Hard-Disk 24 MB für CBM

Technische Unterlagen

Hübner & Worm

HÜBNER & WORM GmbH

1000 Berlin 61
Prinzessinnenstr. 20

Tel.: 614 50 56

INBETRIEBNAHME DER HARD - DISK 24 MB

1. Die Hard - Disk 24 MB darf **unter keinen Umständen** ohne lösen der Verriegelung inbetriebgenommen werden.

2. Um die Verriegelung zu lösen wird der Deckel der Hard - Disk abgenommen und der weiße Hebel am Plattenlaufwerk auf FREE gelegt.

3. Der Deckel wird wieder montiert, das Netz- und IEC - Bus Kabel angeschlossen. Nach dem Einschalten leuchtet das rote LED auf und erlischt nachdem die Platte betriebsbereit ist (ca. 1 min).

4. Die Hard - Disk ist hochempfindlich gegen Stoß. Sie muß unbedingt vor Erschütterungen geschützt werden.

5. Bei jedem Transport der Hard - Disk ist die Verriegelung der Platte vorzunehmen. Der weiße Hebel wird auf LOCK gestellt.

6. Vor öffnen des Gerätes Netzstecker ziehen.

7. **Bei unsachgemäßer Behandlung des Gerätes erlischt die Garantie.**

Inhaltsverzeichnis

1. Technische Daten der Hard - Disk 10 / 24 MB

Die Hard - Disk 10/24 MB ist ein speziell für die CBM-Rechner entwickelter schneller und zuverlässiger Massenspeicher. Ein intelligenter Controller koppelt das Laufwerk über den IEC-Bus mit dem Rechner.

Das acht-Zoll-Laufwerk hat eine Speicherkapazität von 11,7 MByte unformatiert (10,4 MByte formatiert) bzw. 24 MByte unformatiert (20 MByte formatiert). Die beiden Platten und die Schreib- Leseköpfe befinden sich in einer abgedichteten Kammer in einer hochreinen, staubfreien Atmosphäre. Daudurch wird eine hohe Zuverlässigkeit und die große Datendichte erreicht. Der Plattenantrieb erfolgt riemenlos über einen kollektorlosen, quarzgeregelten Gleichstrommotor. Der einfache mechanische Aufbau gewährleistet einen jahrelangen, wartungsfreien Betrieb.

Der Controller enthält einen 8-bit Mikroprozessor und 16 kByte Halbleiterspeicher. Er kann bis zu fünf Laufwerke bedienen. Mit einem hierarchischen Dateiensystem, wie es sonst nur bei Time-sharing-Anlagen vorzufinden ist, kann der große Speicherbereich flexibel und übersichtlich organisiert und verwaltet werden. Von jeder geöffneten Datei liegt immer der gerade aktuelle Sektor im Puffer des Controllers. Dadurch werden auch bei komplexen Programmen, mit mehreren offenen Dateien, die Plattenzugriffe auf ein Minimum reduziert. Dateien mit vertraulichen Daten oder Programmen können mit Passworten vor unberechtigtem Zugriff geschützt werden. Die 24 MB Hard - Disk ist im Gegensatz zur 10 MB Platte logisch in zwei Laufwerke unterteilt.

	10 MB	24 MB
Speicherkapazität		
unformatiert	11,7 MByte	23,4 MByte
formatiert	10,4 MByte	20,0 MByte
Plattendurchmesser	8 Zoll	8 Zoll
Anzahl der Platten	2	4
Anzahl der Plattenoberflächen	4	8
Festkopf (für Takt)	1	1
Spuren pro Oberfläche	244	244
Sektoren pro Spur	22	22
Bytes pro Sektor	512	512
Aufzeichnungsverfahren	MFM	
Aufzeichnungsdichte (innerste Spur)	6100 bits/inch	
Umdrehungszahl der Platten	2983 U/min, quarzgeregelt	
Lese- und Schreibgeschwindigkeit	4,77 Mbits/s	
Positionierung:	Schrittmotor mit Hochlaufsteuerung	
Organisation der Dateien		
Minimale Größe einer Datei	512 Bytes	
Maximale Größe einer Datei	10,4 MBytes	10 MBytes*
Maximale Anzahl von Dateien:		
ist nur durch den Speicherplatz begrenzt	etwa 19000	etwa 19000
Minimale Größe einer Directory	1024 Bytes	1024 Bytes
Maximale Größe einer Directory	10,4 MBytes	10,0 MBytes
Maximale Anzahl von Directories:		
ist nur durch den Speicherplatz begrenzt	etwa 10000	etwa 10000

* Die 24 MB Platte ist logisch in zwei Drives a ca.10 MB unterteilt.

2. Übersicht über die Befehle der Hard - Disk

Dateien

Der Speicherplatz der Hard-Disk ist in Dateien unterteilt. Eine Datei enthält ein Programm oder Daten. Der Speicherplatz wird in Einheiten von 512 Bytes vergeben. Dateien haben also einen Umfang von 512 Bytes, oder einem Vielfachen davon. Maximal kann eine Datei den gesamten Speicherplatz einnehmen.
Die Dateien werden über Namen angesprochen. Die Namen sind in einer oder mehreren Directories (Inhaltsverzeichnissen) eingetragen. In diesem Kapitel wird nur der Fall betrachtet, daß alle Dateien eines Laufwerks in der gleichen Directory eingetragen sind, und daß jede Datei nur über einen einzigen Namen angesprochen wird. Die Befehle wirken dadurch ähnlich wie die eines herkömmlichen Floppy-Disk-Betriebssystems. Ihre volle Leistungsfähigkeit wird erst im nächsten Kapitel bei der Beschreibung des hierarchischen Dateiensystems ersichtlich.

Der Directory - Befehl

Der Inhalt einer Directory wird durch die Befehle

```
load "$0",8
list
```

oder

```
directory                          beim CBM BASIC 4.0
```

auf dem Bildschirm des CBMs ausgegeben. Die Angabe "$0" bewirkt, daß die Directory von Laufwerk 0 gelesen wird. Diejenige von Laufwerk 1 kann mit

```
load "$1"
list
```

oder

```
directory d1                       beim BASIC 4.0
```

aufgelistet werden.
In diesem wie in allen folgenden Beispielen wird angenommen, daß die Hard - Disk 24 MB auf die Gerätenummer 8 eingestellt ist. Falls eine Floppy-Disk an den CBM angeschloßen ist, kann diese auf die Adresse 9 umprogrammiert werden.

Wenn viele Namen in einer Directory stehen, werden die Listen lang und unübersichtlich. Es ist deshalb oft von Nutzen, nur einen Teil der Namen aufzulisten. Zu diesem Zweck kann eine Suchzeichenfolge beim DIRECTORY-Befehl angegeben werden. Dann werden nur die Namen ausgegeben, die mit der Zeichenfolge übereinstimmen.

Beispiel:

```
load "$1:SCHACH",8
```

sucht in der Directory von Laufwerk 1 nach dem Namen "SCHACH". Ist er vorhanden, so wird die entsprechende Eintragung ausgegeben.

In einer Suchzeichenfolge können die beiden Sonderzeichen "*" und "?" verwendet werden. Das "?" steht für ein beliebiges Zeichen. Beispiel:

```
load "$0:BELEGE-??",8
```

sucht in der Directory von Laufwerk 0 nach allen Namen die mit "BELEGE-" beginnen, und dann noch genau zwei Zeichen haben, wie z.B.

```
BELEGE-80
BELEGE-81
```

Das Zeichen "*" steht für eine unbestimmte Anzahl von "?". Beispiel:

```
load "$0:*",8
```

ist identisch mit

```
load "$0",8
```

da es alle Namen der Directory von Laufwerk 0 auflistet.

```
load "$0:*PROGRAM*",8
```

sucht nach Namen welche die Zeichenfolge "PROGRAM" enthalten. Beispiele dafür sind:

```
TESTPROGRAMM-1
PROGRAMM-ABCD
```

Im ersten Namen ersetzt das "*" die Zeichenfolge "TEST", das zweite "M-1". Wie der zweite Name zeigt, muß an der Stelle eines "*" kein Zeichen stehen. Die beiden Sonderzeichen können beliebig oft und an beliebiger Stelle in einer Suchzeichenfolge stehen.

Die Hard - Disk kennt eine weitere Form des Directory - Befehls. Bei der normalen Form des Directory - Befehls werden die Informationen in Form eines ladbaren Programms an den CBM übergeben. Durch die Option `,d ` werden diese als ASCII - Zeichen übergeben. Damit besteht eine einfache Möglichkeit eine Directory aus einem Programm einzulesen.

Beispiel:

```
100 open 1,8,0, "$0:*A*,d"
.
.
.
150 input 1,a$,b$,c$
160 if st=64 then goto 1000
.
.
.
1000 close 1
```

In 100 wird der Befehl gegeben alle Eintragungen welche ein `A `enthalten in der aktuellen Directory zu suchen und als ASCII - Daten zu übergeben. In 150 wird eine Eintragung eingelesen. Wenn keine weiteren Eintragungen vorhanden sind, setzt die Disk EOI, d.h. st enthält den Wert 64. Die Überschrift und die Schlußzeile der normalen Directory - Liste werden bei dieser Form nicht übergeben.

Das Abspeichern von Programmen

Programme, welche im Speicher des CBM stehen, können mit dem SAVE-Befehl auf die Hard-Disk gespeichert werden. Beispiel:

```
save "PROGRAMM-1",8
```

oder

```
dsave "PROGRAMM-1"                    beim BASIC 4.0
```

speichert das Programm, welches im Speicher des Rechners steht, unter dem Namen "PROGRAMM-1" auf Laufwerk 0.
Mit dem LOAD-Befehl können Programme von der Hard-Disk geladen werden.

Beispiel:

```
load "MEIN-PROGRAMM",8
```

oder

```
dload "MEIN-PROGRAMM"                 beim BASIC 4.0
```

Eröffnen und Schließen von Dateien

Wenn Daten aus einer Datei gelesen oder in eine Datei geschrieben werden sollen, muß diese zuvor mit dem OPEN-Befehl eröffnet werden. Beispiel:

```
open 5,8,2,"0:ADRESS-DATEI"
```

eröffnet auf Laufwerk 0 die Datei mit dem Namen "ADRESS-DATEI". Die Hard - Disk 24 MB kann bis zu acht Dateien gleichzeitig geöffnet haben. Alle geöffneten Dateien müssen verschiedene Kanäle, d.h.Sekundär-Adressen, haben. Die Sekundäradressen 0 und 1 sind für SAVE und LOAD besetzt. Die Sekundär-Adresse 15 hat eine spezielle Bedeutung als Befehls- und Statuskanal und kann nicht zum Lesen und Schreiben in Dateien verwendet werden. Nachdem eine Datei eröffnet ist, kann mit dem Befehl

```
input#fn,Variablenliste
```

aus ihr gelesen werden. "fn" ist die logische File-Nummer, welche beim OPEN-Befehl festgelegt wird. Mit

```
print#fn,Liste
```

wird in die Datei geschrieben. PRINT und INPUT sind BASIC Befehle. Sie sind in den CBM Unterlagen genauer beschrieben.

Nachdem eine Datei abgearbeitet ist, muß sie mit dem CLOSE-Befehl geschlossen werden. Beispiel:

```
close 3
```

schließt die Datei, welche mit der logischen File-Nummer 3 eröffnet worden war. Eine Datei in die geschrieben worden ist, muß unbedingt geschlossen werden, da sonst Daten verloren gehen können. Beim PRINT-Befehl werden die Daten in einem Pufferspeicher im Controller abgelegt. Sie werden erst auf die Platte geschrieben wenn der Puffer voll ist, der Zeiger der Datei (siehe unten) in einen anderen Bereich positioniert wird, oder die Datei mit dem CLOSE-Befehl geschlossen wird. Ferner muß eine Datei spätestens dann geschlossen werden, wenn auf dem gleichen Kanal eine andere eröffnet werden soll.

Beim BASIC 4.0 kann auch die Befehlsform

```
dopen   und
dsave
```

verwendet werden.

Adressierung der Daten einer Datei

Jede eröffnete Datei besitzt einen Zeiger. Nach dem Eröffnen der Datei mit dem OPEN-Befehl zeigt dieser auf das erste Byte, ausgenommen wenn die Datei mit der Option ",a" eröffnet wurde. Dann zeigt der Zeiger hinter das letzte Byte.
Beispiel:
Mit dem Befehl

```
open 6,9,5,"1:KUNDEN,a"
```

wird die Datei mit dem Namen "KUNDEN" auf Laufwerk 1 eröffnet. Der Zeiger steht anschließend hinter dem letzten Byte der Datei.

Im BASIC 4.0 lautet der Befehl APPEND.

```
append#6,"KUNDEN"
```

Beim INPUT oder PRINT Befehl beginnt das Lesen oder Schreiben bei der aktuellen Position des Zeigers. Dieser wird dabei um die Anzahl der gelesenen bzw. geschriebenen Bytes weitergerückt. Wenn beim PRINT Befehl das Ende einer Datei erreicht wird, wird diese (sofern noch freier Speicherplatz auf der Platte vorhanden ist) automatisch um 512 Bytes verlängert.

Relative Dateien

Dateien können in Sätze unterteilt werden. Alle Sätze einer Datei haben gleiche Länge. Diese kann zwischen 1 und 255 Bytes betragen. Sie wird beim OPEN Befehl durch die Option ",lx" angegeben (x = Satzlänge).

Beispiel:

```
dopen#3 "PATIENTEN-KARTEI",l200                    beim BASIC 4.0
```

eröffnet die Datei "PATIENTEN-KARTEI", setzt den Zeiger auf das erste Byte des ersten Satzes, und legt die Satzlänge auf 200 Bytes fest.
Mit dem RECORD-Befehl kann der Byte-Zeiger in einer Datei beliebig positioniert werden.
Mit den Befehlen

```
record#fn,x,y                                      beim BASIC 4.0
```

wird der Zeiger in der Datei, welche auf Kanal "kn" eröffnet ist, positioniert. Wenn ",y" nicht angegeben ist, wird er auf das erste Byte, sonst auf das y-te Byte, des x-ten Satzes gestellt.

