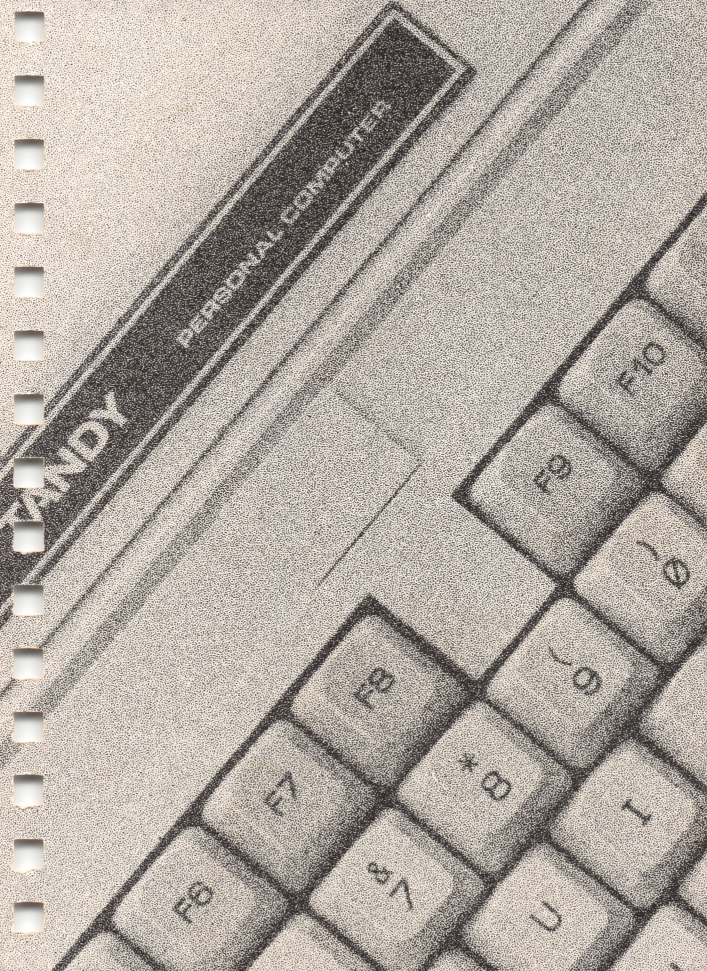
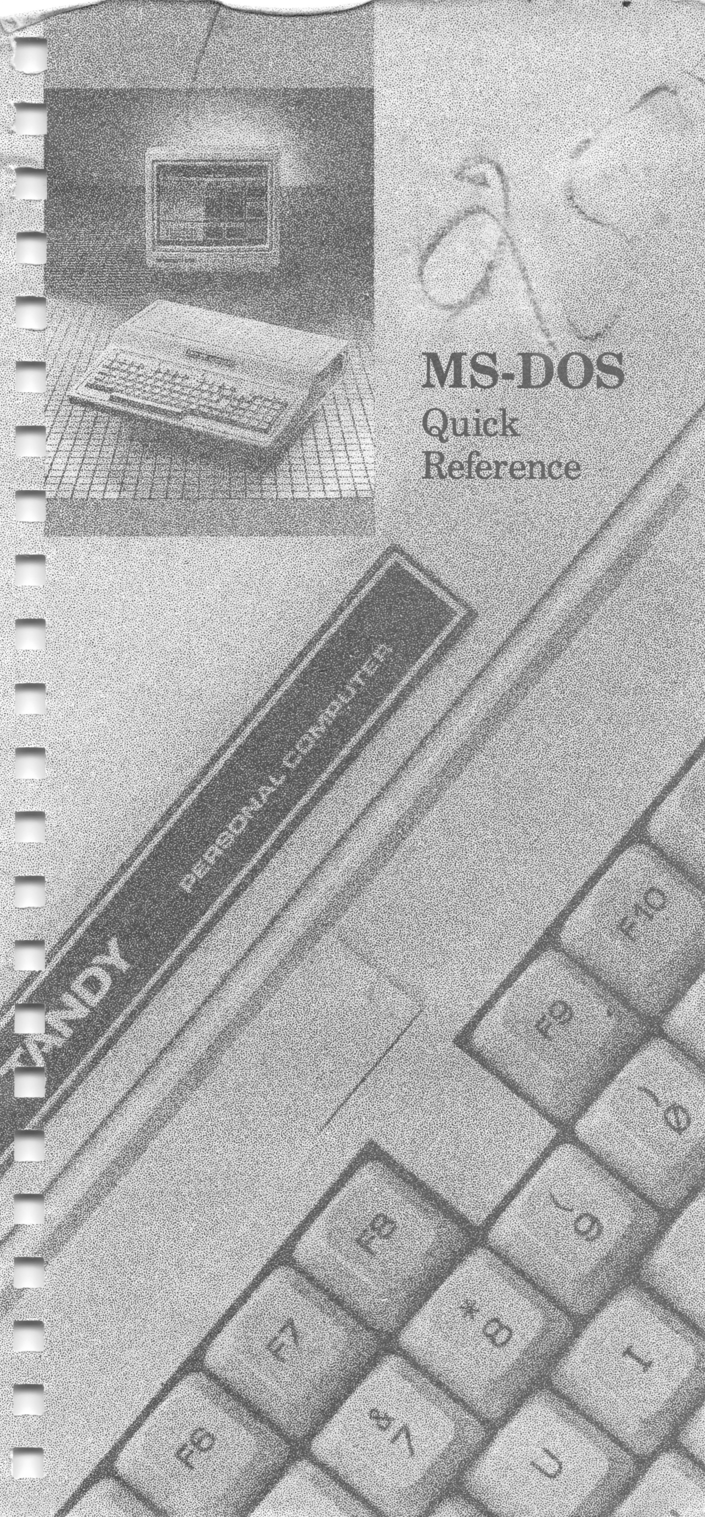
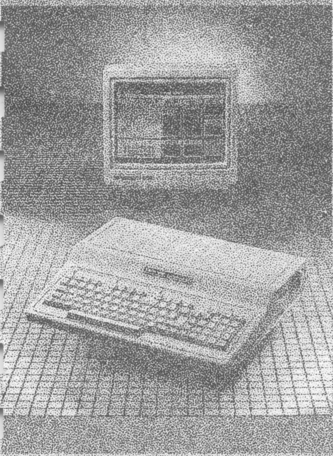
TANDY®
1000 EX
Quick
Reference



