MZ-80B - intern Nr. 10

Für alle diejenigen MZ-80B-Besitzer, die Schwierigkeiten mit dem englischen Handbuch zum Disk-BASIC SB-6510 haben, werden an dieser Stelle alle Befehle, die über das Kassetten-BASIC SB-5510 hinausgehen, erklärt. (Das hier verwendete Paragraphenzeichen 'ß' entspricht beim MZ-80B dem Kaufmännischen 'a', auch als Schneckennudel, Affenschaukel usw. bekannt.)

RUN

RUN hat neben der schon bekannten Funktion nun auch die Möglichkeit, ein auf Diskette vorhandenes BASIC-Programm zu laden und auszuführen. Dabei werden natürlich auch alle Variablen gelöscht und ein eventuell im Speicher befindliches Programm beseitigt. Dieser Befehl gestattet wie alle Diskettenbefehle den Zusatz des Laufwerks (z. B. FD2) sowie den Zusatz der Nummer der Diskette (z. B. §5). Wird kein Laufwerk angegeben, wird das gerade aktuelle verwendet (s. DIR), wird keine Nummer angegeben, wird sie nicht überprüft. Diese Nummer ist für die meisten Fälle belanglos. Sie wird beim Formatieren zugeteilt und ist eine Sicherungsmaßnahme, um z. B. das versehentliche Verwenden einer falschen Diskette zu verhindern. Hat die Diskette in dem Laufwerk, auf das sich der RUN-Befehl bezieht, eine andere Nummer als im RUN-Befehl angegeben, wird ein ERROR 52 erzeugt. Dieser kann mit Hilfe von ON ERROR GOTO (s. u.) im Programm verarbeitet werden. Mit RUN kann auch ein Maschinenprogramm geladen und ausgeführt werden. Dabei ist es möglich, den BASIC-Interpreter zu überschreiben, wenn das Maschinenprogramm diesen Adressbereich belegt. Maschinenprogramme lassen sich nur mit Hilfe des Programms "Filing-CMT" (befindet sich auf Ihrer Disk-BASIC-Diskette) von Kassette auf Diskette bringen, eine andere Möglichkeit gibt es nicht. RUN läßt sich wie fast alle Diskettenbefehle auch innerhalb von Programmen verwenden, anstelle des Programmnamen in Anführungszeichen kann auch eine Stringvariable (z. B. A$) stehen. Der Programmname darf maximal 16 Zeichen umfassen.

Beispiel: RUN FD2 §3 "Reversi" überprüft, ob die Diskette in Laufwerk 2 die Nummer 3 hat, wenn ja, wird das Programm "Reversi" von Laufwerk 2 geladen und ausgeführt.
Beispiel: RUN "Musik-Editor" lädt und startet das Programm "Musik-Editor" vom gerade aktuellen Laufwerk.

LOAD

Mit LOAD wird ein auf Diskette befindliches BASIC-Programm in den Speicher geladen. Auch LOAD gestattet den Zusatz des Laufwerks und der Diskettennummer. Ansonsten gilt auch hier, was beim Befehl RUN beschrieben wurde. Bei den folgenden Befehlen wird darauf nicht mehr hingewiesen. Mit LOAD können auch Maschinenprogramme geladen werden. Der Bereich, in den sie kommen, sollte aber vorher mit LIMIT abgetrennt werden. Der LOAD-Befehl für Kassette heißt jetzt LOAD/T und funktioniert wie gehabt.

Beispiel: LOAD "Integration" lädt das Programm "Integration" vom gerade aktuellen Laufwerk.

SAVE

SAVE ist das Gegenstück zu LOAD und schreibt das im Speicher befindliche Programm auf Diskette. Befindet sich unter gleichem Namen schon ein Programm oder eine Datei auf der Diskette, wird ein ERROR 42 erzeugt. Mit SAVE können leider keine Maschinenprogramme auf Diskette aufgezeichnet werden. SAVE/T zeichnet auf Kasette auf.