Suchen in einer Datei

Das Durchsuchen einer Datei nach einer bestimmten Zeichenkette ist ein oft benötigter und, sofern das Suchen mittels BASIC-Programm erfolgt, zeitaufwendiger Vorgang. Die Daten müssen (1) von der Platte gelesen, dann (2) über den relativ langsamen IEC-Bus des CBM's in den Rechner übertragen, und schließlich (3) in einer BASIC-Schleife mit der vorgegebenen Zeichenkette verglichen werden. Mit dem SEARCH-Befehl der Hard-Disk entfällt der zweite Schritt. Die Daten werden gleich im Controller von einem vergleichsweise zu (3) sehr schnellen Suchprogramm verglichen. Wenn die Zeichenfolge gefunden wird, steht der Zeiger anschließend am Anfang der gesuchten Daten. Diese können dann mit einem INPUT-Befehl gelesen werden. So müssen nur die gesuchten Daten, und nicht alle zu durchsuchenden, über den Bus übertragen werden.
Der Befehl zum Suchen lautet:

```
open fn,8,15              (wenn nicht schon vorher erfolgt)
print#fn,"fkn,Zeichenfolge
```

Die zu durchsuchende Datei muß zuvor mit der Sekundäradresse kn eröffnet worden sein. Das Suchen beginnt an der aktuellen Position des Zeigers. Gegebenenfalls muß dieser zuvor mit dem RECORD-Befehl auf die gewünschte Ausgangsposition gestellt werden.

In der Suchzeichenfolge können die beiden Sonderzeichen "?" und "*" verwendet werden. Sie haben die gleiche Bedeutung wie beim Suchen in einer Directory (siehe Beschreibung des DIRECTORY-Befehls). Mit einem "*" können maximal 40 Zeichen übersprungen werden. Die zu suchende Zeichenkette darf maximal 128 Bytes lang sein. Die zu durchsuchende Datei kann beliebig groß sein. Die Suchzeichenfolge darf nicht mit mit "*" oder "?" beginnen.
Beispiel:

In der Datei "KUNDEN" stehen Namen und Nummern in folgendem Format:

```
$Name,Vorname,Kundennummer CR$Name,Vorname,KundennummerCR$....
```

Das Programm

```
10 open 5,8,3,"0:KUNDEN"
20 open 6,8,15
30 print#6,"f3,$M*ER,P*,1"
```

positioniert den Zeiger auf das "$" vor dem ersten Namen welcher mit "M" beginnt und mit "ER" aufhört und dessen Vorname mit "P" anfängt und dessen Kundennummer mit "1" beginnt. Da der Suchbefehl in Zeile 30 etwas kompliziert aussieht, soll er näher erklärt werden: Der Befehl wird über den Befehlskanal (15) gegeben. Dieser wurde in 20 eröffnet. "f3" besagt, daß in der Datei, welche auf Sekundäradresse 3 eröffnet ist, gesucht werden soll. Die Suchzeichenfolge lautet "$M*ER,P*,1". Es wird also zuerst nach den beiden Zeichen "$M" gesucht, dann können beliebige Zeichen folgen (bis zu 40), anschließend muß die Zeichenfolge "ER,P" stehen, usw.

Das Programm

```
10 open 5,8,3,"0:KUNDEN"
20 open 7,8,15
30 print#7,"f3,$M??ER,P*,1"
40 input#5, N$,V$,K
50 print N$,V$,K
```

druckt den ersten Namen aus, welcher mit M beginnt und dann zwei beliebige Zeichen hat und anschließend mit den beiden Buchstaben "ER" endet und dessen Vornamen mit "P" anfängt usw. Im Gegensatz zum ersten Programm müssen zwischen "M" und "ER" genau zwei Zeichen stehen. Die erste Suchbedingung erfüllen z.B. sowohl "MÜLLER" als auch "MEIER", die zweite nur "MEIER". Beim zweiten Beispiel wird nach dem Suchen der Name, Vorname, und die Kundennummer eingelesen. Dadurch steht der Zeiger anschließend auf dem "$" vor dem nächsten Namen. Mit einem weiteren

```
60 print#7,"f3,$M??ER,P*,1"
```

kann nach dem weiteren Eintrag gesucht werden.
Unmittelbar nach einem Suchbefehl kann über Kanal 15 die Position des Zeigers eingelesen werden. Beispiel: die Befehlsfolge

```
70 input#7,A,B$,C,D
80 print A,B$,C,D
```

liefert auf dem Bidschirm

```
5        STRING FOUND              X        Y
```

oder

```
6        STRING NOT FOUND          X        Y
```

je nachdem, ob ein weiterer Eintrag mit entsprechendem Name usw. gefunden wurde, oder nicht. X gibt die Satznummer an, Y die Position des Bytes in X. Weitere Eintragungen können in einer Programmschleife gesucht werden, welche bei A=6 abbricht.

Beim Suchen in sequentiellen Files beträgt eine Satzlänge bei der Berechnung von x und y automatisch 256 Bytes.

Löschen von Dateien

Dateien werden mit dem SCRATCH-Befehl gelöscht. Eine einzelne Datei wird durch Angabe des vollständigen Namens gelöscht. Durch die Verwendung der Sonderzeichen "?" und "*" können mehrere Dateien mit einem Befehl gelöscht werden.

```
scratch d0,"BELEGE-196?"                    bei BASIC 4.0
```

löscht alle Dateien deren Name mit "BELEGE-196" beginnen, und dann noch genau ein weiteres Zeichen besitzen.

Ändern von Namen

Mit dem RENAME-Befehl können Namen geändert werden. Befehlsform:

```
rename dx, "AlterName" to "NeuerName"
```

Kopieren von Dateien

Mit dem COPY-Befehl werden Dateien von einer Hard-Disk auf eine andere, von einer Directory in eine andere (siehe Kapitel über hierarchisches Dateisystem), oder in die gleiche Directory kopiert. Es gibt drei Formen des COPY-Befehls.
Bei der Befehlsform

```
copy dx to dy                                beim BASIC 4.0
```

werden alle Dateien der aktuellen Directory von Laufwerk x in die aktuelle Directory auf Laufwerk y kopiert.

Die Befehlsform

```
copy dx,"Name1" to dy,"Name2"
```

kopiert die Datei "Name1" von Laufwerk x auf Laufwerk y, und trägt sie dort unter "Name2" in die Directory ein. Wenn in die gleiche Directory kopiert wird, müssen "Name1" und "Name2" verschieden sein.
Mit der dritten Form des COPY-Befehls können mehrere Dateien durch Verwendung der Sonderzeichen "?" und "*" in Name1 gleichzeitig kopiert werden. Name2 muß auf eine Directory zeigen. Die kopierten Dateien erhalten die gleichen Namen wie die Originale.

Beispiel: der Befehl

```
copy "*PROG" to "MEINE DIRECTORY"
```

kopiert alle Dateien mit Namen in denen "PROG" vorkommt in die Directory "MEINE DIRECTORY".

Zusammenfügen von Dateien

Der CONCAT-Befehl ist eine besondere Form des COPY-Befehls. Mit ihm werden zwei Dateien in eine neue kopiert. Die beiden ersten bleiben unverändert erhalten.

Befehlsform:

concat "NAME1" to "NAME2"

Dabei werden die Dateien "NAME1" und NAME2" in eine neue Datei kopiert, welche den Namen "NAME2" erhält. Die alte Datei mit "NAME2" wird gelöscht.

3. Einführung in das hierarchische Dateiensytem

Die CBM 8050 Floppy - Disk hat ein Speichervermögen von ca. 500 k pro Laufwerk. Das Betriebsystem ist so organisiert, daß alle Programme und Dateien in einer Directory vermerkt sind. Selbst bei diesem relativ kleinen Speicherraum wird die Directory durch die Vielzahl der Eintragungen sehr unübersichtlich. Programme und Dateien wiederzufinden wird mühsam, Fehler entstehen durch Verwechselung der Namen und ein Schutz für Programme oder Dateien ist nicht möglich.

Der Speicherraum der Hard - Disk 24 MB ist etwa 40 mal so groß wie bei einem CBM 8050 Laufwerk. Dies erfordert eine leistungsfähigere Dateienverwaltung als sie bei herkömmlichen Floppy - Disk Systemen üblich ist. Bei einigen hundert Eintragungen ist ein schneller Überblick nicht mehr möglich. Besonders chaotisch werden die Verhältnisse, wenn mehrere Programmierer in einer einzigen Directory arbeiten.

Bei Rechnern mit großen Kapazitäten bzw. Mehrbenutzeranlagen wird in der Regel ein hierarchisches Dateiensystem implementiert. Es bietet die Möglichkeit mehrere Directories auf einem Laufwerk zu haben. Für jedes Projekt und jeden Anwender kann eine eigene Directory angelegt werden. Ungewollte gegenseitige Beeinflußung durch Verwechselung der Dateien, Namenskonflikte und dergleichen sind damit weitgehend ausgeschlossen. Ein Schutz vor Fremdzugriff für bestimmte Bereiche ist möglich. Dieses hierarchische Dateiensystem ist auf der Hard - Disk 24 MB implementiert.

Dies wird an folgendem Beispiel näher erläutert:

Beispiel 1

Die Hard - Disk wird von drei Personen benutzt, die an den Projekten Buchhaltung, Ausschreibung und Techn. Handbuch mit den zugehörigen Programmen und Dateien arbeiten.
Bild 1 zeigt ein System mit mehreren Directories. Die Directories sind durch Kreise, die Dateien durch Rechtecke und die Zeiger durch Pfeile mit Namen dargestellt. Die Buchstaben und Ziffern in den Kreisen und Rechtecken dienen nur zur Erläuterung.

In der Haupt - Directory (Root - Directory) stehen vier Namen:

"Buchhaltung"
"Ausschreibung"
"Techn. Handbuch"
"Schach"

Die ersten drei Namen zeigen auf die weiterführenden Directories. "Schach" zeigt auf ein Programm, daß direkt geladen werden kann. An dieser Stelle trennen sich die Bereiche der Bearbeiter; jeder sieht nur seinen eigenen Bereich mit den zugehörigen Programmen und Dateien.

In diesem Beispiel ist der Weg "Buchhaltung" mit dem Passwort P3820 belegt. Dieses Passwort erscheint nicht in der Directory. Nur derjenige Benutzer, der das Passwort kennt kann in diesem Bereich arbeiten. Die Dateien A,B,D,E,F sind also geschützt. Lediglich die Datei C kann über "Ausscheibung" erreicht werden.

Die Directory (2) enthält die Namen:

"FIBU"
"Belege"
"Lohn+Gehalt"
"Adress-Kartei"

"Belege" zeigt wiederum auf eine Directory, die anderen Namen zeigen auf Dateien.

Ein Programm oder eine Datei kann unter Angabe des Weges geladen werden. Z.B. sind die "Belege 1979" über den Weg "Buchhaltung/Belege/Belege 1979" erreichbar. Dateien aus anderen Directories sind nur dann erreichbar, wenn sie durch den "Link" Befehl angebunden worden sind. In diesem Beipiel kann aber nur derjenige, der das Passwort für den Bereich "Buchhaltung" kennt die Datei "Adress - Kartei" für die "Ausschreibung" freigeben. Sie wird von dort aus unter dem Namen "Kunden" abgespeichert.

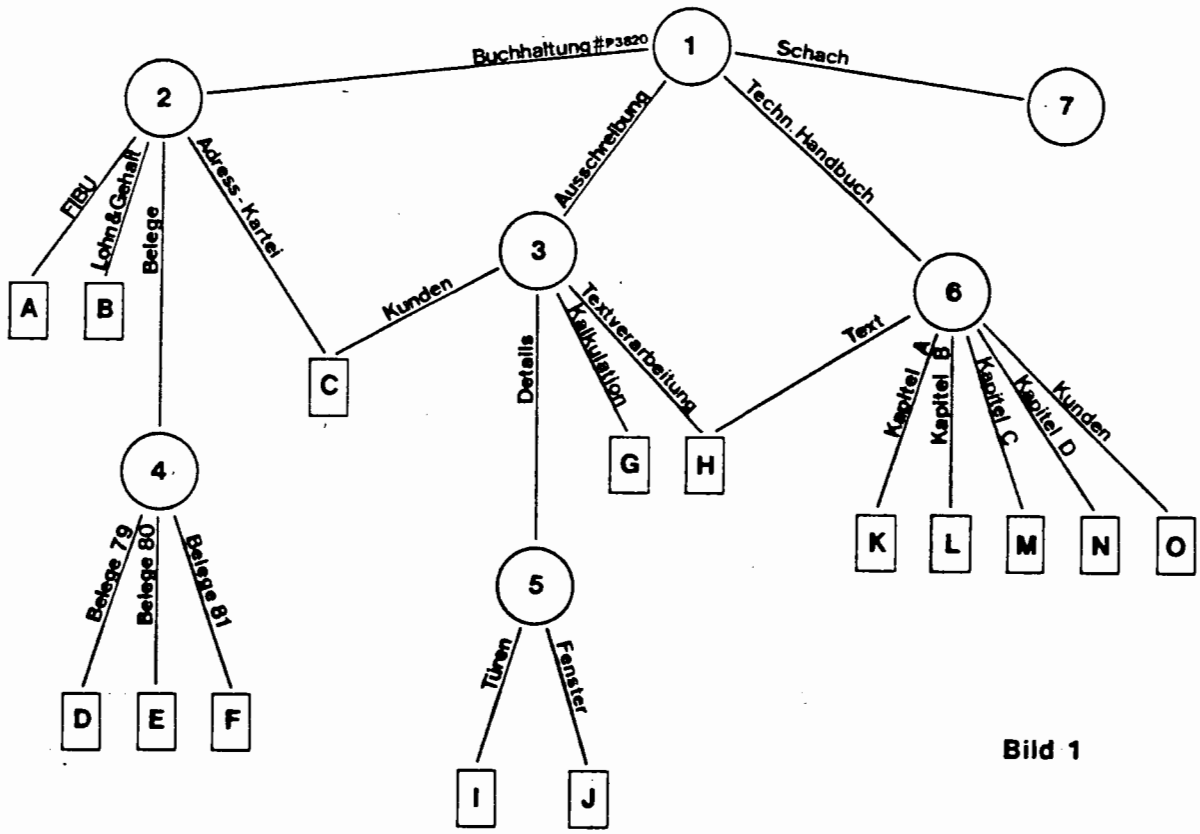

Bild 1

4. Das hierarchische Dateiensytem

Ein allgemeiner Überblick mit Beispielen für das Arbeiten mit dem hierarchischen Dateiensystem wurde schon in Kapitel 1 gegeben.

Im Gegensatz zu den meisten Floppy-Betriebssystemen haben Dateien bei der Hard - Disk keine Namen. Die Namen, welche in den Directories eingetragen sind, sind vielmehr Zeiger auf Dateien. Auf eine Datei können gleichzeitig mehrere Namen aus einer oder aus verschiedenen Directories zeigen. Solange auf eine Datei nur ein Zeiger zeigt, spielt es keine Rolle, ob dieser als Name der Datei oder als Zeiger angesehen wird. Da diese Einschränkung im vorigen Kapitel gemacht wurde, war es dort nicht notwendig diesen Unterschied zu berücksichtigen.

Namen können außer auf Dateien auch auf Directories zeigen.

Jedes Laufwerk besitzt nach dem Formatieren eine Directory. Diese wird im Folgenden als "Haupt-Directory" (Root-Directory) bezeichnet.

Bild 1 zeigt ein System mit mehreren Directories. Die Directories sind durch Kreise, die Dateien durch Rechtecke, und die Zeiger durch Pfeile mit Namen dargestellt. Die Buchstaben und Ziffern in den Rechtecken und Kreisen dienen nur zur Erläuterung. Sie sind keineswegs Namen von Dateien oder Directories. In der Haupt-Directory stehen vier Namen:

"BUCHHALTUNG"
"AUSSCHREIBUNG"
"TECHN. HANDBUCH"
"SCHACH"

Die ersten drei zeigen auf Directories, der vierte Name zeigt auf ein Programm (Datei).
Mit dem Befehl

```
dload "SCHACH"
```

kann das Programm geladen werden.
Die Directory (2) auf welche "BUCHHALTUNG" zeigt, hat ihrerseits vier Zeiger: drei auf Dateien und einen auf eine weitere Directory.

Eine Datei wird über einen Weg angesprochen. Ein Weg besteht aus einer Aneinanderreihung von Namen, welche durch "/" getrennt sind.
Beispiel:
Der Befehl

```
open 3,8,4,"AUSSCHREIBUNG/KALKULATION"
```

eröffnet die Datei (G). Der Befehl

```
open 4,8,5,"AUSSCHREIBUNG/DETAILS/TÜREN"
```

eröffnet die Datei (I).
Die Anzahl der Namen in einem Weg wird nur durch die maximale Länge der BASIC-Befehle begrenzt.

Ein Weg beginnt in der "aktuellen Directory". Nach dem Einschalten der Hard-Disk ist die Haupt-Directory zugleich auch aktuelle Directory. Mit dem CHANGE-DIRECTORY-Befehl kann eine andere Directory zur aktuellen Directory gemacht werden. Er hat die Form:

```
print#fn,"ddr:Weg"
```

dr ist die Nummer des Laufwerks, "Weg" muß auf eine Directory zeigen. Der CHANGE-DIRECTORY-Befehl muß über Kanal 15 gegeben werden.
Beispiel: nach den Befehlen

```
open 3,8,15
print#3,"d0:"AUSSCHREIBUNG"
```

ist (3) die aktuelle Directory. Anschließend kann z.B. mit dem Befehl

```
dload "TEXTVERARBEITUNG"
```

das Programm aus der Datei (H) geladen werden, oder mit

```
open 6,8,8,"DETAILS/FENSTER"
```

die Datei (J) eröffnet werden.

Der DIRECTORY-Befehl bezieht sich auf die aktuelle Directory, sofern nicht expliziet ein Weg angegeben ist.

```
load "$0"
list
```

listet (nach dem obigen CHANGE-DIR-Befehl) die drei Namen

```
"KUNDEN"
"KALKULATION"
"DETAILS"
"TEXTVERARBEITUNG"
```

Wenn vor dem ersten Namen eines Weges das Zeichen "/" steht, beginnt der Weg in der Haupt-Directory, unabhängig davon welche Directory gerade aktuelle Directory ist.
Beispiel:

```
open 6,8,8,"/AUSSCHREIBUNG/DETAILS/FENSTER"
```

eröffnet immer Datei (J).
Beispiel eines DIRECTORY-Befehles mit Weg:

```
load "$0:/BUCHHALTUNG663820/BELEGE/*"
list
```

listet die drei Eintragungen "Belege 79", "Belege 80" und "Belege 81".

In jeder Directory gibt es automatisch zwei Eintragungen: "." und "..". Diese sind in Bild 1 nicht eingetragen. "." zeigt immer auf die Directory in der die Eintragung steht. ".." zeigt auf die Directory davor (d.h. in Richtung Haupt-Directory).

Bei den Befehlen COPY, CONCAT, SCRATCH und RENAME können, anstelle der in den vorhergehenden Kapiteln verwendeten Namen, auch Wege angegeben werden. Beispiel:

```
open 7,8,15
print#7,"r0:KUNDEN664911=/AUSSCHREIBUNG/KUNDEN"
```

verpaßt dem Namen "KUNDEN" das Passwort "P4911".

Neue Directories werden mit dem MKDIR-Befehl (make directory) erzeugt. Befehlsform:

```
print#fn,"mdr:Weg/Name"                    über Kanal 15
```

dr ist die Laufwerksnummer, "Weg" muß auf eine Directory zeigen. Wenn kein Weg angegeben ist, wird die neue Directory in die aktuelle Directory eingetragen. Beispiel eines MKDIR-Befehls:

```
open 3,8,15
print#3,"m0:/BUCHHALTUNG663820/BILANZEN"
```

Der Weg in diesem Beispiel lautet "/BUCHHALTUNG". "BILANZEN" ist der Name welcher auf die neue Directory zeigt. Nach diesen Befehlen kann z.B. mit

```
open 4,8,5,"0:/BUCHHALTUNG/BILANZEN/B1981,w"
```

eine neue Datei erzeugt, und in die neu angelegete Directory eingetragen werden.
Es kann vorkommen, daß Benutzer öfter auf die gleichen Programme oder Daten zugreifen wollen. Dann gibt es u.a. folgende Möglichkeiten:
- die gemeinsam benötigten Dateien sind in der Directory eines Benutzers eingetragen und alle andern geben jeweils den vollen Weg zu diesen an,
- oder alle gemeinsamen Dateien sind in der Haupt-Directory eingetragen,
- oder jeder Benutzer hat eine Kopie in seinem Bereich (Speicherplatzverschwendung).

Die Hard-Disk bietet eine weitere Möglichkeit: auf eine Datei können mehrere Zeiger aus unterschiedlichen Directories zeigen. Das ist in Bild 1 bei den Dateien (C) und (H) der Fall. Datei (C) kann mit dem Namen "ADRESS - KARTEI" aus Directory (2) oder mit "KUNDEN" über Directory (3) angesprochen werden.

Wenn einer der beiden Benutzer den Inhalt einer gemeinsamen Datei ändert, ist dieser natürlich für beide geändert. Das ist bei Daten gewöhnlich auch erwünscht, z.B. wenn die Datei Adressen enthält und es wird eine Adressänderung eingetragen. Bei Programmen kann es aber zu Verwirrungen kommen, besonders dann, wenn die Änderungen den übrigen Benutzern nicht mitgeteilt werden. Deshalb sollte dieses Verfahren nur bei fertigen, ausgetesteten Programmen angewendet werden.

Eine weitere Eigenschaft des hierarchischen Dateiensystems soll anhand von Bild 1 erläutert werden. Angenommen die Datei (H) enthält ein Programm mit der Anweisung

```
10 open 3,8,2,"KUNDEN"
```

Weiterhin sei angenommen, daß (3) oder (6) die aktuelle Directory ist. Wenn dies (3) ist, wird mit dem OPEN-Befehl die Datei (c) eröffnet, wenn es (6) ist, die Datei (O). Damit besteht die Möglichkeit, daß mehrere Benutzer das gleiche Programm mit ihrem jeweils eigenen Datensatz verwenden.

Zusätzliche Zeiger auf Dateien können auch dazu dienen, anderen Benutzern nur diese zugänglich zu machen, alle übrigen Dateien aber vor deren Zugriff zu schützen.

Für das Löschen von Dateien auf welche mehrere Namen zeigen gilt folgendes: durch einen SCRATCH-Befehl werden nur die angegebenen Namen aus der Directory gelöscht. Andrere Zeiger auf die Datei und die Datei selbst bleiben unerändert erhalten. Erst wenn der letzte Zeiger auf eine Datei gelöscht wird, wird auch diese gelöscht.

Die Befehle LINK und REMDIR

Der LINK Befehl erzeugt einen neuen zusätzlichen Zeiger auf eine existierende Datei. Befehlsform:

```
print#fn,"lx:NeuerWeg=ExistierenderWeg"          über Kanal 15
```

Es ist nicht möglich Zeiger von einer Directory auf einem Laufwerk auf eine Datei eines anderen Laufwerks zu setzen. Mit dem LINK-Befehl können nur Zeiger auf Dateien, nicht aber auf Directories erzeugt werden.
Beispiel eines LINK-Befehls:

```
open 3,8,15
print#3,"I0:/TECHN.HANDBUCH=/AUSSCHEIBUNG/DETAILS/FENSTER"
```

Mit dem REMDIR-Befehl (remove directory) kann eine Directory gelöscht werden. Die zu löschende Directory muß leer sein, d.h. sie darf keine Eintragungen außer "." und ".." enthalten. Befehlsform:

```
open fn,8,15
print#fn,"zdr:Weg"
```

dr ist die Laufwerksnummer. "Weg" zeigt auf die zu löschende Directory.

5. Namen und Passworte

Namen können bis zu 20 Zeichen lang sein. Folgende Zeichen dürfen in Namen nicht vorkommen:

= ; , / ?

Das erste Zeichen eines Namens muß ein Buchstabe oder das Zeichen "#" sein. Letzteres hat eine besondere Bedeutung: wenn es in einem Namen vorkommt, werden alle Zeichen welche rechts davon stehen beim DIRECTORY-Befehl nicht mit ausgegeben. Es besteht auch keine andere Möglichkeit den vollen Namen zu lesen. Beim Zugreifen auf eine Datei, z.B. beim OPEN-Befehl, muß aber immer der vollständige Name angegeben werden. Damit bilden die Zeichen rechts von "#" ein Passwort. "#" kann an beliebiger Stelle des Namens stehen. Passworte können also bis zu 19 Zeichen lang sein ("#" ist Teil des Namens). Beispiel: der Name

KONTEN#DCK33A$2

wird in der DIRECTORY-Liste als "KONTEN" aufgelistet.
Passworte können mit dem RENAME-Befehl geändert werden. Beispiel:

rename"KONTEN#DCK33A$2" to "KONTEN0ALTER"

Beim COPY- und SCRATCH-Befehl müssen die Passworte angegeben werden. Es ist nicht möglich sie mit "*" oder "?" zu umgehen.

Achtung! Passworte sollten nur für wirklich vertrauliche Daten verwendet werden, da keine Möglichkeit besteht, vergessene Passworte wieder herauszufinden, oder sonstwie an die Daten zu kommen. Es sei denn, das Passwort ist kurz. Dann können mit einem BASIC-Programm alle Möglichkeiten durchprobiert werden. Deshalb sollten wichtige Passworte relativ lang sein (10 Zeichen oder mehr). Dann gibt es derart astronomisch viele Möglichkeiten, daß ein Durchprobieren praktisch unmöglich ist.

6. Kurzbeschreibung der Befehle

append#lfn,"name"(,dx) — APPEND

eröffnet die sequentielle Datei, auf welche "name" zeigt, auf Laufwerk x, zum Schreiben. Der Zeiger wird ans Ende der Datei gesetzt, so daß zusätzliche Daten angehängt werden können. "name" kann ein Weg durch mehrere bestehende Directories sein.

concat"name1"to"name2" — CONCAT

kopiert die Dateien auf welche "name2" und "name1" zeigen zusammen in eine neue Datei, und trägt diese unter "name2" ein. Die vorherige Eintragung "name2" wird gelöscht. Die Datei auf welche sie zeigte wird nur gelöscht, wenn keine weiteren Links mehr auf diese bestehen. Anstelle von "name1" kann eine Liste von Namen, welche durch Komma getrennt sind, stehen. Die Daten werden in folgender Reihenfolge kopiert: zuerst der Inhalt von "name2", dann von "name1". Wenn "name1" aus einer Liste von Namen besteht, wird diese von links nach rechts abgearbeitet. Alle Namen dürfen aus Wegen durch bestehende Directories zusammengesetzt sein.

copy"name1"to"name2" — COPY

kopiert die Datei auf welche "name1" zeigt, und trägt sie unter "name2" ein. "name1" und "name2" können Wege durch bestehende Directories sein. Wenn beide in die gleiche Directory zeigen, müssen ihre letzten Komponenten verschieden sein.

copy"name"to"directory"

kopiert Dateien und trägt sie in der Directory ein, auf die "directory" zeigt. "name" kann einen Weg durch existierende Directories enthalten. Die letzte Komponente von "name" muß "*" oder "?" enthalten. Alle Dateien auf die die Bedingungen von "name" zutreffen, werden kopiert. Directories können nicht kopiert werden. "directory" kann ein Weg durch existierende Directories sein. "name" und "directory" dürfen nicht auf die gleiche Directory zeigen. Die Kopien erhalten die gleichen Namen wie die Originale.

dclose#lfn — DCLOSE

schließt die Datei, welche unter der logischen Filenummer 'lfn' eröffnet worden war. Alle offenen Dateien müssen nach Gebrauch wieder geschlossen werden

directory — DIRECTORY

listet die aktuelle Directory auf dem Bidschirm. Die Liste enthält folgende Angaben:

Nummer, Name, Länge, Links, Typ, Dateinummer

Nummer ist eine laufende Durchnummerierung der aufgelisteten Eintragungen
Name ist der Name der Eintragung

Länge gibt die Länge der Datei als Anzahl ihrer Sektoren a 512 Bytes an

Typ gibt an, ob die Eintragung auf eine Directory, ein Programm, oder Daten zeigt

Dateinummer ist eine Nummer die jeder Datei und Directory vom Betriebssystem zuteilt wird. An ihr kann man erkennnen, welche Links auf die gleiche Datei zeigen.

dload"fn",ddr — DLOAD

lädt ein Programm von Laufwerk dr. "fn" Kann ein Weg durch mehrere Directories sein. Wenn "fn" nur einen Namen enthält, wird dieser in der aktuellen Directory gesucht.

dopen#lfn,"name"(,ly)(,dx)(,w) — DOPEN

eröffnet die Datei, auf welche "name" zeigt, auf Laufwerk x. Ist kein Laufwerk angegeben, so wird automatisch Laufwerk 0 benutzt. "name" kann ein Weg durch existierende Directories sein. Wenn 'ly' angegeben ist, wird eine relative Datei mit Records der Länge y eröffnet, bzw. neu angelegt, wenn es nicht existiert. 'w' muß angegeben werden, wenn eine neue sequentielle Datei angelegt werden soll.

dsave"fn",ddr — DSAVE

schreibt das Programm, welches im Speicher des CBM steht, auf Laufwerk dr. "fn" kann einen Weg durch mehrere, existierende Directories enthalten. Wenn "fn" nur aus einem Namen besteht, wird das Programm in der aktuellen Directoy eingetragen.

ds$ und ds — DS$, DS

sind zwei BASIC 4.0 Variablen, welche die Statusmeldungen der Disk enthalten

header"name",dx — HEADER

formatiert das Laufwerk x, sofern der Formatierer nicht gesperrt ist. Dabei werden alle Directories und Dateien gelöscht. Damit dies nicht versehentlich geschieht, ist der Formatierer normalerweise gesperrt. Der HEADER-Befehl führt dann zu der Meldung "bad disk" und "format protect on". Durch Entfernen eines Jumpers auf der Controller-Platine wird die Sperre aufgehoben. Es wird empfohlen den Jumper nach dem Formatieren wieder anzubringen. Das Formatieren dauert ca. anderthalb Minuten.

record#lfn,r(,b) — RECORD

positioniert den Zeiger in der relativen Datei, welche zuvor unter der logischen Filenummer 'lfn' eröffnet worden sein muß, in das r-te Record. Wenn b angegeben ist, wird der Zeiger auf das b-te Byte, sonst auf das erste Byte des Records gesetzt.

rename"name1"to"name2" — RENAME

benennt die Eintragung "name1" in "name2" um. "name1" darf aus einem Weg durch existierende Directories zusammengesetzt sein, "name2" nicht.

scratch(ddr,)"name" — SCRATCH

löscht Eintragungen in der Directory. Durch Verwendung von "?" und "*" können mehrere Eintragungen mit einem Befehl gelöscht werden. Eintragungen welche auf eine Directory zeigen können nicht mit dem SCRATCH Befehl gelöscht werden (siehe ZERO-Befehl). "name" kann aus einem Weg durch existierende Directories bestehen. Beim Löschen einer Eintraung wird der Link-Zähler der entsprechenden Datei um 1 erniedrigt. Wenn er dabei den Wert 0 erreicht, wird die Datei gelöscht, d.h. der Speicherplatz der Datei wird dem Pool der freien Sektoren zugeordnet.

Neue Befehle

Für die folgenden Befehle gibt es keine direkten BASIC Anweisungen. Sie müssen deshalb mit dem print Befehl über den Befehlskanal (Sekundäradresse 15) übermittelt werden. In den folgenden Beispielen wird angenommen, daß dieser zuvor mit der Anweisung

```
open 1,8,15
```

eröffnet worden ist.

print#1,"ddirectory" — CHANGE DIR

macht die Directory auf die "directory" zeigt, zur aktuellen Directory. "directory" kann ein Weg durch bestehende Directories sein. Nach dem Einschalten ist die "root" Directory aktuelle Directory.

print#1,"mname" — MAKE DIR

erzeugt eine neue Directory, und trägt sie unter "name" ein. "name" kann aus einem Weg durch existierende Directories bestehen.

print#1,"zname" — ZERO DIR

löscht die Directory, auf die "name" zeigt. Diese muß leer sein, d.h. es dürfen keine Eintragungen außer "." und ".." in ihr stehen. "name" kann ein Weg durch bestehende Directories sein.

print#1,"fx,string" — FIND

sucht die Zeichenkette 'string' in der Datei, welche zuvor unter der Sekundäradresse x eröffnet worden ist. 'string' kann die beiden Suchzeichen "*" und "?" beliebig oft, jedoch nicht als erste Zeichen, enthalten. Mit "*" können maximal 40 Zeichen übersprungen werden. Die zu suchende Zeichenkette kann maximal 128 Zeichen lang sein. Wenn eine passende Zeichenkette gefunden wurde, steht der Zeiger anschließend auf dem ersten Zeichen der gefundenen Zeichenkette. Seine Position kann anschließend über den Statuskanal eingelesen werden. Meldung:
05 string found , r, b
oder
06 string not found
r ist die Nummer des Records, in der der Zeiger steht, b ist die Byte-Positon des Zeigers
Da beim DOPEN Befehl die Sekundäradresse unter der die Datei eröffnet wird nicht bekannt ist, müssen Dateien in denen mit dem FIND-Befehl gesucht werden soll, mit dem OPEN-Befehl eröffnet werden.

7. Unterschiede zur CBM 8050

1) Der COLLECT-Befehl ist durch die andere interne Organisation der Hard Disk überflüssig. Er wird ignoriert.
Das gleiche gilt für den INITIALIZE-Befehl der 3040.

2) Der BACKUP-Befehl ist nicht implementiert

3) Das Directory-Listing hat ein anderes Format

4) Die beiden Zeichen '/' und '#' haben eine besondere Bedeutung in Namen. '/' ist Trennungszeichen von Namen bei der Angabe eines Weges durch mehrere Directories. '#' trennt den Passwortteil vom Rest eines Namens.

5) Da die Hard - Disk intern mit einem anderen Prozessor arbeitet und anders organisiert ist, sind die Block-, User- und Memory-Befehle nicht implementiert.

6) Bei verschiedenen Fehlern erfolgt eine andere Fehlermeldung. Auch die Aktion nach verschiedenen Fehlern ist anders (Sperre weiterer Befehle bis zur Abfrage der Fehlerursache, Initialisierung des Betriebssystems)

7) Bei der Hard - Disk muß jeder offene Kanal geschlossen werden, bevor er neu eröffnet werden kann. (Sonst Fehlermeldung)

8) Relative Dateien arbeiten im Großen und Ganzen wie bei der 8050. Sie sind allerdings intern anders organisiert. Das führt in zwei Fällen zu Abweichungen gegenüber der 8050:

   a) Bei Recordlängen von mehr als 80 Bytes, soll der eingespeicherte Text mit 'carriage return' (CR) abgeschlossen werden, da sonst die Fehlermeldung 'string to long' beim input-Befehl auftreten kann. Grund: Die Hard-Disk übergibt die Null-characters welche das Ende eines Records auffüllen, es sei denn, vom CBM kommt ein 'untalk'. Die Null-characters werden zwar vom input Befehl ignoriert, können aber das Eingabebuffer des CBM zum Überlaufen bringen. Abhilfe: CR ans Ende des Textes schreiben, dann sendet der CBM ein 'untalk', und der Zeiger in der Hard - Disk springt zum nächsten Record.
   b) Der folgende Unterschied macht sich nur beim Lesen aus relativen Dateien mit dem GET-Befehl bemerkbar. Die 8050 übergibt das letzte gültige Zeichen (d.h. das letzte Zeichen welches von Null verschieden ist) mit einem EOI. Die Hard - Disk übergibt dieses Zeichen ohne EOI.

8. Fehler und Statusmeldungen

Beim Auftreten von Fehlerbedingungen leuchtet die rote Anzeige auf der Vorderseite der Hard Disk auf. Über Kanal 15 (Sekundäradresse 15) kann die Status- bzw. Fehlermeldung abgefragt werden. Nach der Befehlsfolge:

```
open1,8,15
input#1,a,b$,c,d
```

steht in a die Nummer der Meldung, in b$ eine kurze Erläuterung; c und d werden nur bei einigen Meldungen benutzt (siehe unten), sonst haben sie den Wert 0.

Beim BASIC 4.0 kann der Status auch mittels der Variablen ds$ abgefragt werden:

```
?ds$
```

liest die Statusmeldung ein, und druckt sie auf dem Bildschirm aus. Die Meldung wird allerdings nur einmal (nach einer Disk Operation) eingelesen. Bei weiteren Abfragen von ds$ wird die im Speicher des CBM abgelegte Meldung ausgegeben. Das kann bei einigen Befehls- und Fehlerkombinitionen zu Verwirrung führen. Der input Befehl (siehe oben) fragt den Status jedesmal neu ab, und sollte im Zweifelsfall bevorzugt werden.

Nach dem Auftreten bestimmter, schwerwiegender Fehler sperrt das Betriebssystem, d.h es werden keine weiteren Befehle angenommen, bis der Fehlerstatus abgefragt wurde. Diese Sperre ist zur Sicherheit der Daten auf der Disk eingebaut, damit nicht als Folge eines Fehlers weitere Daten fehlgeleitet werden.

Durch das Abfragen des Status wird eine gegebenenfalls gesetzte Fehleranzeige gelöscht. Nach einigen, schwerwiegenden Fehlern wird die Hard Disk dabei neu initialisiert, d.h. alle Kanäle sind anschließend geschlossen, und die "root" Directory ist aktuelle Directory.

Meldungen:

```
0,      'OK'  ,0 ,0
1,      'FILES SCRATCHED'  ,n ,0
2,      'FILES COPIED'  ,n ,0
5,      'STRING FOUND'  ,r ,b
6,      'STRING NOT FOUND'  ,r ,b
12,     'ILLEGAL SECONDARY ADDRESS'  ,0 ,0
13,     'CHANNEL NOT OPEN'  ,0 ,0
16,     'INVALID BYTE POSITION'  ,0 ,0
18,     'CHANNEL ALREADY OPEN'  ,0 ,0
22,     'SECTOR NOT FOUND'  ,t ,s
23,     'CHECKSUM ERROR'  ,t ,s
25,     'WRITE-VERIFY ERROR'  ,t ,s
27,     'ID ERROR'  ,t ,s
30,     'SYNTAX ERROR'  ,0 ,0
31,     'INVALID COMMAND'  ,0 ,0
32,     'LONG LINE'  ,0 ,0
33,     'INVALID FILENAME'  ,0 ,0
```

```
37,     'INVALID SWITCH'  ,0 ,0
39,     'NAME EXISTS'  ,0 ,0
41,     'DIRECTORY NOT FOUND'  ,0 ,0
42,     'FILE NOT DIRECTORY FILE'  ,0 ,0
43,     'UNKNOWN NAME IN PATH'  ,0 ,0
44,     'DIRECTORY NOT EMPTY'  ,0 ,0
46,     'ATTEMP TO SCRATCH DIRECTORY'  ,0 ,0
47,     'DIFFERENT DRIVES'  ,0 ,0
48,     'WILD CHARACTER IN PATH'  ,0 ,0
49,     'PASSWORD IN SEARCH STRING'  ,0 ,0
50,     'RECORD NOT PRESENT'  ,0 ,0
51,     'OVERFLOW IN RECORD'  ,0 ,0
55,     'ILLEGAL OPERATION'  ,0 ,0
56,     'ILLEGAL COPY COMMAND'  ,0  ,0
57,     'ILLEGAL CONCAT COMMAND' ,0 ,0
58,     'ILLEGAL OPERATION'  ,0  ,0
62,     'FILE NOT FOUND'  ,0 ,0
63,     'FILE EXISTS'  ,0 ,0
70,     'NO CHANNEL'  ,0 ,0
72,     'DISK FULL'  ,0 ,0
73,     'H & W DISK   VERS. 1.0'  ,0 ,0
74,     'DRIVE NOT READY'  ,0 ,0
75,     'FORMAT PROTECT ON'  ,0 ,0
78,     'HARDWARE FAILURE'  ,c ,0
99,     'ERROR'  ,c ,0
```

Anmerkungen zu verschiedenen Fehlercodes:

1 n gibt die Anzahl der gelöschten Files an.

2 n gibt die Anzahl der kopierten Files an.

5 r gibt die Nummer des Records an, in dem die Zeichenkette gefunden wurde, b die Position des ersten Bytes der Zeichenkette innerhalb des Records.

22-27 t ist die Nummer der Spur auf der ein Fehler auftrat, s ist die Nummer des Sektors.

78 diese Meldung faßt eine Reihe von Hardware - Fehlerbedingungen zusammen, welche beim Selbsttest des Kontrollers während Lese- , Schreib-, und Positionieroperationen festgestellt werden können. c gibt an, welche Signalleitung bei welcher Operation fehlerhaft war. Tritt der Fehler wiederholt auf, so muß das Gerät, mit Angabe des Fehlercodes c, zur Reparatur.

99 diese Meldung faßt eine Reihe von internen System-Fehlerbedingungen zusammen. Nach dem Auftreten dieses Fehlers, wird die Hard - Disk neu initialisiert, d.h. alle Files geschlossen, und die "root" Directory aktuelle Directory. Tritt der Fehler wiederholt auf, so muß die Disk neu formatiert werden.

9. Jumper

Auf der Controller Platine befindet sich eine Steckleiste mit 6 Steckplätzen für Jumper.

Solange auf Nummer 1 (zeigt auf die vier EPROM's) ein Jumper sitzt ist der Formatierer gesperrt. Der Header - Befehl wird dann nicht ausgeführt.

Nummer 2 ist nicht benutzt.

Mit 3-6 kann die IEC-Bus Adresse eingestellt werden:

IEC-Bus Ad.	3	4	5	6
1	x	x	x	
2	x	x		x
3	x	x		
4	x		x	x
5	x		x	
6	x			x
7	x			
8		x	x	x
9		x	x	
10		x		x
11		x		
12			x	x
13			x	
14				x

x bedeutet Jumper aufgesteckt.

Ab Werk ist die Hard Disk auf Adresse 8 eingestellt.

Unterschiede der Version 1.1 zur Version 1.0

1) Beim Einlesen mit dem "input"-Befehl aus einer relativen Datei werden pro " input" Befehl maximal 80 Zeichen übergeben. Beim 80. Zeichen wird das "EOI"-Signal gesetzt. Dadurch wird der " string to long" Fehler vermieden.

2) Im Directory Listing wird zwischen relativen "rel" und sequentiellen "seq" Dateien (wie bei der 8050) unterschieden. Die alte Angabe "dat" entfällt.

3) Beim Einlesen der Directory als ASCII Daten (siehe Hard-Disk Beschreibung Seite 3) werden alle Informationen (laufende Nummer,Name,Länge,Links, Typ,Nummer) wie beim normalen Directory Befehl übergeben.

4) Die Record-Länge eines existierenden Files kann eingelesen werden. Dazu wird die Datei eröffnet und anschließend der Status eingelesen. dieser hat das Format:

0, ok, 0, record-Länge

5) Zum besseren Schutz der Dateien und Programme wurden folgende Änderungen vorgenommen:

* Soll ein sequentielles File neu angelegt werden, so muß die Option ",w" im Open-Befehl angegeben werden.
* In existierende sequentielle Dateien und Programme kann nicht geschrieben werden. Ausnahme: mit dem Append-Befehl können Daten angehängt werden.

6) Nicht implementierte IEC-Bus Befehle werden ignoriert und führen nicht zu einem "time out".

Außerdem sind die uns bekannten Fehler der Version 1.0 behoben.

Berlin, den 27.6.82

Bedienungsanleitung: Stand Juni `82

HÜBNER & WORM GmbH - Prinzessinnenstr. 20 - 1000 Berlin 61 - Tel. 030 6145056