10 9 8 7 6 5 4 3 2





MS-DOS
Quick
Reference

CONTENTS

LOADING MS-DOS

1. Turn on your computer.

2. Insert the MS-DOS system diskette into Drive A. Turn the drive lever clockwise to close the disk drive.

3. When the prompt appears, enter the current date in the *mm-dd-yyyy* format. (For example, type `9-7-86` or `Ø9-Ø7-1986`, and press [ENTER] for September 7, 1986.)

4. When it appears, press [ENTER] to skip the time prompt, or enter the time in the *hh:mm:ss.cc*, 24-hour format. (For example, type `14:3Ø`, and press [ENTER] for 2:30 p.m.)

   MS-DOS displays the system prompt:

   ```
   A>
   ```

   Your computer is now ready for use.

MS-DOS COMMANDS

Notation:

BOLD UPPERCASE indicates a command name. (Type a command exactly as it appears.)

lowercase italics represent variable words, letters, characters, or values

UPPERCASE indicates information you type exactly as it appears.

[ ] (square brackets) indicate optional parameters.

... (ellipsis) indicates that you can repeat a parameter.

Also, certain commands are valid only for a specific version of MS-DOS or apply only to a hard disk. These commands are indicated as follows:

2.11	MS-DOS V 2.11 command	3.2	MS-DOS V 3.2 command	HD	MS-DOS hard disk command

If you are not sure which version of the operating system your computer uses, type `VER` [ENTER].

APPEND [;] [*pathname* [;*pathname*] . . .]

3.2 (Internal) Sets a data file path, telling MS-DOS in which drives and directories to search for data files. APPEND ; sets the NUL data path, which searches only the current directory. APPEND with no parameters displays the current data path.

```
append b:\sales\region1;a:
```

ASSIGN [*drive1* = *drive 2* . . .]

3.2 (External) Reassigns drive letters so that requests for one drive are routed to another.

drive1 is the drive letter to reassign.

drive2 is the drive letter to be given to *drive1*.

```
assign a=c b=c
```

ATTRIB [*read*] [*archive*] *pathname*

(External) Sets the read-only and archive attributes of the file specified by *pathname*. Displays the attributes of the file if you omit the *read* and *archive* parameters.

read can be:

+R	sets the read-only mode.
−R	disables the read-only mode.

archive can be:

+A	sets the archive attribute.
−A	clears the archive attribute.

```
attrib +r b:\mydir\myfile.txt
```

BACKUP [*pathname*] *drive*: [/S] [/M] [/A] [/D:*date*]

(External) Copies one or more files from a hard disk to floppy diskettes.

pathname specifies the file to backup.

drive: specifies the disk to receive the files.

/S	copies all files in the specified directory and all files below it.
/M	copies only files modified since the last backup.
/A	adds the files to the diskette in the specified drive.
/D:*date*	copies only those files created or modified on or after the specified date (*mm-dd-yy*).

```
backup c:store1\sales.dat a:/a
```

BACKUP [*pathname*] *drive*: [/S] [/M] [/A] [/P] [/D:*mm-dd-yy*] [/T:*hh*:*mmx*] [/L:*filename*]

(External) Copies one or more files from a hard disk to floppy diskettes.

pathname specifies the file to backup.

drive: specifies the disk to receive the files.

/S	copies all files in the specified directory and all files below it.
/M	copies only files modified since the last backup.
/A	adds the files to the diskette in the specified drive.
/P	saves the file in a packed format.
/D	copies only those files created or modified on or after the specified date.
/T	copies only those files created or modified on or after the specified time. *hh* can be 1-12. *mm* can be 0-59. *x* indicates either a.m. (a) or p.m. (p).
/L:*filename*	creates a backup log entry in the file specified.

```
backup c: a: /l
```

BREAK [*switch*]

(Internal) Turns on and off the CTRL C check. Displays the current CTRL C setting if all parameters are omitted.

switch can be ON or OFF.

```
break on
```

CHDIR *pathname*
CD *pathname*

(Internal) Changes the current (home) directory of the specified drive to the directory specified by *pathname*. Displays the current directory if you omit the *pathname*.

```
chdir \bin\user      cd b:
```

CHKDSK [*pathname1*] [/F] [/V] [>*pathname2*]

(External) Checks the directory of the Tandy MS-DOS diskette or hard disk in the current or specified *drive* for errors.

pathname1 specifies either an entire drive or an individual file to be checked. If you specify a file, CHKDSK displays information about both the drive and the file.

/F fixes errors where possible and updates the disk. (Do not specify *pathname2* with /F.)

/V displays messages and error details while CHKDSK is running.

pathname2 specifies the file to which CHKDSK redirects its output. (Do not use /F with *pathname2*.)

```
chkdsk a: >b:\sales\errors
```

CLS

(Internal) Clears the screen.

```
cls
```

COMMAND [*pathname*] [*device*] [/E:*size*] [/P] [/C *string*]

3.2

(Internal) Starts a new command processor.

pathname specifies in which drive and directories the command processor looks for the COMMAND.COM file if it needs to reload the transient portion of the file into memory.

device specifies a different device for input and output:

AUX an auxilliary device, usually the RS-232 serial port 1.

COM1 COM2 the RS-232 serial ports 1 and 2.

CON the console (keyboard input, screen output).

E:*size* specifies the environment size. The *size* is a number, in bytes, in the range 128-32768. 128 is the default.

/P tells the command processor not to exit to a higher level.

/C *string* tells the command processor first to execute the command or commands specified by *string*, then to return. The /C switch is valid only as the last parameter.

```
command b:\bin /c chkdsk a:
```

COPY *source pathname* [*target pathname*] [/A] [/B] [/V]

(Internal) Copies one or more files to the same directory as the *source* (giving them different filenames) or to another directory (giving them the same or different filenames). To leave the filename the same, omit the filename from the *target pathname*. If you omit the /A and /B parameters, COPY uses /B.

/A source file: treats the file as an ASCII text or data) file.
target file: adds an EOF character to the end of the file.

/B source file: treats the file as a binary (program) file.
target file: does not add an EOF character to the end of the file.

/V verifies the sectors written to disk.

```
copy memo.txt /a corr.txt
copy b:sales/memo.txt memo.bak
```

COPY *target pathname* + *source pathname1* [+ *source pathname2* . . .] [/A] [/B] [/V]

(Internal) Appends one or more *source* files to the *target* file. If you omit the /A and /B parameters, COPY uses /A.

/A source file: treats the file as an ASCII file (text or data) file.
target file: adds an EOF character to the end of the file.

/B source file: treats the file as a binary (program) file.
target file: does not add an EOF character to the end of the file.

/V verifies the sectors written to disk.

```
copy b:print.dat + read.dat +
  write.dat
```

COPY *source pathname1* [+ *source pathname2* . . .]
target pathname [/A] [/B] [/V]

(Internal) Combines any number of *source* files into a new *target* file. If you omit the /A and /B parameters, COPY uses /A.

/A	source file: treats the file as an ASCII file (text or data) file. target file: adds an EOF character to the end of the file.
/B	source file: treats the file as a binary (program) file. target file: does not add an EOF character to the end of the file.
/V	verifies the sectors written to disk.

```
copy memos.txt + memos.txt
  b:letters.txt
```

COPYDOS [p]

2.11

(External) Copies operating system files onto an application program diskette.

p	copies the files without erasing any data previously stored on the diskette. The p parameter must be in lowercase.

```
copydos p
```

CTTY *device*

(Internal) Changes the I/O *device*.

device can be:

AUX	an auxiliary device, such as the RS-232 serial port 1.
COM1 COM2	both RS-232 serial ports, 1 and 2.
CON	the console (keyboard input, screen output).

```
ctty aux
```

DATE *[mm-dd-yyyy]*

(Internal) Enters or changes the system date. Displays the current date if you omit the date parameter.

mm-dd-yyyy specifies the month, day, and year to set as the date.

```
date 11-15-1986          date 11-15-86
```

DEL

See ERASE.

DIR *[pathname]* [/P] [/W]

(Internal) Displays information about: (1) all files in the current directory, (2) all files on the drive or in the directory specified by *pathname*, or (3) one file specified by *pathname*.

/P	selects the page mode.
/W	selects a wide display.

```
dir b:     dir \user\*.bat /p
```

DISKCOMP *[drive1:] [drive2:] [/1] [/8]*

(External) Compares the contents of two diskettes.

drive1: is the drive containing the source diskette.

drive2: is the drive containing the target diskette.

/1	compares only the first side of the diskettes, even if both are double-sided. If you omit this parameter, DISKCOMP compares both sides.
/8	compares only the first 8 sectors of each track. If you omit this parameter, DISKCOMP compares 9 sectors.

```
diskcomp a: b:
```

DISKCOPY [*source drive*:] [*target drive*:]

(External) Copies the contents of the diskette in the *source drive* to the disk in the *target drive*. DISKCOPY formats the target diskette if it is not the same format as the source diskette.

```
diskcopy        diskcopy a: b:
```

DISKTYPE [*drive*:]

(External) Displays size and capacity information for the specified drive.

```
disktype c:
```

ECHO [*switch*] [*message*]

(Internal) Turns on or off the batch ECHO feature, displays a message, or displays the current ECHO setting.

switch can be OFF or ON.

message is a batch file message you want to print on the screen.

```
echo off
echo Insert diskette.
```

ERASE *pathname*
DEL *pathname*

(Internal) Erases (deletes) one or more files from the current directory or the directory specified by *pathname*. If *pathname* does not include a filename, ERASE deletes **all** files in the specified directory.

```
erase b:\sales\region1
del b:\sales\region1\joe-sales
```

EXE2BIN *source pathname* [*target pathname*]

(External) Converts an executable (.exe) file to a binary file (.bin) format.

source pathname specifies the executable file.

target pathname specifies a new binary format file to receive the converted file. The *source* filename, with a .bin extension, is used for the new file if you omit the *target pathname*.

```
exe2bin testfile.exe b:
```

EXIT

(Internal) Exits the command processor and returns to a previous level, if one exists.

```
exit
```

FC [/B] [/C] [/W] [/*number*] *pathname1 pathname2* [>*target pathname*]

2.11

(External) Compares the contents of the two files specified by *pathname1* and *pathname2*. FC sends the output to the screen or to the file specified by *target pathname*.

/B	forces a binary comparison of the files.
/C	interprets all letters in the files as uppercase. Do not use with /B.
/W	compresses tabs and spaces. Do not use with /B.
/*number*	specifies the number of lines (1-9) that must match for the file to be considered as matching again after FC finds a difference. The default is 3 lines. Do not use with /B.

```
fc /b test1.src test2.src
   >test3.src
```

FC [/A] [/B] [/C] [/L] [/LB *n*] [/N] [/T] [/W] [*/number*] *pathname1 pathname2* [>*target pathname*]

(External) Compares the contents of the two files specified by *pathname1* and *pathname2*. FC sends the output to the screen or to the file specified by *target pathname*.

/A	abbreviates the output of an ASCII comparison. This option displays only the first and last lines in each block of different lines.
/B	forces a binary comparison of the files. This is the default when comparing .BIN, .COM, .EXE, .LIB, .OBJ, or .SYS. files
/C	interprets all letters in the files as uppercase. Do not use with /B.
/L	compares the files in ASCII mode. This is the default when comparing all files except .BIN, .COM, .EXE, .LIB, .OBJ, and .SYS.
/LB *n*	sets the internal line buffer to *n* lines. The default is 100 lines.
/N	displays the line numbers in an ASCII comparison.
/T	does not expand tabs to spaces. The default is to treat tabs as eight spaces.
/W	compresses tabs and spaces. Do not use with /B.
/number	specifies the number of lines (1-9) that must match for the file to be considered as matching again after FC finds a difference. The default is 3 lines. Do not use with /B.

```
fc /n test1.src test2.src
   >test3.src
fc \user\working\file.txt
   \user\backup\file.txt
```

FDISK

(External) Displays or modifies hard disk partitions.

```
fdisk
```

FIND [/V] [/C] [/N] "string" [*pathname* . . .]

(External) Searches for the specified *string* of text in one or more files, specified by *pathname(s)*. Searches for *string* in data from the standard input device if you omit *pathname*.

/V	displays all lines that do not contain *string*.
/C	displays only the count of lines in each file that contain *string*.
/N	displays the relative line number along with the lines that contain *string*. Do not use with /C.

```
find /n "mispell" datafile.txt
```

FOR %*c* IN (*set*) DO *command*
(regular command)
FOR %%*c* IN (*set*) DO *command*
(batch file command)

(Internal) Executes the specified *command* once for each item in the *set*.

set	is either a list of items separated by spaces or one wildcard item.
c	can be any one-character variable except 0 to 9.
command	is the command to be executed. If you include %*c* or %%*c* at the end of the *command*, MS-DOS sequentially substitutes each member of *set* in the *command*.

```
for %f in (taxfile autofile
  homefile) do del %f
```

FORMAT [*drive*:] [/V] [/1] [/8] [/B] [/S]

2.11

(External) Formats the blank floppy diskette in the specified *drive* to prepare it for use.

/V	prompts for a volume label. Do not use with /B.
/1	formats a single-sided diskette. The default is double-sided. (The /1 switch is not valid for a 3 1/2-inch microfloppy diskette format.)
/8	formats 8 sectors per track. The default is 9 sectors per track. (The /8 switch is not valid for a 3 1/2-inch microfloppy diskette format.)
/B	formats 8 sectors per track and allocates space for the hidden system files. (The /B switch is not valid for a 3 1/2-inch microfloppy diskette format.)
/S	copies the system files from the default drive to the newly formatted diskette. (/S must be the last parameter.) Do not use with /B.

```
format b: /b     format b: /v /s
```

FORMAT *drive*: [/V] [/1] [/8] [/B] [/S]

3.2

(External) Formats the blank floppy diskette in the specified *drive* to prepare it for use, or formats the hard disk specified by *drive*.

/V	prompts for a volume label. Do not use with /B. The /V switch is valid for both a floppy disk format and a hard disk format.
/1	formats a single-sided diskette. (This parameter is valid only when formatting a floppy diskette.)
/8	formats 8 sectors per track. (This parameter is valid only when formatting a floppy diskette.)
/B	formats 8 sectors per track and allocates space for the hidden system files. (This parameter is valid only when formatting a floppy diskette.) Do not with /S or /V.
/S	copies the system files from the default drive to the newly formatted disk. Do not use with /B. The /S switch is valid for both a floppy disk format and a hard disk format.

```
format c: /v /s     format b: /b
```

GOTO *label*

(Internal) Used in a batch file to transfer execution to the next line after the line that contains *:label*.

label is a character string.

```
:g
rem looping...
goto g
```

GRAPHICS *ptype* [/R] [/B] [/CR] [/LF]

(External) Enables SHIFT PRINT to reproduce a graphics screen in color on the Tandy CGP-220 printer or in shades of black and white on other printers.

ptype is one of these printer types:

CGP220 specifies the Tandy CGP-220.

DMP110 specifies the Tandy DMP-110.

PCMODE specifies a Tandy printer with a DIP switch set for the PC mode. Also for other PC-compatible printers.

TMODE specifies a Tandy printer with a DIP switch set for the Tandy mode.

STANDARD specifies any other Tandy printer.

/R	prints black as black and white as white. (Do not use with a CGP-220 printer.)
/B	prints the background color as black. (Use only with a CGP-220 printer.)
/CR	makes the end-of-line character a carriage return only.
/LF	makes the end-of-line character a line feed only.

```
graphics cgp220 /b
```

HFORMAT [*drive*:] [/S] [/V] [/B]

(External) Prepares a hard disk partition for use.

drive	is the hard disk drive (C: or D:).
/S	copies the system files to the disk.
/V	prompts for a volume label.
/B	issues prompts before locking out flawed sectors.

```
hformat /s        hformat D: /V
```

HSECT [*drive*:]

2.11 HD

(External) Formats hard sectors on a hard disk.

```
hsect d:
```

HSECT

3.2 HD

(External) Formats track and sector information on a hard disk. HSECT prompts for the drive, number of heads, number of cylinders, interleave factor, and bad track information.

```
hsect
```

IF [NOT] *condition command*

(Internal) Allows conditional execution of commands in batch file processing.

NOT executes the *command* only when the *condition* is false.

conditions are:

ERRORLEVEL *number* executes the *command* only if the program previously executed by COMMAND has an exit code of *number* or higher.

string1 = = *string2* executes the *command* only if *string1* and *string2* are identical after parameter substitution.

EXIST *filename* executes the *command* only if *filename* exists.

```
if exist memo.txt goto g
```

JOIN [*drive*:] [*pathname*] [/D]

3.2

(External) Links the root of *drive* to the *pathname* specified. Displays the current JOIN status if all parameters are omitted.

drive: is the drive name you are joining.

pathname is the empty path, including the drive, to which the *drive* is joined.

/D turns off a previous JOIN command.

```
join b: c:\sales
join b: /d
```

KEYTXX [/US]

(External) Replaces the current ROM BIOS keyboard program with the international keyboard program specified by *XX*. *XX* can be FR (France), GR (Germany), IT (Italy), SP (Spain), or UK (United Kingdom). To return to the U.S. keyboard, press CTRL SHIFT F1. (Use the left SHIFT key.)

/US converts character scan codes to US scan codes.

```
keytuk /us
```

LABEL *drive*:[*label*]

(External) Creates, changes, or deletes a volume label. Omit the label to delete the existing label for the specified drive.

drive: is the disk whose volume label you want to modify.

label is the new volume label.

```
label a:mydisk
```

LF

(External) Suppresses the line feed after a carriage return in printer output.

```
lf
```

LPSETUP *switch, ptype, port* [, *page*]

(External) Enables a printer filter that allows pagination.

switch is the filter status:

ON enables the printer pagination filter.

OFF disables the printer pagination filter. Do not use any other parameters with OFF.

ptype is the printer type:

TANDY is a Tandy printer that is not in IBM mode.

IBM is a Tandy or other printer in IBM mode.

port is the parallel printer port the filter uses (1, 2, 3, or 4).

page is the starting page number, in the range 1-65535, of the printout. The default page number is 1.

```
lpsetup on, tandy, 1, 100
lpsetup off
```

MKDIR *pathname*
MD *pathname*

(Internal) Makes a new directory.

pathname tells MS-DOS the directory under which to create the new directory and specifies the name to give it.

```
mkdir \user           md b:\letters
```

MLFORMAT *drive*

(External) Formats a hard disk DOS2 partition created previously using MLPART. To access the partition, you need to install the MLPART.SYS device driver.

drive is the logical drive letter that refers to the DOS2 partition to format. This letter is automatically assigned and displayed when the MLPART.SYS device driver is installed during the boot procedure.

```
mlformat e:
```

MLPART

(External) Creates, changes, deletes, or displays non-bootable DOS2 hard disk partitions. This command is for use with hard disks that have a capacity of more than 32 megabytes.

```
mlpart
```

MODE *characters* [*shift* [T]]

(External) Shifts the video screen the specified number of *characters*. (Each *character* equals two characters in a 80 column screen and one character in a 40 column screen.)

shift	is the direction of the shift, either R or L.
T	creates a test screen.

```
mode 40 r t          mode 1
```

MODE *linefeed*

(External) Turns the printer line feed on or off. *Linefeed* can be either LFON or LFOFF.

```
mode lfon       mode lfoff
```

MODE *printer*

(External) Sets the printer type. *Printer* can be either DMP (dot matrix), DWP (daisy wheel), or NL (reset).

```
mode dwp
```

MODE *scan*

(External) Sets video scan lines. *Scan* can be either 200 or 225.

```
mode 225
```

MODE [*video*] [*characters*]

(External) Sets the video mode and characters per line. *Video* can be BW (black and white) or CO (color). *Characters* can be 40 or 80.

```
mode co 80
```

MODE COLORMAP [*oldcolor newcolor*]

(External) Changes the video palette color specified by *oldcolor* to the color specified by *newcolor*. The colors available are: black, blue, green, cyan, red, magenta, yellow, gray, dark gray, light blue, light green, light cyan, light red, light magenta, light yellow, and white. If you omit the parameters, all colors are reset to their original colors.

```
mode colormap black blue
```

MODE COM*number:* [*baud*] [*parity*] [*databits*] [*stopbits*] [P]

(External) Sets the RS232 parameters.

number: is the RS232 serial port number, either 1 or 2.

baud is the baud rate: 110, 150, 300, 600, 1200, 2400, 4800, or 9600. The default is 300.

parity can be: N (no parity), E (even parity), or O (odd parity). The default is E.

databit can be either 7 or 8. The default is 7.

stopbit can be either 1 or 2. The default is 1.

P specifies that a serial printer is using the RS232 port and tells the printer driver to continuously retry to output on timeout errors.

```
mode com1: 1200 n 8 2 p
```

MODE FAST

(External) Sets the CPU speed to 7.16 MHz.

```
mode fast
```

MODE LPT1: *characters*[/*type*] [,P]

(External) Sets printer characters per line.

characters is the characters per line, either 80 or 132.

/type is the printer type: DMP for dot matrix printers or PC for PC-compatible printers. The default is DMP.

P tells the printer driver to continuously retry to output on timeout errors.

```
mode lpt1: 132/pc,p
```

MODE LPT1: *timeout*

(External) Sets the line printer timeout delay.

timeout is the timeout delay: LONG (2 minutes) or SHORT (45 seconds).

```
mode lpt1: short
```

MODE LPT1:=COM*serial*

(External) Redirects the printer output from the specified parallel port to the specified RS232 port. (Initialize the RS232 port with the MODE COM command before redirecting printer output.)

serial is the RS232 port number (1 or 2).

```
mode lpt1:=com1
```

MODE MONO [*switch*]

(External) Sets up the computer to use color-oriented software with a monochrome monitor. Software color requests are translated to black, white, and intense white. If you omit the *switch*, MODE MONO asks if you have a 350 scan line or a 200 scan line monitor. (Choose 350 only if your computer supports an optional video board and you install a Mono/Text video board.)

switch can be:

ON	turns on MODE MONO.
OFF	turns off MODE MONO.
200	turns on MODE MONO.
350	turns on MODE MONO and changes the active video adapter to Mono/Text.

```
mode mono on
```

MODE SLOW

(External) Sets the CPU speed to 4.77 MHz.

```
mode slow
```

MODE TV

(External) Sets up the computer to use a color TV. (Sets 200 video scan lines, color video mode, and 40 characters per line.)

```
mode tv
```

MORE

(External) Reads from standard input and displays one screen of information at a time, with the message, -MORE-, at the bottom.

```
type b:acctspay.dat | more
```

PATCH *pathname,address,data1,data2*

(External) Lets you make minor modifications to a disk file.

pathname is the file you want to change.

address is the byte number starting (hexadecimal) address of the data string to change.

data1 is the hexadecimal data value to change.

data2 is the hexadecimal data value to replace *data1.*

```
patch b:progx.com,16a3,000a106c2f,
   010b116d30
```

PATH [;] [*pathname*[;*pathname* . . .]]

(Internal) Sets a command path, telling MS-DOS in which drives or directories to search for external commands. PATH ; sets no path and searches only the current directory. If you omit all parameters, PATH displays the current path setting.

pathname specifies a drive or a directory.

```
path \bin\user\joe;b:\bin\user\joe
```

PAUSE [*message*]

(Internal) Suspends execution of a batch file.

message is a message to be displayed when the file pauses.

```
pause Insert diskette.
```

PRINT [*pathname* [/C] [/P] . . .] [/T]

2.11 C (External) Puts as many as ten files in the print queue for background printing. If you omit all parameters, PRINT displays the contents of the print queue.

pathname is the name of the file you want to print.

/C deletes (cancels) from the print queue the file that immediately precedes and all files that follow /C in the command line.

/P adds to the print queue (prints) the file that immediately precedes and all files that follow /P in the command line.

/T deletes (terminates) all files from the print queue. Do not use /T with a *pathname*.

```
print /t
print temp1.txt /c
   temp2.tst /p temp3.tst
```

PRINT [*pathname* [/D:*device*] [/B:*size*] [/Q:*value*] [/C] [/P] . . .] [/T]

(External) Puts as many as ten files in the print queue for background printing. If you omit all parameters, PRINT displays the contents of the print queue.

pathname is the name of the file you want to print.

/D:*device* specifies the print device. LPT1 is the default.

/B:*size* sets the size (in bytes) of the internal buffer.

/Q:*value* sets the number of files (4-32) allowed in the print queue. The default is 10.

/C deletes (cancels) from the print queue the file that immediately precedes and all files that follow /C in the command line.

/P adds to the print queue (prints) the file that immediately precedes and all files that follow /P in the command line.

/T deletes (terminates) all files from the print queue. Do not use /T with a *pathname*.

```
print /t
print temp1.txt /c
   temp2.tst /p temp3.tst
```

PROMPT *[text]*

(Internal) Changes the system prompt to *text*. Sets the prompt to the current drive specification if you omit *text*.

text is a string of characters, a special prompt, or a combination of the two. Special prompts can be:

t	current time
d	current date
p	current directory
v	MS-DOS version number
n	current drive
g	> symbol
l	< symbol
b	| symbol
—	carriage return and line feed
q	= symbol
b	backspace
e	escape sequence

Precede a special prompt with the $ character.

```
prompt $n$g
```

RECOVER *pathname*
RECOVER *drive*:

(External) Recovers a file that contains bad sectors, specified by *pathname*, or all files on a disk that contains bad sectors in its directory, specified by *drive*.

```
recover oldbook.txt     recover b:
```

REM *remark*

(Internal) Includes the specified *remark* in a batch file.

```
rem This file is called
  billfile.bat
```

REN *pathname filename*

(Internal) Changes the name of the file specified by *pathname* to *filename*.

```
ren b:\sales\region1\joe-sls bob-
   sls
```

REPLACE *source pathname* [*target pathname*] [/A] [/D] [/P] [/R] [/S] [/W]

(External) Updates previous versions of files.

source pathname is the drive or directory that contains the replacement files. The *source pathname* can also be a single file or a wildcard filename.

target pathname is the drive or directory that contains the files you want to replace.

/A	adds files that exist in the *source* directory, but not in the *target* directory, to the *target* directory. Do not use /A with /D.
/D	replaces files in the *target* directory with *source* files only if the the *source* files are newer than the corresponding *target* files. Do not use /D with /A.
/P	prompts before replacing a *target* file or adding a *source* file.
/R	replaces read-only files as well as unprotected files.
/S	searches all subdirectories of the *target* directory while replacing matching files. Do not use /S with /A.
/W	waits for you to press any key before replacing files.

```
replace a:\phones.cli c:\ /s
```

RESTORE *source drive*: [*target drive*:] *pathname* [/S] [/P]

(External) Restores one or more files from diskettes to a hard disk. The files must have been stored on diskette with the BACKUP command.

source drive: specifies the drive that contains the backup diskette.

target drive: specifies the hard disk drive to which you want to restore.

pathname specifies the hard disk directories and/or file you want to restore.

/S restores the directory specified by *pathname* as well as its subdirectories.

/P prompts before restoring read-only files and before restoring any files that have been changed since the last backup.

```
restore a: c:*.dat /p
```

RESTORE *source drive*: [*target drive*:] *pathname* [/S] [/P] [/B:*date*] [/A:*date*] [/E:*time*] [/L:*time*] [/M] [/N]

(External) Restores one or more files from the *source drive* disk to the *pathname* and the *target drive* specified. You can specify either a floppy disk drive or a hard disk drive for the *source drive* and for the *target drive*. The source files must have been stored with the BACKUP command.

source drive: specifies the disk drive that contains the backup files.

target drive: specifies the disk drive to which you want to restore.

pathname specifies the disk directories and/or file you want to restore.

/S restores the directory specified by *pathname* as well as its subdirectories.

/P prompts before restoring hidden or read-only files and before restoring any files that have been changed since the last backup.

/B:*date* restores only those files that were last modified on or before the given *date* (*mm-dd-yy*).

/A:*date* restores only those files that were last modified on or after the given *date* (*mm-dd-yy*).

/E:*time* restores only those files that were last modified at or earlier than the given *time* (*hh:mm*).

/L:*time* restores only those files that were last modified at or later than the given *time* (*hh:mm*).

/M restores only those files that have been modified since the last backup.

/N restores only those files that no longer exist on the *target drive*.

```
    restore a: c:*.dat /n
```

RMDIR *pathname*
RD *pathname*

(Internal) Removes the subdirectory specified by *pathname* from the specified disk.

```
    rmdir \bin\user\jim
```

SELECT *country* [*keyboard* [/US]]

3.2 (External) Changes the current country code or creates an internationally configured backup MS-DOS diskette.

country is the country code that selects the date and time format, the currency symbol, and the decimal separator.

keyboard is a two-character identifier of the keyboard layout. If you specify a *keyboard* code, a new MS-DOS diskette is created.

/US specifies U.S. scan codes, necessary for some application programs that are configured for a U.S. keyboard. Use this switch **only** along with the *keyboard* parameter.

The *country* and *keyboard* codes can be:

	Ctry	**Kbrd**		**Ctry**	**Kbrd**
Belgium	032	FR	Australia	061	US
France	033	FR	French Canada	002	US
Germany	049	GR	Israel	972	US
			Middle East	785	US
Italy	039	IT	United States	001	US
Spain	034	SP			
Portugal	351	SP	Denmark	045	**
			Finland	358	**
Netherlands	031	UK	Norway	047	**
United Kingdom	044	UK	Sweden	046	**
			Switzerland	041	**

** Keyboard program provided separately.

```
select 044 uk     select 49 gr /us
```

SET [*name* = [*string*]]

(Internal) Sets *name* equal to *string* in the environment for use in later programs and batch files. Displays the SET values if you omit all parameters. Including the *name* parameter without the *string* parameter removes the *name* from the environment. *name* and *string* cannot be numeric.

```
set drive=b:     set tty=vt52
```

SHARE [/F:*space*] [/L:*locks*]

(External) Installs file sharing and locking for active networking.

/F:*space* allocates file space (in bytes) to record file sharing information.

/L:*locks* specifies the number of locks allowed.

```
share /f:4096 /l:30
```

SHIFT

(Internal) Lets you use more than the usual ten replaceable parameters (%0-%9) in batch file processing. Each parameter definition shifts up one place.

```
shift
```

SHIPTRAK

(External) Parks the heads of a hard disk in preparation for moving the drive unit.

```
shiptrak
```

SORT [/R] [/+*n*] [<*input pathname*] [>*output pathname*]

(External) Reads input from the keyboard or the file specified by *input pathname*, sorts the data, and writes it to the screen or to the file specified by *output pathname*.

/R	reverses the sort (Z-A).
/+*n*	begins the sort at column *n*. The default is column 1.

```
sort /r <unsort.txt >sort.txt
```

SPOOLER [*printer*] [/P] [/S] [/C] [/G]

(External) Lets you send commands to and get the status of the print spooler loaded by CONFIG.SYS.

/P	pauses spooler printing. Issue the SPOOLER /P command again to resume printing. (The pause is automatically canceled and printing resumes when the printer buffer is full.)
/S	interrupts the spooler in order to print the current printer data.
/C	clears the spooler. Any data that is in the spooler when you clear it is not printed.
/G	returns the current spooler status, including whether the spooler is installed, if pause is on or off, if buffering is on or off, what the size of the buffer is, and what percentage of the buffer is full.

```
spooler /p
```

SUBST [*drive*:] [*pathname*] [/D]

3.2

(External) