S,W"

DATEN : Filename
S     : Sequentiell
W     : Schreiben (Write)
```

2. SCHRITT : Schreiben in das File

```
PRINT#2,A$,B$,C$
```

Die Inhalte der drei Stringvariablen werden in das File geschrieben

3. SCHRITT : Das File wird geschlossen

```
CLOSE2
```

4. SCHRITT : Das File mit dem Namen DATEN wird geöffnet

```
OPEN2,8,2,"DATEN,S,R"

R     : Lesen (Read)
```

5. Schritt : Die Daten werden in den Rechnerspeicher eingelesen

```
INPUT#2,X$,Y$,Z$
```

Die Werte von A$, B$ und C$ werden jetzt den Variablen X$, Y$ und Z$ zugeordnet.

6. SCHRITT : Die Statusvariable, die am Fileende den Wert 64 annimmt,wird abgefragt.

```
IF ST=64 THEN CLOSE2
```

Beispiel eines Programms, das mit sequentiellen Files arbeitet

Dieses Programm befindet sich unter dem Namen "SEQUENTIAL FILE" auf der TEST/DEMO-Diskette.

```
 1 REM ***************
 2 REM *   EXAMPLE
 3 REM * READ & WRITE
 4 REM * A SEQUENTIAL
 5 REM * DATA FILE
 6 REM ***************
10 PRINT"■■■■■■INITIALIZE DISK"
20 DIMA$(25)
30 DIMB(25)
40 OPEN15,8,15,"I0"
60 GOSUB 1000
70 CR$=CHR$(13)
80 PRINT
90 PRINT" ■WRITE SEQ TEST FILE"
95 PRINT
100 REM **************
101 REM *
102 REM *  WRITE SEQ
103 REM *  TEST FILE
104 REM *
105 REM **************
110 OPEN2,8,2,"@0:SEQ TEST FILE ,S,W"
115 GOSUB 1000
117 PRINT"ENTER A WORD, COMMA, NUMBER"
118 PRINT"ENTER WORD 'END' TO STOP"
120 INPUT"A$,B";A$,B
130 IFA$="END"THEN 160
140 PRINT#2,A$","STR$(B)CR$;
145 GOSUB 1000
150 GOTO 120
160 CLOSE 2
200 REM **************
201 REM *
202 REM *  READ SEQ
203 REM *  TEST FILE
204 REM *
205 REM **************
206 PRINT
207 PRINT"  READ SEQ TEST FILE"
208 PRINT
210 OPEN2,8,2,"0:SEQ TEST FILE ,S,R"
215 GOSUB 1000
220 INPUT#2,A$(I),B(I)
224 RS=ST
225 GOSUB 1000
230 PRINTA$(I),B(I)
240 IFR S=64 THEN 300
250 IF RS<>0 THEN 400
260 I=I+1
270 GOTO 220
300 CLOSE 2
310 END
```



```
400 PRINT"BAD■■DISK■■STATUS■■IS■■"RS
410 CLOSE 2
420 END
1000 REM *************
1001 REM *
1002 REM *  READ
1003 REM * THE ERROR
1004 REM *  CHANNEL
1005 REM *
1006 REM *************
1010 INPUT#15,EN,EM$,ET,ES
1020 IF EN=0 THEN RETURN
1030 PRINT"■■■■ERROR ON DISK"
1040 PRINTEN;EM$;ET;ES
1050 CLOSE 2
1060 END

READY.
```

Ein Direkt-Zugriffs-File wird mit Hilfe der in Kapitel 5 besprochenen BLOCK- und USER-Befehle angelegt. Um diese Befehle richtig anwenden zu können, ist es wichtig, den Datenfluß zwischen Rechner, Pufferspeicher und Diskette zu verstehen. In Abbildung 6. werden diese Zusammenhänge illustriert:

Zuerst werden Befehlskanal und Puffer eröffnet:

OPEN 5,8,15 : OPEN 4,8,3,"#7" (*)

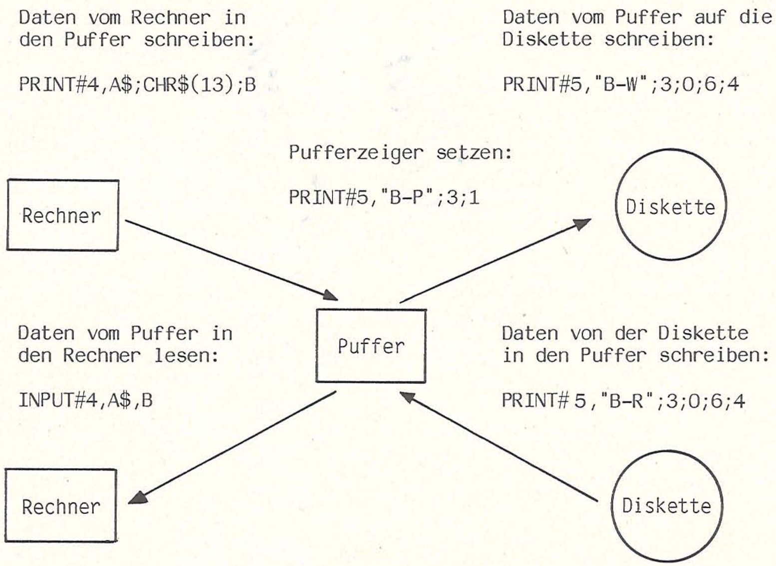


Abb. 6. Datenfluß zwischen Rechner, Puffer und Diskette

(*) Der im vorliegenden Beispiel angewandte OPEN-Befehl ordnet dem Kanal 3 (Sekundäradresse) den Puffer 7 zu. Ist dieser Puffer nicht verfügbar, so wird dies durch die Fehlermeldung : NO CHANNELS angezeigt. Benutzt man den OPEN-Befehl in der Form : OPEN4,8,3,"#", so wird der nächste freie Pufferspeicher dem Kanal 3 zugeordnet.

Programmbeispiel : Fileverwaltung mit Direkt-Zugriffs-Befehlen

Das folgende Programm verwaltet Files mit Hilfe von "Block"- und "USER"-Befehlen und kann z.B. zum Aufbau einer Adress-Datei verwendet werden. Unter dem Namen "RANDOM FILE" ist dieses Programm auf der TEST/DEMO-Diskette gespeichert. Die verschiedenen Programmteile und ihre Funktion sind der folgenden Aufstellung zu ennehmen (ZN = Zeilennummer) :

ZN	50 - 76	Menu
Zn	100 - 190	Erzeugen eines Files
ZN	200 - 280	Löschen eines Files
ZN	300 - 390	Suchroutine
ZN	5100 - 5190	Dateneingabe
ZN	5200 - 5260	Datenausgabe
ZN	5260 - 5320	Fehlerausgabe
ZN	5322 - 5390	Umrechnen der Recordnummer in Spur und Sektor

Um die Handhabung der Datenblocks zu vereinfachen, wurde das Konzept der Record-Nummer eingeführt. Aus dieser Nummer errechnet der letzte Programmteil (ab ZN 5322) Spur und Sektor des gesuchten Datensatzes. Die Spur 18 wird bei dieser Programmroutine automatisch ausgespart, sodaß die an dieser Stelle abgespeicherte Directory nicht überschrieben werden kann. Die Aufteilung der Record-Nummern auf die Spuren der Diskette zeigt Tabelle 11.

Tabelle 10 : Zuordnung der Record-Nummern zu den Diskettenspuren (Programm: RANDOM FILE)

SPUR	SEKTOR	SEKTOREN PRO SPUR	RECORD-NUMMERN
1 bis 17	0 bis 20	21	1-357
18 bis 24	0 bis 18	19	358-471
25 bis 30	0 bis 17	18	478-579
31 bis 35	0 bis 16	16	586-664

```
10 REM ************************
12 REM * RANDOM FILE EXAMPLE *
14 REM ************************
16 DIMI$(664):FD=0:FX=5:CH=2:FP=1
18 PRINT"________________________"
20 PRINT"  INSERT DATA SHEET"
22 PRINT"________________________"
24 PRINT"     START PRESS 'S'"
26 GETP$:IFP$<>"S"THEN26
28 OPEN15,8,15,"I0":OPEN2,8,2,"#"
30 PRINT"________________________"
32 PRINT"INDEX FILE OPERATION"
34 PRINT"________________________"
36 INPUT"     NEW SHEET? N   ";O$:PRINT""
38 IFO$="N"THEN46
40 IFO$<>"Y"THEN36
42 PRINT:PRINT"    WAIT!":FORI=1TO664
```

```
44 PRINT"■■■■■■■■■■■■■■■■■■■■        ■■■■■"I;:I$(I)="/"
45 PRINTI$(I):NEXT:GOTO50
46 OPEN5,8,5,"0:INDEX,S,R"
47 FORI=1TO664:INPUT#5,I$(I)
48 PRINT"■■■■■■■■■■■■■■■WAIT!     ■■■■■"I;I$(I)
49 NEXTI:CLOSE5
50 PRINT"■■________________________________"
52 PRINT"       JOB MENU          "
54 PRINT"________________________________"
56 PRINT
58 PRINT"       1=CREATE"
60 PRINT"       2=DELETE"
62 PRINT"       3=SEARCH"
64 PRINT"       0=END"
65 PRINT
66 INPUT"1 , 2 , 3 , 0    1■■■";O$
68 IFO$="0"THENCLOSE15:CLOSE5:CLOSE2:END
70 IFO$="1"THEN104
72 IFO$="3"THEN300
74 IFO$<>"2"THEN50
76 GOTO200
100 REM ************************
102 REM * MASTER FILE CREATE   *
103 REM ************************
104 PRINT"■■________________________________"
105 PRINT"  MASTER FILE CREATE"
106 PRINT"________________________________"
107 INPUT"RECORD NO. = 0■■■";F
109 IF F=0THEN170
110 INPUT"NAME       =.■■■";FB$(1)
120 INPUT"ADDRESS    =.■■■";FB$(2)
130 INPUT"ZIP        =.■■■";FB$(3)
132 INPUT"TEL        =.■■■";FB$(4)
134 INPUT"COMMENT    =.■■■";FB$(5)
140 GOSUB5200
150 I$(F)="1"
160 GOTO104
170 OPEN5,8,5,"@0:INDEX,S,W"
175 FORI=1TO664:PRINT#5,I$(I);CHR$(13);
180 PRINT"■■■■■■■■■■■■■■■WAIT!     ■■■■■"I;I$(I)
185 NEXT:CLOSE5
190 GOTO74
200 REM ************************
201 REM *  MASTER FILE DELETE *
202 REM ************************
210 PRINT"■■________________________________"
212 PRINT"  MASTER FILE DELETE"
214 PRINT"________________________________":PRINT
220 INPUT"RECORD NO. = 0■■■";F
230 IFF=0THEN260
235 IFI$(F)<>"1"THEN220
240 I$(F)="/":PRINT"■RECORD NO."F;"DELETE"
250 GOTO220
260 OPEN5,8,5,"@0:INDEX,S,W"
265 FORI=1TO664:PRINT#5,I$(I);CHR$(13);
270 PRINT"■■■■■■■■■■■■■■■WAIT!     ■■■■■"I;I$(I)
275 NEXT:CLOSE5
280 GOTO50
```



```
300 REM ************************
301 REM *  FILE SEARCH         *
302 REM ************************
310 PRINT"■■________________________________"
312 PRINT"         SEARCH        "
314 PRINT"________________________________":PRINT
320 INPUT"RECORD NO. = 0■■■";F
321 IFF=0THEN50
322 IFI$(F)<>"1"THEN320
325 GOSUB5100
360 PRINT"NAME      = ";FB$(1)
370 PRINT"ADDRESS   = ";FB$(2)
380 PRINT"ZIP       = ";FB$(3)
382 PRINT"TEL       = ";FB$(4)
383 PRINT"COMMENT   = ";FB$(5)
385 PRINT"------------------------"
390 GOTO320
5100 REM ************************
5105 REM * FDD BLOCK READ       *
5108 REM ************************
5110 GOSUB5330
5120 PRINT#15,"U1:";CH;FD;FT;FS
5130 PRINT#15,"B-P:";CH;FP
5140 GOSUB5270
5150 FORFI=1TOFX
5160 INPUT#CH,FB$(FI):
5180 NEXT
5190 RETURN
5200 REM ************************
5201 REM * FDD BLOCK WRITE      *
5202 REM ************************
5210 GOSUB5330
5220 PRINT#15,"B-P:";CH;FP
5230 FORFI=1TOFX:PRINT#CH,FB$(FI);CHR$(13);:NEXT
5240 PRINT#15,"U2:";CH;FD;FT;FS
5250 GOSUB5270
5260 RETURN
5270 REM ************************
5275 REM *  ERROR CHECK         *
5278 REM ************************
5280 INPUT#15,EN,EM$,ET,ES
5290 IFEN=0THENRETURN
5300 PRINT"ERROR STATUS:";EN;EM$;ET;ES
5310 INPUT"CONTINUE?";Y$:IFY$="Y"THENRETURN
5320 STOP
5322 REM ************************
5324 REM * SET TRACK & SECTOR   *
5326 REM ************************
5330 IFF<358THENF1=0:F2=22:F3=1:GOTO5370
5340 IFF>357ANDF<472THENF1=357:F2=20:F3=19:GOTO5370
5350 IFF>471ANDF<580THENF1=471:F2=19:F3=25:GOTO5370
5360 IFF>579THENF1=579:F2=18:F3=31
5370 FT=INT(((F-F1)-1)/(F2-1))+F3
5380 FS=F-F1-(FT-F3)*F2+(FT-F3-1)
5390 RETURN

READY.
```

KAPITEL 8 : Relative Datei

Was ist eine "RELATIVE DATEI" ?

Die Relative Datei ist eine besondere Art des DIREKT-ZUGRIFFS, die leicht zu programmieren ist. Dabei wird ein Satz wahlfrei, durch seine Satznummer identifiziert, gelesen oder geschrieben. Eine Relative Datei ist in gleichlange Sätze unterteilt, ein Satz wiederum kann in Felder untergliedert sein.

Aufbau und Verwaltung einer Relativen Datei :

Die Dateiverwaltung übernimmt das DOS der Floppy Disk. Dazu werden SIDE SEKTORS (Zeigerblöcke) angelegt. Ein Sektor kann auf 120 Datenblöcke zeigen, 6 dieser Sektoren können in einer Datei stehen.

Kapazität einer Relativen Datei :

Eine Datei kann aus 167132 Zeichen bestehen, d.h. einer Diskettenseite. Die Datensatzlänge beträgt 1 - 254 Zeichen.

Tabelle 11 : Format der Relativen Datei

Byte	Inhalt
0,1	Spur und Sektor des nächsten Datensatzes
2-255	254 Datenbytes
	Ein leeren Satz enthält FF (hex) im ersten Byte
	Nicht belegte Bytes werden auf 00 (hex) gesetzt

Tabelle 12 : Format des SIDE SEKTORs

Byte	Inhalt
0,1	Spur und Sektor des nächsten Side Sektors
2	Nummer des Side Sektor (0 - 5)
3	Satzlänge
4,5	Spur und Sektor des ersten Side Sektor
6,7	Spur und Sektor des zweiten Side Sektor
8,9	Spur und Sektor des dritten Side Sektor
10,11	Spur und Sektor des vierten Side Sektor
12,13	Spur und Sektor des fünften Side Sektor
14,15	Spur und Sektor des sechsten Side Sektor
16-255	Spur- und Sektorzeiger auf 120 Datenblöcke

Anlegen einer Relativen Datei :

Die Datei wird mit dem OPEN Befehl erzeugt, im Befehlsstring wird die Satzlänge mit übergeben.

Format : OPEN lfn,8,sa,"fn,L,"+CHR$(satzlänge)

Beispiel : OPEN 2,8,2,"REL,L,"+CHR$(100)
Eine Relative Datei mit Satzlänge 100 wird angelegt.

Eröffnen einer existierenden REL Datei zum Lesen und Schreiben :

Format : OPEN lfn,8,sa,"fn"

Beispiel : OPEN 2,8,2,"REL"
Die Relative Datei REL wird zum Lesen und Schreiben eröffnet.

Positionieren auf einen Datensatz :

Da auf jeden einzelnen Satz wahlfrei zugegriffen wird, muß zunächst die entsprechende Satznummer bereit gestellt werden. Das DOS der Floppy erwartet diese Satznummer als "ZWEI-BYTE-ZAHL", deshalb muß die Satznummer erst umgerechnet werden.

Umrechnung Satzadresse in das "ZWEI-BYTE-FORMAT" :
HB = INT(satznummer/256) : LB = satznummer - HB * 256

Diese Satznummern werden über den Befehlskanal zum DOS der Floppy geschickt (der Befehlskanal muß mit OPEN eröffnet sein!).

Format : PRINT# lfn,"P"+CHR$(sa)+CHR$(LB)+CHR$(HB)+CHR$(byte)
- sa = Sekundäradresse der Relativen Datei
- LB = Satznummernteil (LOW-BYTE)
- HB = Satznummernteil (HIGH-BYTE)
- byte = Bytezahl (1 - 254) innerhalb des Satzes

Beispiel : PRINT#15,"P"+CHR$(2)+CHR$(10)+CHR$(0)+CHR$(1)
Es wird auf das 1. Byte im 10. Satz positioniert.

Schreiben die REL Datei :

Sätze werden mit PRINT# lfn, (satz) in die Datei geschrieben. Sollen mehrere Felder in einen Satz, müssen diese mit CHR$(13) = CR getrennt und mit e i n e m PRINT# Befehl abgesetzt werden.

Beispiel : PRINT#2,A$;CHR$(13);B$;CHR$(13);C
Die Felder A$, B$ und C werden in den gleichen Satz geschrieben.

Lesen der Relativen Datei :

Die Felder einer REL Datei werden mit GET# oder INPUT# gelesen.

Beispiel : INPUT#2,X$,Y$,Z
Die 3 Felder eines Satzes werden gelesen.

Fehlercodes bei REL Dateien :

Der Fehlercode 50, (siehe Kap. 9) muß besonders behandelt werden.

Beispiel-Programm mit RELATIVER DATEI

Programmerklärungen :

Zeilennr.	Kurzbeschreibung
60- 90	Satznummer errechnen und auf den Satz Positionieren
270-310	Satzlänge prüfen, REL Datei anlegen
350-390	10 Leersätze als Platzhalter erzeugen
430-500	Sätze in die Datei schreiben
470	wenn beim Positionieren der Fehler 50 auftritt, sollte über die Datei hinaus ein Satz geschrieben werden.
480	Begrenzung der Eingabelänge durch den INPUT# Befehl
540-640	Sätze aus der Datei lesen
580	siehe Zeile 470, nur beim Lesen
780-800	Fehlerkanal lesen und ausgeben, der Fehler 50 wird gesondert gezeigt.

```
10 PRINT"■":GOTO130:REM ZUM HAUPTPROGRAMM
20 REM  ------------------------------
30 REM  SATZNUMMER UMRECHNEN UND AUF
40 REM  DEN SATZ POSITIONIEREN
50 REM  ------------------------------
60 HB=INT(RN/256):LB=RN-HB*256
70 PRINT#15,"P"+CHR$(8)+CHR$(LB)+CHR$(HB)+CHR$(1)
80 GOSUB780
90 RETURN
100 REM  ------------------------------
110 REM  MENUE
120 REM  ------------------------------
130 OPEN15,8,15,"I0":CLOSE15
140 PRINT"■■  DATEIVERWALTUNG RELATIVER FILES C-64"
150 PRINT"  ================================="
160 IFXX=0THENPRINT"    ■■■1■  ANLEGEN EINER DATEI"
170 PRINT"    ■■■2■  DATENSAETZE SCHREIBEN"
180 PRINT"    ■■■3■  DATENSAETZE LESEN"
190 PRINT"    ■■■4■  OEFFNEN EINER DATEI"
200 PRINT"    ■■■5■  PROGRAMMENDE"
210 GETA$:IFA$=""THEN210
220 ONVAL(A$)GOTO270,430,540,660,800
230 GOTO210
240 REM  ------------------------------
250 REM  RELATIVE DATEI ANLEGEN
260 REM  ------------------------------
270 PRINT"■■DATEINAME :■";:INPUTNA$:PRINT
280 PRINT"■SATZLAENGE :■";:INPUTL
290 IFL<1ORL>254THENPRINT"■MUSS ZWISCHEN 1...254 LIEGEN■":GOTO280
300 OPEN8,8,1,NA$+",L,"+CHR$(L)
310 OPEN15,8,15:GOSUB780
320 REM  ------------------------------
330 REM  SCHREIBEN VON 10 LEERSAETZEN
340 REM  ------------------------------
350 PRINT#15,"P"+CHR$(1)+CHR$(10)+CHR$(0)+CHR$(1)
360 PRINT#8,CHR$(255);
370 RM=INT(167100/L)
380 CLOSE8:CLOSE15
390 XX=1:GOTO140
```



```
400 REM  ------------------------------
410 REM  SAETZE SCHREIBEN
420 REM  ------------------------------
430 IFXX<1THENGOSUB710:GOTO140
440 OPEN15,8,15:OPEN8,8,8,NA$:GOSUB780
450 PRINT"■■■  SAETZE SCHREIBEN■":PRINT
460 PRINT"■■■■SATZNUMMER : (0=ENDE) ■";:INPUTRN:IFRN=0THEN740
470 GOSUB60:IFX=50THENPRINT"■ZU HOHE SATZNUMMER■":GOTO460
480 PRINT:PRINT"■EINGABEDATEN :  (MAX. 79 ZEICHEN)■"
490 INPUTD$:IFLEN(D$)>79THEND$=LEFT$(D$,79)
500 PRINT#8,D$:GOTO450
510 REM  ------------------------------
520 REM  SATZE LESEN
530 REM  ------------------------------
540 IFXX<1THENGOSUB710:GOTO140
550 OPEN15,8,15:OPEN8,8,8,NA$:GOSUB780
560 PRINT"■■■  SAETZE LESEN■"
570 PRINT"■■■■SATZNUMMER : (0=ENDE) ■";:INPUTRN:IFRN=0THEN740
580 GOSUB60:IFX=50THENPRINT"■ZU HOHE SATZNUMMER■":GOTO570
590 INPUT#8,D$
600 IFD$=CHR$(255)THEND$="LEERER DATENSATZ"
610 PRINT"■■GESPEICHERTE DATEN : ■":PRINT:PRINTD$
620 GOTO570
630 REM  ------------------------------
640 REM  DATEI EROEFFNEN
650 REM  ------------------------------
660 PRINT"■■DATEINAME : ■";:INPUTNA$
670 XX=2:GOTO140
680 REM  ------------------------------
690 REM  FEHLERMELDUNG
700 REM  ------------------------------
710 PRINT"■ZUERST MUSS EINE DATEI ANGELEGT WERDEN■"
720 FORI=1TO4000:NEXT
730 RETURN
740 CLOSE8:CLOSE15:GOTO140
750 REM  ------------------------------
760 REM  FEHLERKANAL LESEN
770 REM  ------------------------------
780 INPUT#15,X,X$,Y,Z:IFX<20ORX=50THENRETURN
790 PRINT"■FEHLER : ■";X,X$;Y,Z
800 CLOSE8:CLOSE15
810 END

READY.
```

KAPITEL 9 : FEHLERMELDUNGEN - DER "JOKER"

LESEN DES FEHLERKANALS

Das Blinken der roten LED der Floppy zeigt das Auftreten eines Fehlers an. Mit Hilfe des folgenden kleinen Programms können sie ermitteln, um welchen Fehler es sich handelt, und wenn dies nach Art des Fehlers möglich ist, wo er lokalisiert ist.

```
10 OPEN1,8,15
20 INPUT#1,A,B$,C,D
30 PRINTA,B$,C,D
40 CLOSE1
```

A : Nummer des Fehlers
B$: Fehlerbezeichnung im Klartext
C : Spur
D : Sektor

LISTE DER BEDEUTUNGEN DER FEHLERNUMMERN

 0 : Kein Fehler
 1 : Rückmeldung für Filelöschung (keine Fehlermeldung)
20 : Block-Header nicht gefunden
21 : SYNC-Zeichen nicht gefunden
22 : Datenblock nicht vorhanden
23 : Fehler in der Prüfsumme
24 : Byte falsch decodiert
25 : Fehlerhaftes Verify beim Schreiben
26 : Schreibversuch auf Diskette mit Schreibschutz
27 : Prüfsummenfehler im Header
28 : Datenblock ist zu lang
29 : ID der Diskette fehlerhaft
30 : Allgemeiner Syntax-Fehler
31 : Ungültiger Befehl
32 : Befehl ist zu lang
33 : Ungültiger Filename
34 : File existiert nicht
39 : Befehl kann nicht interpretiert werden
50 : Record existiert nicht
51 : Recordlänge zu klein
52 : Recordnummer ist zu hoch
60 : File bereits zum Schreiben geöffnet
61 : File nicht offen
62 : File nicht gefunden
63 : File existiert bereits
64 : Falscher Filetyp
65 : Kein Block mehr frei
66 : Falsche Spur- oder Sektornummer
67 : Falsche Spur- oder Sektornummer
70 : Kein Kanal mehr frei
71 : Fehler in der BAM
72 : Diskette oder Directory voll
73 : Angabe der DOS-Version
74 : Keine Diskette im Laufwerk

LISTE DER FEHLERMELDUNGEN UND BESCHREIBUNG

20 : READ ERROR
Header des gesuchten Datenblocks wird nicht gefunden.

21 : READ ERROR
SYNC-Markierung wird nicht gefunden. Gründe dafür können schlechte Justierung des Schreib/Lesekopfes, nicht formatierte Disketten oder schlechter Sitz der Diskette sein. Kann auch Hardware-Defekt anzeigen.

22 : READ ERROR
Dieser Fehler tritt im Zusammenhang mit den BLOCK-Befehlen auf und zeigt an, daß ein Block gelesen oder geprüft (verified) werden sollte, der nicht ordnungsgemäß geschrieben wurde.

23 : READ ERROR
Die Daten, die in den DOS-Speicher geschrieben wurden enthalten einen Prüfsummenfehler, d.h. ein Datenbyte oder mehrere sind fehlerhaft. Dieser Fehler kann auch Erdungsprobleme anzeigen.

24 : READ ERROR
Die Daten oder der Header wurden in den DOS-Speicher eingelesen, aber in den Datenbytes existieren fehlerhafte Bitmuster. Dieser Fehler kann auch Erdungsprobleme anzeigen.

25 : WRITE ERROR
Fehlende Übereinstimmung zwischen den Daten im DOS-Memory und den Daten auf der Diskette.

26 : WRITE PROTECT ON
Es wird versucht, auf eine Diskette zu schreiben, die mit einem Schreibschutz versehen ist.

27 : READ ERROR
Es liegt ein Fehler im Header des zu lesenden Datenblocks vor (Prüfsumme stimmt nicht). Dieser Fehler kann auch Erdungsprobleme anzeigen.

28 : WRITE ERROR
Nach dem Schreiben eines Datenblocks sucht der Controller die SYNC-Zeichenfolge des nächsten Datenblocks. Wenn er diese Synchronisationsmarkierung nicht innerhalb eines bestimmtem Zeitraums findet, wird diese Fehlermeldung ausgegeben. Grund dafür kann z.B. ein zu langer Datenblock oder ein Hardwarefehler sein.

29 : DISK ID MISMATCH
Die im DOS-Speicher vorhandene ID stimmt nicht mit der ID auf der Diskette überein. Die Diskette wurde nicht initialisiert, oder der Header ist fehlerhaft.

30 : SYNTAX ERROR
Das DOS kann den über den Befehlskanal geschickten Befehl nicht interpretieren.

31 : SYNTAX ERROR
Befehl wird nicht erkannt.

32 : SYNTAX ERROR
Befehl ist länger als 58 Zeichen.

33 : SYNTAX ERROR
Falsche Verwendung des "Jokers" im OPEN- oder SAVE-Befehl.

34 : SYNTAY ERROR
Filename fehlt, oder kann vom DOS nicht erkannt werden.
Typischer Fehler: Ein Doppelpunkt (:) fehlt.

39 : SYNTAX ERROR
Befehl wird nicht erkannt.

50 : RECORD NOT PRESENT
Die Fehlermeldung wird ausgegeben, wenn versucht wird, nach den letzten aufgezeichneten Daten mit INPUT# oder GET# weiter zu lesen.

51 : OVERFLOW IN RECORD
Die Anzahl der Zeichen (inklusive CR) ist größer als die vorgegebene Recordlänge

52 : FILE TOO LARGE
Recordnummer (in einem relativen File) ist zu groß; das Fassungsvermögen der Diskette wird überschritten.

60 : WRITE FILE OPEN
Ein Schreibfile, das nicht geschlossen wurde, soll geöffnet werden.

61 : FILE NOT OPEN
Es soll ein File angesprochen werden, das nicht geöffnet wurde.

62 : FILE NOT FOUND
File existiert nicht.

63 : FILE EXISTS
Ein File soll einen Namen erhalten, der bereits auf der Diskette existiert.

64 : FILE TYPE MISMATCH
File-Typ stimmt nicht mit Directory-Eintrag überein

65 : NO BLOCK
Diese Fehlermeldung steht im Zusammenhang mit dem B-A-Befehl. Der Block, der als belegt gekennzeichnet werden soll, ist bereits belegt. Die Parameter die ausgegeben werden, geben Spur und Sektor des nächsten frei verfügbaren Blocks an. Werden zwei Nullen ausgegeben, so sind alle Blocks mit höheren Nummern belegt.

66 : ILLEGAL TRACK AND SECTOR
Das DOS findet nicht den Zeiger zum nächsten Block

67 : ILLEGAL T OR S
Ungültige Spur- oder Sektornummer

70 : NO CHANNEL AVAILABLE
Ein direkt angesprochener Kanal ist bereits belegt oder (falls kein bestimmter Kanal angesprochen wurde) kein Kanal ist mehr frei.

71 : DIR ERROR
Die BAM kann nicht gelesen werden, was eventuell seinen Grund darin haben kann, daß sie im DOS-Speicher überschrieben wurde. Ein Reinitialisieren der Diskette kann diesen Fehler in manchen Fällen beheben.

72 : DISK FULL
Alle 664 Blocks der Diskette sind belegt, oder die Directory hat die maximale Anzahl von 144 Einträgen.

73 : CBM DOS V2.6 V170
Diese Meldung erscheint direkt nach dem Einschalten bzw. dann, wenn sie mit der VC-1541 auf eine Diskette schreiben wollen, die mit einer anderen DOS-Version formatiert wurde. Auf COMMODORE-Rechnern existieren vier verschiedene DOS-Versionen: DOS 1.0 : CBM 2040/3040
DOS 2.0 : CBM 4040
DOS 2.5 : CBM 8050
DOS 2.6 : CBM 1541
Voll kompatibel sind die Versionen 2.0 und 2.6. Mit DOS 1.0 und 2.0 bzw. 1.0 und 2.6 formatierte Disketten können wechselseitig gelesen, aber nicht beschrieben werden. Bei einem Schreibversuch erscheint die obige Fehlermeldung. Die DOS-Version 2.5 ist mit allen anderen Versionen weder schreib- noch lesekompatibel.

74 : DRIVE NOT READY
Es wurde versucht die Floppy anzusprechen, ohne eine Diskette in das Laufwerk einzulegen.

DER "JOKER"

Jokerzeichen werden benutzt, um bei einigen Floppy-Befehlen den Filenamen in verkürzter Form einzugeben. Man gibt nur den Teil des Namens ein, in dem er sich von den anderen Filenamen auf der Diskette unterscheidet. Der Rest wird durch einen Stern (*) oder ein Fragezeichen (?) ersetzt.
Der Stern wird benutzt, um nichtsignifikante Zeichen a m E n d e des Filenamens zu ersetzen.

BEISPIEL : DAT* kann für folgende Filenamen stehen :

DATEI
DATUM
DATENFILE

d.h., für alle Filenamen, die mit den Buchstabe DAT beginnen

Das Fragezeichen wird benutzt, um e i n Z e i c h e n an einer bestimmten Stelle zu ersetzen.

BEISPIEL : ????FILE kann stehen für

TESTFILE
DIAGFILE
DATAFILE

aber n i c h t für

ABFILE oder FILEPROG

Kombinationen von Stern und Fragezeichen müssen mit Vorsicht gehandhabt werden. So ist z.B. die Kombination

*ABC???? identisch mit *

Der Grund dafür ist, daß alle Stellen hinter dem Stern als nicht signifikant angesehen werden. Ein Stern kann aber für jeden beliebigen Filenamen stehen, so daß z.B. bei Verwendung dieser Jokerkombination in Verbindung mit dem SCRATCH-Befehl alle auf der Diskette befindlichen Files gelöscht würden.
Eine sinnvolle Kombination von Stern und Fragezeichen wäre z.B.:

A?????DAT* , die stehen kann für

ADRESSDATEN
ALTERSDATEI
ABCDE DATEIEN (ein Leerzeichen ist auch ein Zeichen)

Bei den einzelnen Floppy-Befehlen wirkt sich das Arbeiten mit den Joker-Symbolen verschieden aus. Der SCRATCH-Befehl löscht alle Namen, die durch die eingegebe Joker-Kombination abgedeckt werden. Bei Ausführung des LOAD-Befehls wird das erste File mit "passendem" Namen geladen. Dasselbe gilt für den OPEN-Befehl; hier kann jedoch nur ein File mit bereits existierendem Namen eröffnet werden. Die Befehle RENAME, SAVE oder COPY dürfen prinzipiell nicht in Verbindung mit Jokersymbolen verwendet werden.

KAPITEL 10 : Ändern der Geräteadresse

Da an den C-64 bis zu 5 Floppy Disk's anzuschließen sind, müßen die Geräte über verschiedene "Namen" angesprochen werden. Der "Name" einer Floppy ist die Gerätenummer. Die Gerätenummer kann Software- oder Hardwaremäßig geändert werden.

Softwarelösung:
PRINT# lfn,"M-W";CHR$(119)+CHR$(0)+CHR$(2)+CHR(nr+32)+CHR$(nr+64)
nr : neue Gerätenummer (9 - 13)

Beispiel:
PRINT#15,"M-W";CHR$(119)+CHR$(0)+CHR$(2)+CHR$(9+32)+CHR$(9+64)
Die neue Gerätenummer ist 9.

Nachteil der Softwarelösung: Dieser Befehl muß nach jedem Einschalten der Floppy wiederholt werden!

Hardwarelösung:

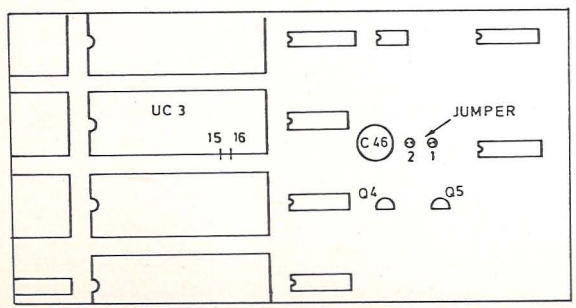
Die Adresse kann auch hardwaremäßig so umgestellt werden, daß sich das Gerät gleich beim Einschalten mit einer anderen Adresse meldet. (Ein nachträgliches Umstellen wie oben beschrieben bleibt trotzdem möglich)

Die Adresse des Gerätes wird durch die Beschaltung von Pin 15 und 16 des Port-Bausteines 6526 bestimmt, der auf der Leiterplatte in einem Sockel mit der Bezeichnung UC 3 steckt. Diese beiden Pins liegen normalerweise über je eine Lötbrücke (Jumper) an Masse. Durch Unterbrechen einer oder beider Verbindungen können die Adressen 9, 10 oder 11 eingestellt werden (siehe Tabelle).

Die Jumper liegen je nach Leiterplattentyp neben dem Kondensator C 46 oder zwischen den beiden Transistoren Q 4 und Q 5 (siehe Abbildung). Auf einer seltener vertretenen Leiterplatte steckt der Port-Baustein 6526 auf Position UAB 1. In diesem Fall liegen die beiden Jumper direkt vor den entsprechenden Pins 15 bzw. 16.

Die Jumper werden fachgerecht mit einem scharfen Trennmesser durchtrennt. Behelfsmäßig können auch die entsprechenden Beinchen aus dem Sockel gezogen und zur Seite etwas abgebogen werden.

Vorsicht: Diese Arbeiten dürfen nur von Fachkundigen vorgenommen werden. Für zerstörte Leiterplatten oder abgebrochene Pins wird keine Garantie geleistet.



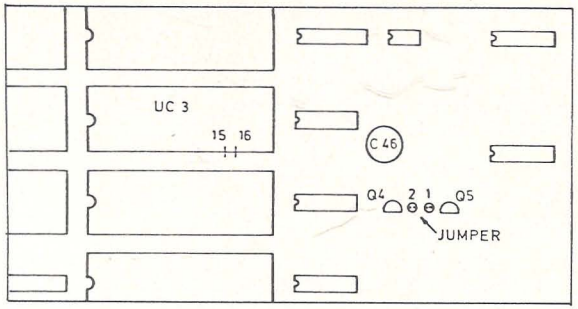


Bild : Lage der Jumper auf der Leiterplatte

Herausgebogener Pin UC 3	Durchgetrennte Lötbrücke	Neue Geräte-Adresse
15	1	9
16	2	10
15 und 16	1 und 2	11

Tabelle : Einstellen der Adressen 9-11

A N H A N G

Folgende Programme sind auf der beigelegten TEST/DEMO-Diskette enthalten. Das "Listing" dieser Programme ist auf den nächsten Seiten abgedruckt.

```
0 "TEST/DEMO       " 2A 2A
13   "HOW TO USE"          PRG
5    "HOW PART TWO"        PRG
4    "VIC-20 WEDGE"        PRG
1    "C-64 WEDGE"          PRG
4    "DOS 5.1"             PRG
11   "COPY/ALL"            PRG
9    "PRINTER TEST"        PRG
4    "DISK ADDR CHANGE"    PRG
4    "DIR"                 PRG
6    "VIEW BAM"            PRG
4    "CHECK DISK"          PRG
14   "DISPLAY T&S"         PRG
9    "PERFORMANCE TEST"    PRG
5    "SEQUENTIAL FILE"     PRG
13   "RANDOM FIAL"         PRG
```

HOW TO USE und HOW PART TWO
Eine Beschreibung der Programme auf der TEST/DEMO-Diskette.

VIC-20 WEDGE
Dieses Programm für den VC-20 erlaubt "verkürzte" Floppy-Befehle.

C-64 WEDGE
Ein Pogramm für den C-64, es läd ein "Utility-Programm" nach.

DOS 5.1
"Utility-Programm" für den C-64, es verkürzt die Floppy-Befehle.

COPY/ALL
Ein Kopierprogramm, daß alle Programme und Dateien von einer Floppy Disk auf eine andere überträgt.

PRINTER TEST
Der Zeichensatz eines CBM-Druckers wird ausgedruckt.

DISK ADDR CHANGE
Mit Hilfe dieses Programms kann man die Gerätenummer softwaremäßig umstellen.

DIR
Vereinfacht einige Floppyoperationen, wie z.B. das Anzeigen der Directory.

VIEW BAM
Liest die BAM von der Diskette und zeigt sie auf dem Bildschirm an. Hier bei werden die Sektornummern vertikal und die Spurnummern horizontal gezählt. Die leeren Blocks erscheinen hell und die belegten Blocks dunkel.

DISPLAY T&S
Gibt den Inhalt jedes gewünschten Blocks (einschließlich BAM und Directory) auf den Bildschirm oder den Drucker aus. Die Daten werden im ASCII-Code oder als Hexadezimalzahlen dargestellt.

CHECK DISK
Führt den Befehl VALIDATE aus und gibt an, welche Blocks nicht gelesen werden können. Erlaubt festustellen, inwieweit eine Diskette zerstört ist.

PERFORMANCE TEST
Allgemeiner Floppy-Funktionstest.

SEQUENTIAL FILE
Programmbeispiel, an dem das Schreiben und Lesen eines sequentiellen Files gezeigt wird.

RANDOM FILE
Programmbeispiel, an dem das Schreiben und Lesen von Files mit Direkt-Zugriffs-Befehlen (BLOCK und USER-Befehle) gezeigt wird.

```
10 PRINT"■"TAB(6)"■VIC WEDGE
20 PRINT"■   BY DAVID A. HOOK
30 PRINT"■D           DISK STATUS
40 PRINT"@          OR COMMANDS
50 PRINT"■D>$0          DIRECTORY
60 PRINT"@$0
70 PRINT"■/FILENAME          LOAD
80 SYS(PEEK(43)+256*PEEK(44)+215)
90 NEW

READY.
```

```
10 IFA=0THENA=1:LOAD"DOS 5.1",8,1
20 IFA=1THENSYS12*4096+12*256
30 NEW

READY.
```



```
100 PRINT"■ DI■K.COPY.ALL      JIM BUTTERFIELD"
110 DIM L2(232),L1%(232),N$(232),T%(232),T$(4)
120 DATA XXX,SEQ,PRG,USR,REL
130 FORJ=0TO4:READT$(J):NEXTJ
132 DATA10,32,67,72,77,82,87,98,117,147
134 SI=2348:SB=PEEK(44):MB=SB*256+SI
136 FORI=0TO9:READA:POKEMB+A,SB+9:NEXT
140 INPUT"FROM UNIT  8■■■■";F
150 GOSUB800
160 F$=D$
170 INPUT"TO UNIT  9■■■■";T
180 GOSUB800
190 T$=D$
200 IFF=T ANDF$=T$THENRUN
210 GOSUB860
220 CLOSE1:CLOSE15:OPEN 15,F,15:PRINT#15,"I"+F$
230 GOSUB830:IF E THEN STOP:GOTO220
240 INPUT"PATTERN  *■■■■";P$
250 FORJ=1TOLEN(P$):IFMID$(P$,J,1)<>"*"THENNEXTJ
260 P1=J-1:IFP1>0THENP$=LEFT$(P$,P1)
270 PRINT"HOLD DOWN 'Y' OR 'N' KEY TO SELECT"
280 PRINT"PROGRAMS TO BE COPIED..."
290 OPEN 1,F,3,"$"+F$
300 GOSUB830:IFETHENSTOP:GOTO220
310 GET#1,A$:A=ASC(A$+" ")
320 IFA=1ORA=65THENL1=253:GOTO350
330 IFA=67THENL1=761:GOTO350
340 CLOSE1:PRINT"?????":STOP
350 FORJ=1TOL1:GET#1,X$:NEXTJ
360 N=0:N1=0:R=255:Z=89
370 N$="":GOSUB1020:T9=Y-128:GOSUB1000
380 J1=1:Z$=CHR$(160):FORJ=1TO16:GET#1,X$:N$=N$+X$
390 IFX$<>Z$THENJ1=J
400 NEXTJ
410 GOSUB990:L1%=Y
420 GOSUB980:GOSUB980
430 L2=X+256*Y
440 IFT9<1ORT9>4GOTO540
450 IFP1>0THENIFP$<>LEFT$(N$,P1)GOTO540
460 PRINTN$;" ";T$(T9)
470 P=PEEK(151)ANDR
480 GETZ$:IFZ$=""ANDP<255GOTO520
490 IFZ$="Y"ORZ$="N"THENZ=ASC(Z$):R=255:GOTO520
500 IFZ$=CHR$(13)THENR=0:GOTO520
510 GOTO480
520 IFZ<80THENPRINT"■                              ■":GOTO540
530 N=N+1:L2(N)=L2:N$(N)=LEFT$(N$,J1):T%(N)=T9:L1%(N)=L1%
540 IFST>0GOTO570
550 N1=N1+1:IFN1=8THENN1=0:GOTO370
560 GOSUB1000:GOTO370
570 CLOSE1:CLOSE15:PRINT" * * * * *"
580 FORJ=1TON
590 L2=L2(J):T%=T%(J):IFL>L2GOTO640
600 PRINT"*** OUTPUT DISK FULL"
610 INPUT"DO YOU HAVE A NEW ONE";Z$
620 IFASC(Z$)<>89THENEND
630 GOSUB860:GOTO590
640 OPEN14,F,15:OPEN15,T,15
650 PRINTN$(J);LEFT$("                ",17-LEN(N$(J)));
660 OPEN3,F,3,F$+":"+N$(J)+","+T$(T%)
670 INPUT#14,E,E$,E1,E2:GOSUB840:IFETHENPRINT"** ";E$;E:GOTO750
```

```
 680 IFT%=4THENOPEN4,T,4,T$+":"+N$(J)+",L,"+CHR$(L1%(J)):GOTO700
 690 OPEN4,T,4,T$+":"+N$(J)+","+T$(T%)+",W"
 700 L=L-L2:GOSUB830:IFETHENPRINT"*** ";E$;E:GOTO750
 710 IFT%=4THENSYS(MB+43):GOTO730
 720 SYS(MB+3)
 730 N$(J)="":GOSUB830:IFETHENPRINT"**** ";E$;E:GOTO750
 740 PRINT"J"
 750 CLOSE4:CLOSE3:CLOSE15:CLOSE14
 760 NEXTJ
 770 X=FRE(0):PRINT"ANOTHER INPUT DISK READY":INPUT"   N■■■■";Z$
 780 IFASC(Z$)=89GOTO220
 790 END
 800 INPUT"DRIVE  0■■■■";D
 810 IFD*D<>DGOTO800
 820 D$=CHR$(D+48):RETURN
 830 INPUT#15,E,E$,E1,E2
 840 IFE=0THENE=(ST AND 191):E$="*ST*"
 850 RETURN
 860 OPEN15,T,15:PRINT"WANT TO NEW THE OUTPUT DISK":INPUT"    N■■■■";Z$
 870 IFASC(Z$)<>89GOTO910
 880 INPUT"DISK NAME,ID";X$,Y$
 890 PRINT#15,"N"+T$+":"+X$+","+Y$
 900 GOSUB830:IFETHENSTOP:GOTO860
 910 PRINT#15,"I"+T$:OPEN1,T,0,"$"+T$+":!#$%&"
 920 GOSUB830:IFETHENSTOP:GOTO860
 930 GOSUB980:GOSUB980
 940 Q=Q+1:GET#1,X$:IFX$<>""GOTO940
 950 GOSUB980
 960 L=X+Y*256:PRINT"(";L;"BLOCKS FREE )"
 970 CLOSE1:CLOSE15:RETURN
 980 GET#1,X$
 990 GET#1,X$
1000 GET#1,X$
1010 X=LEN(X$):IFXTHENX=ASC(X$)
1020 GET#1,X$:Y=LEN(X$):IFYTHENY=ASC(X$)
1030 RETURN

READY.
```



```
100 REM PRINTER TEST FOR VIC AND 64
110 NL$=CHR$(10):CR$=CHR$(13)
120 BS$=CHR$(8):SO$=CHR$(14):SI$=CHR$(15)
130 PO$=CHR$(16):ESC$=CHR$(27):SUB$=CHR$(26)
140 CD$=CHR$(17):CU$=CHR$(145)
150 RVS$=CHR$(18):OFF$=CHR$(146)
160 :
170 OPEN4,4:PRINT#4:PRINT#4
180 PRINT#4,SO$"              PRINTER TEST"
190 :
200 PRINT#4:PRINT#4
210 PRINT#4,SO$"1. CHECK GRAPHIC (CURSOR UP) MODE"SI$
220 GOSUB840:CLOSE4
230 :
240 OPEN4,4,7 :PRINT#4
250 PRINT#4,SO$"2. CHECK BUSINESS(CURSOR DOWN) MODE"SI$
260 GOSUB840:CLOSE4
270 :
280 OPEN4,4:PRINT#4:PRINT#4,SO$"3. CHR$(10) TEST"SI$:PRINT#4
290 FORI=48TO52:PRINT#4,PO$"25"CHR$(I);NL$:NEXT
300 :
310 PRINT#4:PRINT#4,SO$"4. CHR$(13) TEST"SI$:PRINT#4
320 FORI=53TO57:PRINT#4,PO$"25"CHR$(I);CR$:NEXT
330 :
340 PRINT#4:PRINT#4,SO$"5. CHR$(8) TEST"SI$:PRINT#4
350 FORJ=1TO2
360 A$="":FORI=128TO255:A$=A$+CHR$(I):NEXT
370 PRINT#4,BS$A$:PRINT#4,SI$:NEXTJ
380 FORJ=1TO2
390 B$="":FORI=255TO128STEP-1:B$=B$+CHR$(I):NEXT
400 PRINT#4,BS$B$:PRINT#4,SI$:NEXTJ
410 :
420 PRINT#4:PRINT#4,SO$"6. CHR$(14) & CHR$(15) TEST"SI$:PRINT#4
430 FORJ=2TO4STEP2
440 A$="":FORI=J*16TOJ*16+31:A$=A$+CHR$(I):NEXT
450 PRINT#4,SO$A$CR$SI$A$:NEXT
460 :
470 PRINT#4:PRINT#4,SO$"7. CHR$(16) TEST"SI$:PRINT#4
480 FOR J=0TO7:PRINT#4,PO$CHR$(48+J)CHR$(0)"COMMODORE":NEXT
490 :
500 PRINT#4:PRINT#4,SO$"8. CHR$(26) TEST"SI$:PRINT#4
510 FOR J=0TO3:PRINT#4,BS$SUB$CHR$(10+10*J↑2)CHR$(255):NEXT
520 :
530 PRINT#4:PRINT#4,SO$"9. CHR$(27) TEST"SI$:PRINT#4
540 SP$="           "
550 FOR I=0TO360STEP10
560 I$=RIGHT$(SP$+STR$(I),4)
570 YO=220+120*SIN(I*π/180)
580 YH=INT(YO/256):YL=YO-YH*256
590 PRINT#4,I$ESC$PO$CHR$(YH)CHR$(YL)"*"
600 NEXT:CLOSE4
610 :
620 OPEN4,4
630 PRINT#4:PRINT#4,SO$"10. CHR$(17) TEST"SI$:PRINT#4
640 FORJ=10TO13
650 A$="":FORI=J*16TOJ*16+15:A$=A$+CHR$(I):NEXT
660 PRINT#4,A$"    "CD$A$
670 NEXT
680 :
690 PRINT#4:PRINT#4,SO$"11. CHR$(145) TEST"SI$:PRINT#4:CLOSE4
```

```
700 OPEN4,4,7:FORJ=10TO13
710 A$="":FORI=J*16TOJ*16+15:A$=A$+CHR$(I):NEXT
720 PRINT#4,A$"    "CU$A$
730 NEXT:CLOSE4
740 :
750 OPEN4,4
760 PRINT#4:PRINT#4,SO$"12. CHR$(18) & CHR$(146) TEST "SI$:PRINT#4
770 A$=" COMMODORE "
780 FORI=0TO3:PRINT#4,RVS$A$OFF$"  "A$"  "RVS$A$OFF$"  "A$"  "RVS$A$
790 NEXT
800 :
810 PRINT#4:PRINT#4:CMD4,SO$"PROGRAM LIST"SI$:LIST
820 END
830 :
840 PRINT#4
850 FORI=32TO47
860 FORJ=1TO8:PRINT#4,CHR$(I);:NEXT:PRINT#4,"  ";
870 FORJ=1TO8:PRINT#4,CHR$(I+16);:NEXT:PRINT#4,"  ";
880 FORJ=1TO8:PRINT#4,CHR$(I+32);:NEXT:PRINT#4,"  ";
890 FORJ=1TO8:PRINT#4,CHR$(I+48);:NEXT:PRINT#4,"  ";
900 FORJ=1TO8:PRINT#4,CHR$(I+128);:NEXT:PRINT#4,"  ";
910 FORJ=1TO8:PRINT#4,CHR$(I+144);:NEXT:PRINT#4,"  ";
920 FORJ=1TO8:PRINT#4,CHR$(I+160);:NEXT:PRINT#4,"  ";
930 FORJ=1TO8:PRINT#4,CHR$(I+176);:NEXT:PRINT#4,"  ";
940 NEXT
950 PRINT#4:RETURN

READY.
```



```
100 POKE59468,12
110 PRINT"■■■■■■■DRIVE ADDRESS CHANGE PROGRAM"
111 PRINT"■TURN OFF ALL DRIVES NOW"
112 PRINT"EXCEPT THE ONE TO BE CHANGED."
120 PRINT"■OLD DEVICE ADDRESS  8■■■";
130 INPUT OD: IF OD<8 OR OD>15 GOTO 120
140 PRINT"■NEW DEVICE ADDRESS  9■■■";
150 INPUT ND: IF ND<8 OR ND>15 GOTO 140
160 GOSUB 300: REM FIND DRIVE TYPE
170 GOSUB 600: REM CHANGE ADDRESS
180 PRINT"■THE SELECTED DRIVE HAS BEEN CHANGED..."
185 PRINT"NOW TURN ON THE OTHER DRIVE(S)"
190 END
300 REM: IDENTIFY DRIVE TYPE
310 CLOSE15:OPEN15,OD,15
320 PRINT#15,"M-R"CHR$(255)CHR$(255):GET#15,C$:C=ASC(C$+CHR$(0))
330 IF ST THEN 1000
340 IF C=254 THEN MT=119: REM: 2031 V2.6
350 IF C=226 THEN MT=50: REM: 2040 V1.2
360 IF C=213 THEN MT=12: REM: 4040 V2.1
370 IF C=242 THEN MT=12: REM: 8050 V2.5
380 IF C=198 THEN 400
390 RETURN
400 PRINT#15,"M-R"CHR$(234)CHR$(16):GET#15,ZB$:ZB=ASC(ZB$+CHR$(0))
410 IF ZB=0 THEN MT=12: REM: 4040 V2.7
420 IF ZB=1 THEN 440
430 IF ST THEN 1000
440 PRINT#15,"M-R"CHR$(172)CHR$(16):GET#15,ZC$:ZC=ASC(ZC$+CHR$(0))
450 IF ZC=1 THEN MT=12: REM: 8050 V2.7
460 IF ZC=2 THEN MT=12: REM: 8250 V2.7
470 RETURN
600 CLOSE15: OPEN15,OD,15
610 PRINT#15,"M-W"CHR$(MT)CHR$(0)CHR$(2)CHR$(ND+32)CHR$(ND+64)
630 RETURN
1000 PRINT" DEVICE  ERROR "
1010 END

READY.
```

```
4 OPEN2,8,15
5 PRINT"■":GOTO 10000
10 OPEN1,8,0,"$0"
20 GET#1,A$,B$
30 GET#1,A$,B$
40 GET#1,A$,B$
50 C=0
60 IF A$<>"" THEN C=ASC(A$)
70 IF B$<>"" THEN C=C+ASC(B$)*256
80 PRINT"■"MID$(STR$(C),2);TAB(3);"■";
90 GET#1,B$:IF ST<>0 THEN 1000
100 IF B$<>CHR$(34) THEN 90
110 GET#1,B$:IF B$<>CHR$(34)THEN PRINTB$;:GOTO110
120 GET#1,B$:IF B$=CHR$(32) THEN 120
130 PRINT TAB(18);:C$=""
140 C$=C$+B$:GET#1,B$:IF B$<>""  THEN 140
150 PRINT"■"LEFT$(C$,3)
160 GET T$:IF T$<>"" THEN GOSUB 2000
170 IF ST=0 THEN 30
1000 PRINT" BLOCKS FREE"
1010 CLOSE1:GOTO 10000
2000 IF T$="Q" THEN CLOSE1:END
2010 GET T$:IF T$="" THEN 2000
2020 RETURN
4000 REM DISK COMMAND
4010 C$="":PRINT">";
4011 GETB$:IFB$="" THEN4011
4012 PRINTB$;:IF B$=CHR$(13) THEN 4020
4013 C$=C$+B$:GOTO 4011
4020 PRINT#2,C$
5000 PRINT"■";
5010 GET#2,A$:PRINTA$;:IF A$<>CHR$(13)GOTO5010
5020 PRINT"■"
10000 PRINT "D-DIRECTORY"
10010 PRINT ">-DISK COMMAND"
10020 PRINT "Q-QUIT PROGRAM"
10030 PRINT "S-DISK STATUS "
10100 GETA$:IFA$=""THEN10100
10200 IF A$="D" THEN 10
10300 IF A$="." OR A$=">" OR A$="$" THEN 4000
10310 IF A$="Q" THEN END
10320 IF A$="S" THEN 5000
10999 GOTO 10100

READY.
```



```
100 REM ********************************
101 REM *  VIEW BAM FOR VIC & 64 DISK  *
102 REM ********************************
105 OPEN15,8,15
110 PRINT#15,"I0":NU$="N/A N/A N/A N/A N/A":Z4=1
120 OPEN2,8,2,"#"
130 Y$="■■■■■■■■■■■■■■■■■■■■■■■■■"
140 X$="■■■■■■■■■■■■■■■■■■■■■■■■■"
150 DEF FNS(Z) = 2↑(S-INT(S/8)*8) AND (SB(INT(S/8)))
160 PRINT#15,"U1:";2;0;18;0
170 PRINT#15,"B-P";2;1
180 PRINT"■";
190 Y=22:X=1:GOSUB430
200 FORI=0TO20:PRINT:PRINT"■I"RIGHT$(STR$(I)+" ",3);:NEXT
210 GET#2,A$
220 GET#2,A$
230 GET#2,A$
240 TS=0
250 FORT=1TO17:GOSUB450
260 Y=22:X=T+4:GOSUB430:GOSUB540:NEXT
270 FORI=1TO2000:NEXT:PRINT"■"
280 Y=22:X=1:GOSUB430
290 FORI=0TO20:PRINT:PRINT"■I"RIGHT$(STR$(I)+" ",3);:NEXT
300 FORT=18TO35
310 GOSUB450
320 Y=22:X=T-13:GOSUB430:GOSUB540:NEXT
330 FORI=1TO1000:NEXT
340 PRINT"■■■■■■"
350 PRINT#15,"B-P";2;144
360 N$="":FORI=1TO20:GET#2,A$:N$=N$+A$:NEXT
370 PRINT" "N$" "TS-17;"BLOCKS FREE"
380 FORI=1TO4000:NEXT
390 PRINT"■"
400 INPUT"■■■■■ANOTHER DISKETTE N■■■■";A$
410 IFA$="Y"THENRUN
420 IFA$<>"Y"THENEND
430 PRINTLEFT$(Y$,Y)LEFT$(X$,X)"■■";
440 RETURN
450 GET#2,SC$:SC=ASC(RIGHT$(CHR$(0)+SC$,1))
460 TS=TS+SC
470 GET#2,A$:IFA$=""THENA$=CHR$(0)
480 SB(0)=ASC(A$)
490 GET#2,A$:IFA$=""THENA$=CHR$(0)
500 SB(1)=ASC(A$)
510 GET#2,A$:IFA$=""THENA$=CHR$(0)
520 SB(2)=ASC(A$)
530 RETURN
540 PRINT"■■"RIGHT$(STR$(T),1);"■■I■";
550 REM  PRINTT"    "SC"  "SB(0)" "SB(1)" "SB(2)=CHR$(0)
560 IFT>24ANDS=18THEN:PRINTMID$(NU$,Z4,1);:GOTO660
570 FORS=0TO20
580 IFT<18THEN620
590 IFT>30ANDS=17THEN:PRINTMID$(NU$,Z4,1);:GOTO660
600 IFT>24ANDS=18THEN:PRINTMID$(NU$,Z4,1);:GOTO660
610 IFT>24ANDS=19THENPRINTMID$(NU$,Z4,1);:GOTO660
620 IFT>17ANDS=20THENPRINTMID$(NU$,Z4,1);:Z4=Z4+1:GOTO660
630 PRINT"■";
640 IF FNS(S)=0 THEN PRINT"+";:GOTO660
650 PRINT"■+";:REMRIGHT$(STR$(S),1);Z4,1);:GOTO72
660 PRINT"■■";
670 NEXT
680 RETURN

READY.
```

```
1 REM CHECK DISK -- VER 1.4
2 DN=8:REM FLOPPY DEVICE NUMBER
5 DIMT(100):DIMS(100):REM BAD TRACK, SECTOR ARRAY
9 PRINT"■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■"
10 PRINT" CHECK DISK PROGRAM"
12 PRINT"■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■"
20 D$="0"
30 OPEN15,DN,15
35 PRINT#15,"V"D$
45 N%=RND(TI)*255
50 A$="":FORI=1TO255:A$=A$+CHR$(255AND(I+N%)):NEXT
60 GOSUB900
70 OPEN2,DN,2,"#"
80 PRINT:PRINT#2,A$;
85 T=1:S=0
90 PRINT#15,"B-A:"D$;T;S
100 INPUT#15,EN,EM$,ET,ES
110 IFEN=0THEN130
115 IFET=0THEN200:REM END
120 PRINT#15,"B-A:"D$;ET;ES:T=ET:S=ES
130 PRINT#15,"U2:2,"D$;T;S
134 NB=NB+1:PRINT" CHECKED  BLOCKS"NB
135 PRINT" TRACK    ■■■■■"T;" SECTOR    ■■■■■"S":■■"
140 INPUT#15,EN,EM$,ES,ET
150 IF EN=0THEN85
160 T(J)=T:S(J)=S:J=J+1
165 PRINT"■■BAD BLOCK:■■",T;S""
170 GOTO85
200 PRINT#15,"I"D$
210 GOSUB900
212 CLOSE2
215 IFJ=0THENPRINT"■■■■■■NO BAD BLOCKS!":END
217 OPEN2,DN,2,"#"
218 PRINT"■■BAD BLOCKS","TRACK","SECTOR"
220 FORI=0TOJ-1
230 PRINT#15,"B-A:";D$,T(I);S(I)
240 PRINT#,T(I),S(I)
250 NEXT
260 PRINT"■"J"BAD BLOCKS HAVE BEEN ALLOCATED"
270 CLOSE2:END
900 INPUT#15,EN,EM$,ET,ES
910 IF EN=0 THEN RETURN
920 PRINT"■■ERROR #"EN,EM$;ET;ES""
930 PRINT#15,"I"D$

READY.
```



```
100 REM*******************************
110 REM* DISPLAY ANY TRACK $ SECTOR *
120 REM* ON THE DISK TO THE SCREEN  *
130 REM* OR THE PRINTER             *
140 REM*******************************
150 PRINT"■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■"
160 PRINT"DISPLAY BLOCK CONTENTS"
165 PRINT"■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■":
170 REM*******************************
180 REM* SET PROGRAM CONSTANT       *
190 REM*******************************
200 SP$=" ":NL$=CHR$(0):HX$="0123456789ABCDEF"
210 FS$="":FORI=64 TO 95:FS$=FS$+"■"+CHR$(I)+"■":NEXT I
220 SS$="  ":FOR I=192 TO 223:SS$=SS$+"■"+CHR$(I)+"■":NEXT I
240 DIM A$(15),NB(2)
251 D$="0"
253 PRINT"        ■S■CREEN■■■■■■■■■■OR ■■■P■RINTER"
254 GETJJ$:IF JJ$="" THEN254
255 IF JJ$="S"THENPRINT"         ■■SCREEN■"
256 IF JJ$="P"THENPRINT"         ■■PRINTER■"
260 OPEN15,8,15,"I"+D$:GOSUB 650
265 OPEN4,4
270 OPEN 2,8,2,"#":GOSUB 650
280 REM*******************************
290 REM* LOAD TRACK AND SECTOR       *
300 REM* INTO DISK BUFFER            *
310 REM*******************************
320 INPUT"■■■TRACK, SECTOR";T,S
330 IF T=0 OR T>35 THEN PRINT#15,"I"D$:CLOSE2:CLOSE4:CLOSE15:PRINT"END":END
340 IF JJ$="S" THEN PRINT"■■■TRACK"T" SECTOR"S"■"
341 IF JJ$="P" THEN PRINT#4:PRINT#4,"TRACK"T" SECTOR"S:PRINT#4
350 PRINT#15,"U1:2,"D$;T;S:GOSUB650
360 REM*******************************
370 REM* READ BYTE 0 OF DISK BUFFER *
390 REM*******************************
400 PRINT#15,"B-P:2,1"
410 PRINT#15,"M-R"CHR$(0)CHR$(5)
420 GET#15,A$(0):IFA$(0)=""THENA$(0)=NL$
428 IF JJ$="S"THEN430
430 IF JJ$="P"THEN460
431 REM*******************************
432 REM* READ & CRT DISPLAY          *
433 REM* REST OF THE DISK BUFFER     *
434 REM*******************************
436 K=1:NB(1)=ASC(A$(0))
438 FOR J=0 TO 63:IF J=32 THEN GOSUB 710:IF Z$="N"THEN J=80:GOTO 458
440 FOR I=K TO 3
442 GET#2,A$(I):IF A$(I)="" THEN A$(I)=NL$
444 IF K=1 AND I<2 THEN NB(2)=ASC(A$(I))
446 NEXT I:K=0
448 A$="":B$=":":N=J*4:GOSUB 790:A$=A$+":"
450 FOR I=0 TO 3:N=ASC(A$(I)):GOSUB 790
452 C$=A$(I):GOSUB 850:B$=B$+C$
454 NEXT I:IF JJ$="S" THEN PRINTA$B$
458 NEXT J:GOTO571
```

```
460 REM*********************************
462 REM* READ & PRINTER DISPLAY     *
464 REM*********************************
466 K=1:NB(1)=ASC(A$(0))
468 FOR J=0 TO 15
470 FOR I=K TO 15
472 GET#2,A$(I):IF A$(I)="" THEN A$(I)=NL$
474 IF K=1 AND I<2 THEN NB(2)=ASC(A$(I))
476 NEXT I:K=0
478 A$="":B$=":":N=J*16:GOSUB 790:A$=A$+":"
480 FOR I=0 TO 15:N=ASC(A$(I)):GOSUB 790:IF Z$="N"THEN J=40:GOTO 571
482 C$=A$(I):GOSUB 850:B$=B$+C$
484 NEXT I
486 IF JJ$="P" THEN PRINT#4,A$B$
488 NEXT J:GOTO571
571 REM*********************************
572 REM* NEXT TRACK AND SECTOR      *
573 REM*********************************
575 PRINT"NEXT TRACK AND SECTOR"NB(1)NB(2) "■"
580 PRINT"DO YOU WANT NEXT TRACK AND SECTOR"
590 GET Z$:IF Z$="" THEN590
600 IF Z$="Y" THEN T=NB(1):S=NB(2):GOTO330
610 IF Z$="N" THEN 320
620 GOTO 590
630 REM*********************************
640 REM* SUBROUTINES                *
650 REM*********************************
660 REM* ERROR ROUTINE              *
670 REM*********************************
680 INPUT#15,EN,EM$,ET,ES:IF EN=0 THEN RETURN
690 PRINT"■DISK ERROR■"EN,EM$,ET,ES
700 END
710 REM*********************************
720 REM* SCREEN CONTINUE MESSAGE    *
730 REM*********************************
740 PRINT"■▶▶▶▶CONTINUE(Y/N)"
750 GETZ$:IF Z$="" THEN 750
760 IF Z$="N" THEN RETURN
770 IF Z$<>"Y" THEN 750
780 PRINT"□TRACK" T " SECTOR"S "□":RETURN
790 REM*********************************
800 REM* DISK BYTE TO HEX PRINT     *
810 REM*********************************
820 A1=INT(N/16):A$=A$+MID$(HX$,A1+1,1)
830 A2=INT(N-16*A1):A$=A$+MID$(HX$,A2+1,1)
840 A$=A$+SP$:RETURN
850 REM*********************************
860 REM* DISK BYTE TO ASC DISPLAY   *
870 REM* CHARACTER                  *
880 REM*********************************
890 IF ASC(C$)<32 THEN C$=" ":RETURN
910 IF ASC(C$)<128 OR ASC(C$)>159 THEN RETURN
920 C$=MID$(SS$,3*(ASC(C$)-127),3):RETURN

READY.
```



```
1000 REM  PERFORMANCE TEST  2.0
1010 :
1020 REM  VIC-20 AND COMMODORE 64
1030 REM  SINGLE FLOPPY DISK DRIVE
1040 :
1050 OPEN 1,8,15:OPEN15,8,15
1060 LT=35
1070 LT$=STR$(LT)
1080 NT=30
1090 PRINT"□■_____"
1100 PRINT"   PERFORMANCE TEST"
1110 PRINT"_____"
1120 PRINT
1130 PRINT"  INSERT SCRATCH"
1140 PRINT
1150 PRINT"    DISKETTE IN DRIVE"
1160 PRINT
1170 PRINT"■  PRESS ■RETURN■"
1180 PRINT
1190 PRINT"          WHEN READY■"
1200 FOR I=0 TO 50:GET A$:NEXT
1210 GET A$:IF A$<>CHR$(13) THEN 1210
1220 :
1230 :
1240 TI$="000000"
1250 TT=18
1260 PRINT#1,"N0:TEST DISK,00"
1270 C1$="   DISK NEW COMMAND   "+CHR$(13)
1280 C2$="■ WAIT ABOUT 80 SECONDS"
1290 CC$=C1$+C2$:GOSUB 1840
1300 IF TI<NTTHEN1370
1310 PRINT"■SYSTEM IS"
1320 PRINT"■        NOT RESPONDING"
1330 PRINT" CORRECTLY TO COMMANDS"
1340 GOSUB 1880
1350 :
1360 :
1370 PRINT"■■DRIVE PASS"
1380 PRINT"       MECHANICAL TEST■"
1390 TT=21
1400 OPEN 2,8,2,"0:TEST FILE,S,W"
1410 CC$="OPEN WRITE FILE"        :GOSUB 1840
1420 CH=2:CC$="WRITE DATA"        :GOSUB 1930
1430 CC$="CLOSE "+CC$             :GOSUB 1840
1440 OPEN 2,8,2,"0:TEST FILE,S,R"
1450 CC$="OPEN READ FILE"         :GOSUB 1840
1460 CH=2:GOSUB 1990
1470 PRINT#1,"S0:TEST FILE"
1480 CC$="SCRATCH FILE■":TT=1     :GOSUB 1840
1490 :
1500 :
```

```
1510 TT=21
1520 OPEN 4,8,4,"#"
1530 NN%=(1+RND(TI)*254+NN%)AND255:PRINT#1,"B-P";4;NN%
1540 NN$="":FOR I=1 TO 255:NN$=NN$+CHR$(I):NEXT
1550 PRINT# 4,NN$;
1560 PRINT# 1,"U2:";4;0;LT;0
1570 CC$="WRITE TRACK"+LT$:GOSUB 1840
1580 PRINT#1,"U2:";4;0;1;0
1590 CC$="WRITE TRACK 1"          :GOSUB 1840
1600 PRINT#1,"U1:";4;0;LT;0
1610 CC$="READ TRACK"+LT$         :GOSUB 1840
1620 PRINT#1,"U1:";4;0;1;0
1630 CC$="READ TRACK 1"           :GOSUB 1840
1640 CLOSE 4
1650 :
1660 :
1670 PRINT"▯ UNIT HAS PASSED"
1680 PRINT"     PERFORMANCE TEST!"
1690 PRINT"▯ PULL DISKETTE FROM"
1700 PRINT"▯  DRIVE BEFORE TURNING"
1710 PRINT"   POWER OFF."
1720 END
1730 :
1740 :
1750 PRINT"   ▯CONTINUE (Y/N)?";
1760 FOR I=0 TO 50:GET A$:NEXT
1770 GET A$:IF A$="" THEN 1770
1780 PRINT A$"▯"
1790 IF A$="N" THEN END
1800 IF A$="Y" THEN RETURN
1810 GOTO 1760
1820 :
1830 :
1840 PRINT CC$
1850 INPUT# 1,EN,EM$,ET,ES
1860 PRINTTAB(12)""EN;EM$;ET;ES;""
1870 IF EN<2 THEN RETURN
1880 PRINT"▯ UNIT IS FAILING"
1890 PRINT"▯   PERFORMANCE TEST"
1900 TM$=TI$:GOSUB 1750:TI$=TM$:RETURN
1910 :
1920 :
1930 PRINT"WRITING DATA"
1940 FOR I=1000 TO 2000:PRINT#CH,I:NEXT
1950 GOSUB1850
1960 CLOSE CH:RETURN
1970 :
1980 :
1990 PRINT"READING DATA"
2000 GETA$
2010 FOR I=1000 TO 2000
2020 INPUT# CH,J
2030 IF J<>I THEN PRINT"▯READ ERROR:▩":GOSUB 1850
2040 NEXT
2050 GOSUB 1850
2060 CLOSE CH:RETURN

READY.
```


Rescue on Fractalus

World Series Baseball

Ice Palace