Beispiel SAVE FD1 "Shaper" schreibt das Programm "Shaper" auf die Diskette in Laufwerk 1.

VERIFY

Arbeitet nur mit Kassette in bekannter Weise.

DELETE

Damit kann ein Programm oder eine Datei von der Diskette gelöscht werden. Es darf nicht geschützt sein (s. LOCK).

Beispiel: DELETE §17 "Statistik" überprüft die Nummer der Diskette im gerade aktuellen Laufwerk auf 17 und löscht, wenn ja, das Programm "Statistik".

LOCK

Mit LOCK wird ein Programm oder eine Datei gegen Löschen und Verändern geschützt.

UNLOCK

UNLOCK hebt diesen Schutz wieder auf.

RENAME

Damit kann der Name eines Programms oder einer Datei geändert werden. Dazu wird erst der alte Name, gefolgt von einem Komma, und dann der neue Name angegeben.

Beispiel: RENAME "Kuckuck","Kuckucksuhr" ändert den Namen des Programms "Kuckuck" auf der Diskette im gerade aktuellen Laufwerk zu "Kuckucksuhr".

CHAIN

Mit CHAIN kann von dem gerade abgearbeiteten BASIC-Programm ein anderes geladen und ausgeführt werden. Dabei gehen die Variableninhalte und Benutzerfunktionen NICHT verloren, sondern werden an das neue Programm übergeben. Der Befehl CHAIN wirkt praktisch wie GOTO "Programmname".

Beispiel: CHAIN FD2 "Eingabe" übergibt die Ausführung von dem gerade im Speicher befindlichen Programm an das Programm "Eingabe" in Laufwerk 2.

SWAP

SWAP wirkt analog zu CHAIN wie GOSUB "Programmname". Das gerade abgearbeitete Programm speichert sich selbst auf Diskette im gerade aktuellen Laufwerk (!), lädt das durch den SWAP-Befehl angegebene Programm vom eventuell angebenen Laufwerk (!) in den Speicher und übergibt die Ausführung an dieses. Dabei bleiben Variableninhalte und Benutzerfunktionen erhalten. Ist das neue Programm abgearbeitet (bei END angekommen), wird wieder das alte geladen und mit dem Befehl hinter dem SWAP-Befehl fortgesetzt. Der BASIC-Interpreter gestattet nur eine SWAP-Ebene, d. h. in einem durch SWAP nachgeladenen Programm darf kein SWAP-Befehl mehr verwendet werden.

Beispiel: Gegeben seien zwei Programme mit Namen "Programm 1" und "Programm 2", das gerade aktuelle Laufwerk sei Laufwerk 1. "Programm 1" befinde sich im Speicher.

```
10 REM Programm 1
20 A$="C1DEFGAB"
30 SWAP FD2 "Programm 2"
40 PRINT "Das war's!"
50 END
```

```
10 REM Programm 2
20 TEMPO 5
30 MUSIC A$
40 END
```

Mit RUN wird nun "Programm 1" gestartet. Es weist der Variablen A$ den String "C1DEFGAB" zu und stößt dann auf den SWAP-Befehl. Dadurch veranlaßt speichert sich das "Programm 1" in einer speziellen Weise, die den Programmierer nicht zu interesieren braucht, auf das gerade aktuelle Laufwerk, also Laufwerk 1. Danach wird der Speicher bis auf die Variablen gelöscht und das "Programm 2" von Laufwerk 2 geladen und ausgeführt. Es setzt das Tempo, spielt die Noten und ist dann beendet. Daraufhin wird wiederum der Speicher bis auf die Variablen gelöscht und von Laufwerk 1 das "Programm 1" wieder zurückgeholt. Dieses Programm wird jetzt hinter dem SWAP-Befehl, also mit Zeile 40 fortgesetzt. All diese Aktivitäten laufen völlig automatisch ab.

Mit Hilfe der Befehle CHAIN und SWAP ist es möglich, modulare Programme zu erstellen. So könnte zum Beispiel aus einem Hauptprogramm mit SWAP Programme zur Eingabe oder Ausgabe über Masken aufgerufen werden. CHAIN ermöglicht einfach die Programmierung von sehr großen Programmen, die sonst nicht in den Speicher passen würden.

DIR

DIR zeigt eine Liste aller Programme und Dateien auf einer Diskette an. Dabei ist der Zusatz des Laufwerkes und /P zur Ausgabe auf den Drucker zulässig. DIR zeigt an, ob es sich um eine Masterdiskette handelt (s. u.)und wieviele freie Sektoren noch vorhanden sind (ein Sektor entspricht 256 Bytes). Danach folgt die Liste. Ein Sternchen zeigt an, daß das Programm geschützt ist. BTX heißt BASIC-Programm, OBJ Maschinenprogramm, BSD sequentielle Datei und BRD wahlfreie Datei. Wenn nicht alle Namen auf den Bildschirm passen, erscheint der Cursor, und mit Druck auf die CR-Taste geht es weiter. übrigens können Sie Programm einfach starten oder ähnliches, indem Sie mit dem Cursor nach oben zu dem gewünschten Namen fahren und dann die entsprechende Funktionstaste drücken. Das erspart einige Tipperei. Mit DIR wird das aktuelle Laufwer bestimmt. Das Laufwerk, das zuletzt mit DIR angesprochen wurde, ist bis zum nächsten DIR das gerade aktuelle Laufwerk.

Beispiel: DIR FD2 /P druckt die Directory der Diskette in Laufwerk 2 aus und macht das Laufwerk 2 zum gerade aktuellen Laufwerk.

WOPEN

WOPEN richtet eine sequentielle Datei ein, in die anschließend Daten geschrieben werden sollen. Sequentiell heißt, daß die Daten nacheinander geschrieben werden und hinterher auch nur in der gleichen Reihenfolge nacheinander gelesen werden können. Diese Datei kann zum einen auf Kassette eingerichtet werden, dann lautet der Befehl WOPEN/T und funktioniert in bekannter Weise. Zum anderen kann die Datei auf Diskette eingerichet werden, dann muß dem Befehl WOPEN ein Doppelkreuz (#) und eine Dateinummer folgen. Danach sind hier wie bei allen anderen Befehlen die Zusätze Laufwerk und Diskettennummer zulässig, schließlich gefolgt vom Dateinamen. Die Dateinummer dient der Ansprache einer bestimmten Datei, da der Interpreter die gleichzeitige Verwendung von bis zu neun Dateien gestattet.

Beispiel: WOPEN #3, FD2 §9, "Kunden" überprüft, ob die Diskette in Laufwerk 2 die Nummer 9 hat, wenn ja, wird auf der Diskette in Laufwerk 2 eine sequentielle Datei mit Namen "Kunden" eingerichtet.
Beispiel: WOPEN/T A$ richtet auf Kassette eine sequentielle Datei mit dem Namen, der in der Variablen A$ steht, ein.

ROPEN

Mit ROPEN wird eine sequentielle Datei zum anschließenden Lesen von Daten aktiviert. Der Befehl arbeitet völlig analog zu WOPEN.

Beispiel: ROPEN #1, "Kunden" aktiviert die auf der Diskette im gerade aktuellen Laufwerk befindliche Datei "Kunden".

XOPEN

Dieser Befehl dient zum Einrichten oder Aktivieren einer wahlfreien Datei. Wahlfrei heißt, daß in diese Datei gleichzeitig und in beliebiger Reihenfolge sowohl gelesen als auch geschrieben werden kann. Dies wird durch einen speziellen Aufbau der Datei möglich. Diese besteht nämlich aus vielen Datenfeldern von 32 Byte Länge, die durch Angabe ihrer Feldnummer angesprochen werden. Jedes dieser Felder kann entweder eine Zahl oder einen String von bis zu 31 Zeichen Länge aufnehmen. Die Argumente des XOPEN-Befehls entsprechen denen von WOPEN.

Beispiel: XOPEN #2, FD1, "Telefon" aktiviert auf der Diskette in Laufwerk 1 die wahlfreie Datei Telefon. Falls diese nicht vorhanden ist, wird sie eingerichtet.

PRINT

Mit PRINT werden Daten in eine Datei geschrieben. Dabei muß zwischen sequentiellen und wahlfreien Dateien unterschieden werden. Bei sequentiellen Dateien folgen hinter PRINT, dem # und der Dateinummer durch Kommata getrennt ein oder mehrere Daten, Variablen, Zahlen oder Texte. Bei whlfreien Dateien liegt zwischen Dateinummer und erstem Datum noch die Nummer des Datenfeldes, in das das erste Datum geschrieben werden soll. PRINT/T wirkt in bekannter Weise auf Kassette.

Beispiel: PRINT #2, A$, N schreibt in die sequentielle Datei mit der Nummer 2 die Inhalte der Variablen A$ und N. Diese Daten werden an die vorhandenen angefügt. Die Datei muß mit WOPEN eingerichtet worden sein, da in sie ja geschrieben wird.
Beispiel: PRINT #9 (17), ZA, "Test" schreibt in das Datenfeld 17 der wahlfreien Datei mit der Dateinummer 9 den Inhalt der Variablen ZA und in das Datenfeld 18 den String "Test".

INPUT

INPUT ist das Gegenteil zu PRINT mit gleicher Syntax. Es wird das nächste Datum einer sequentiellen Datei bzw. ein beliebiges Datenfeld einer wahlfreien Datei in eine Variable eingelesen. Mit INPUT/T kann in üblicher Weise natürlich auch auf Kassette zugegriffen werden.

Beispiel: Mit INPUT #2, A$, B werden von der sequentiellen Datei mit der Nummer 2 ein String und eine Zahl eingelesen und den Variablen zugewiesen. Die Datei muß mit ROPEN aktiviert worden sein, da aus ihr gelesen wird. Es ist darauf zu achten, daß nicht Zahlen in Stringvariablen oder umgekehrt gelesen werden.
Beispiel: INPUT #9 (ZE), M, N liest aus dem Datenfeld mit der Nummer, die in der Variablen ZE steht, eine Zahl und weist sie der Variablen M zu. Die Zahl im nächsten Datenfeld wird der Variablen N zugewiesen.

CLOSE

Dieser Befehl schließt die Datei mit der angegebenen Nummer. Damit wird eine mit WOPEN eingerichtete Datei endgültig auf Diskette abgespeichert. Mit ROPEN oder XOPEN aktivierte Dateien werden deaktiviert. Der Befehl wirkt in der Form CLOSE/T auf die Kassette. Die Dateinummer ist damit frei für eine neue Datei. Wird keine Dateinummer spezifiziert, werden alle Dateien deaktiviert bzw. abgespeichert.

Beispiel: CLOSE #9 schließt die Datei mit der Nummer 9.

KILL

KILL löscht eine aktivierte oder eingerichtete Datei und gibt die dazu-

gehörende Dateinummer zu neuer Verwendung frei. Ohne die Angabe einer Nummer werden alle gerade aktivierten oder eingerichteten Dateien gelöscht. Dateien, die im Moment nicht angesprochen werden, d. h. keine Dateinummer haben, bleiben davon unberührt.

Beispiel: KILL #1 vernichtet die angefangene Datei mit der Nummer 1.

IF EOF(. .) THEN

Mit diesem Befehl ist es möglich, das Ende einer Datei festzustellen. Ist das Ende der Datei erreicht, daß heißt, das letzte Datum der Datei mit INPUT eingelesen, dann ist die Bedingung erfüllt und die Programmabarbeitung wird mit der hinter THEN angegebenen Zeilennummer fortgesetzt. Anderenfalls wird mit der nächsten Zeile weitergemacht.

Beispiel: IF EOF(#2) - THEN 9000 bewirkt einen Sprung zur Zeile 9000, falls das Ende der Datei mit der Nummer 2 erreicht ist.

Damit wären die Dateibefehle erklärt. WOPEN und ROPEN in Verbindung mit USR (Maschinenprogrammen) wurden hier absichtlich fortgelassen, da mit diesen Befehlen selbst der Profi Schwierigkeiten hat.

ON ERROR GOTO

Mit Hilfe dieses Befehls ist es möglich, Fehler, die während des Programmlaufs auftreten, vom Programm aus zu bearbeiten. Dieser Befehl sollte möglichst am Anfang eines Programms stehen und gilt bis zum nächsten ON ERROR GOTO, auf das der Interpreter stößt. Ergibt sich während des Programmlaufs jetzt ein Fehler, so wird das Programm nicht abgebrochen, sondern mit der hinter ON ERROR GOTO angegebenen Programmzeile fortgesetzt. Dort kann sich eine vom Benutzer geschriebene Routine um den Fehler kümmern.

Beispiel: ON ERROR GOTO 10000 setzt die Programmausführung mit Zeile 10000 fort, wenn ein Fehler auftritt.

IF ERN . . . THEN

Dieser Befehl dient der Abfrage der Fehlernummer. Wenn die Bedingung erfüllt ist, wird die Programmausführung mit dem Befehl oder der Zeilennummer hinter THEN fortgesetzt. Statt THEN Zeilennummer kann auch GOTO Zeilennummer stehen.

Beispiel: IF ERN = 5 THEN PRINT "String länger als 255 Zeichen!" : GOTO 10200 überprüft, ob die Fehlernummer 5 ist, wenn ja, wird ein Text angezeigt und mit Zeile 10200 fortgesetzt, wenn nein, wird mit der nächsten Zeile fortgesetzt.

IF ERL . . . THEN

Arbeitet völlig analog zu IF ERN ... THEN, nur daß hier die Zeile, in der der Fehler auftrat, überprüft wird.

RESUME

Mit Hilfe dieses Befehls wird die Weiterführung des Programms nach der Bearbeitung der Fehlerroutine gesteuert. RESUME allein setzt die Programmausführung mit dem Befehl, der den Fehler verursachte, fort. RESUME NEXT setzt mit dem Befehl direkt nach dem Befehl, der den Fehler bewirkte, fort. RESUME Zeilennummer setzt mit der angegebenen Zeilennummer fort und RESUME 0 setzt mit der ersten Zeile des Programms fort. RESUME bzw. RESUME NEXT ist besonders dann wichtig, wenn der Fehler in Schleifen oder Unterprogrammen aufgetreten ist, da sonst die Verschachtelung solcher Programmteile durcheinander gebracht wird.

Zum Schluß noch zum Programm "Utility" (auf Ihrer Disk-BASIC-Diskette). Damit können Sie Disketten formatieren und kopieren. Starten Sie dieses Programm. Es erscheint ein kleines Menu. I dient dem Formatieren von Disketten. Dies ist vor der ersten Benutzung jeder Diskette erforderlich! Gefragt wird dann nach dem Laufwerk, indem sich die zu formatierenden Diskette befindet. Falls diese Diskette nicht neu ist, werden sämtliche Informationen darauf unwiederbringlich gelöscht! Mit S machen Sie diese Diskette dann zu einer Slave-Diskette, d. h. eine Diskette, die nicht das Disk-BASIC enthält. Dazu müssen Sie das Laufwerk, in dem sich die eben formatierte Diskette befindet, und die schon angesprochene Diskettennummer eingeben (1-127). Diese Nummer bleibt dieser Diskette für immer zugeordnet, es sei denn, Sie formatieren sie neu. Mit B gelangen Sie ins BASIC zurück.

Der BASIC-Interpreter unterscheidet drei Arten von Disketten, Master-, Submaster- und Slavedisketten. Masterdisketten werden von Sharp geliefert und enthalten den Interpreter, der in einer speziellen Art aufgezeichnet worden ist, so daß er von der IPL geladen werden kann. Submasterdisketten sind Kopien von Masterdisketten, die sie sich mit Hilfe der Option C des Utility-Programms von Ihrer Masterdiskette erzeugen können. Es wird dann nach dem Quell- und dem Ziellaufwerk gefragt. Legen Sie die zu kopierende Diskette in Laufwerk 1 und eine leere Diskette in Laufwerk 2 (sollte sie nicht leer sein, werden die darauf enthaltenen Informationen gelöscht). Geben Sie dann eine 1 und eine 2 an. Das Kopieren dauert einige Zeit. Wenn Sie Submasterdisketten kopieren, erhalten Sie Slavedisketten. Slavedisketten enthalten keinen Interpreter, sondern nur Programme und Daten. Sie werden auch nach dem Initialisieren (=Formatieren) (Utility Option I) durch die Option S des Utilitys erzeugt. Auch Slavedisketten lassen sich mit Option C kopieren, Sie erhalten dann wieder Slavedisketten gleichen Inhalts.

Auf Ihrer Masterdiskette finden Sie auch ein Programm mit Namen Filing CMT. Damit haben Sie die (einzige) Möglichkeit, Maschinenprogramme von Kassette auf Diskette zu bringen. Dazu wird nach dem Start von Filing CMT (mit RUN) die Kassette richtig vorgespult eingelegt und die Nummer des Laufwerks, auf das das Programmm kopiert werden soll, angegeben.

Schließlich befindet sich auf Ihrer Masterdiskette das BASIC-Programm AUTO RUN. Dieses Programm wird automatisch nach jedem Urladen des Interpreters geladen und ausgeführt. In der vorliegenden Form definiert es die Funktionstasten. Sie können dieses Programm auf Ihren Bedarf anpassen oder ein beliebiges Programm AUTO RUN nennen, so daß es automatisch ausgeführt wird. Alle anderen Programme auf Ihrer Masterdiskette sind Demonstrationsprogramme mit den dazugehörenden Dateien.

x x x

DISK BASIC ERROR LIST & FILE COMMANDS

SHARP

■Error Message Table

Error No.	Meaning
1	Syntax error.
2	Operation result overflow.
3	Illegal data.
4	Data type mismatch.
5	String length exceeded 255 characters.
6	Insufficient memory capacity.
7	The size of an array defined was larger than that defined previously.
8	The length of a BASIC text line was too long.
10	The number of levels of GOSUB nests exceeded 16.
11	The number of levels of FOR-NEXT nests exceeded 16.
12	The number of levels of functions exceeded 6.
13	NEXT was used without a corresponding FOR.
14	RETURN was used without a corresponding GOSUB.
15	Undefined function was used.
16	Unused reference line number was specified in a statement.
17	CONT command cannot be executed.
18	A writing statement was issued to the BASIC control area.
19	Direct mode commands and statements are mixed together.
20	RESUME statement cannot be executed.
21	A RESUME statement was used without a corresponding error process.
24	A READ statement was used without a corresponding DATA statement.
25	The number of SWAP levels exceeded 1.
40	File was not found.
41	Disk drive hardware error.
42	A file name already used was defined again.
43	OPEN, DELETE, RENAME statements were issued to an open file.
44	A file which is not open was referenced or CLOSE or KILL statement was issued to it.
46	A protected file was accessed for writing.
50	The disk drive is not ready.
51	The total number of files on a volume exceeded 63.
52	Volume number error.
53	File space on the diskette is insufficient.
54	A diskette which has not been initialized was loaded.
55	The number of data of a BSD file exceeded 64K bytes.
56	Data error occurred on an FDC routine call.
57	The diskette cannot be used.
60	Illegal file name was specified.
61	Illegal file mode was specified.
63	Out of file (CMT).
64	Illegal logical number was specified.
65	The printer is not ready.
66	Printer hardware error.
67	Out of paper.
70	Check sum error.

■ File mode

File mode	Disk file symbol	Cassette file symbol
BASIC text file	BTX	2
Sequential access file	BSD	3
Random access file	BRD	—
Object file	OBJ	1

When the file directory is displayed, symbols of all locked files are followed by an asterisk (*) to indicate that the file is write protected.

■ Summary of control commands and statements

(1) Disk file

File mode	File writing	File reading
BASIC text file	SAVE FDd@v, "filename"	LOAD FDd@v, "filename" RUN FDd@v, "filename" CHAIN FDd@v, "filename" SWAP FDd@v, "filename"
Sequential access file	WOPEN #n, FDd@v, "filename" PRINT #n, data CLOSE #n KILL #n (Cancels WOPEN #).	ROPEN #n, FDd@v, "filename" INPUT #n, variable CLOSE #n IF EOF(#n) THEN (Checks for file end).
Random access file	XOPEN #n, FDd@v, "filename" PRINT #n (expression), data CLOSE #n	XOPEN #n, FDd@v, "filename" INPUT #n (expression), variable CLOSE #n IF EOF(#n) THEN (Checks for file end).
Object file		RUN FDd@v, "filename" (Loads and executes program "filename"). LIMIT X (or $XXXX) LOAD FDd@v, "filename" (When linked with the BASIC interpreter).

- FDd and/or @v may be omitted in some cases.
- Any number from 1 through 127 can be used as a logical file number and up to 10 logical files can be opened simultaneously.
- The following file control statements can be executed on files of any mode.

  LOCK ····· Write protects the file.
  UNLOCK ····· Resets write protect.
  RENAME ····· Changes the file name. (Invalid for a locked file).
  DELETE ····· Deletes the file. (Invalid for a locked file).

(2) Cassette file

File mode	File writing	File reading
BASIC text file	SAVE/T "filename"	LOAD/T "filename"
Sequential access file	WOPEN/T "filename" PRINT/T data CLOSE/T	ROPEN/T "filename" INPUT/T variable CLOSE/T

■ File mode

File mode	Disk file symbol	Cassette file symbol
BASIC text file	BTX	2
Sequential access file	BSD	3
Random access file	BRD	—
Object file	OBJ	1

When the file directory is displayed, symbols of all locked files are followed by an asterisk (*) to indicate that the file is write protected.

■ Summary of control commands and statements

(1) Disk file

File mode	File writing	File reading
BASIC text file	SAVE FDd@v, "filename"	LOAD FDd@v, "filename" RUN FDd@v, "filename" CHAIN FDd@v, "filename" SWAP FDd@v, "filename"
Sequential access file	WOPEN #n, FDd@v, "filename" PRINT #n, data CLOSE #n KILL #n (Cancels WOPEN #).	ROPEN #n, FDd@v, "filename" INPUT #n, variable CLOSE #n IF EOF(#n) THEN (Checks for file end).
Random access file	XOPEN #n, FDd@v, "filename" PRINT #n (expression), data CLOSE #n	XOPEN #n, FDd@v, "filename" INPUT #n (expression), variable CLOSE #n IF EOF(#n) THEN (Checks for file end).
Object file		RUN FDd@v, "filename" (Loads and executes program "filename"). LIMIT X (or $XXXX) LOAD FDd@v, "filename" (When linked with the BASIC interpreter).

- FDd and/or @v may be omitted in some cases.
- Any number from 1 through 127 can be used as a logical file number and up to 10 logical files can be opened simultaneously.
- The following file control statements can be executed on files of any mode.

  LOCK ····· Write protects the file.
  UNLOCK ····· Resets write protect.
  RENAME ····· Changes the file name. (Invalid for a locked file).
  DELETE ····· Deletes the file. (Invalid for a locked file).

(2) Cassette file

File mode	File writing	File reading
BASIC text file	SAVE/T "filename"	LOAD/T "filename"
Sequential access file	WOPEN/T "filename" PRINT/T data CLOSE/T	ROPEN/T "filename" INPUT/T variable CLOSE/T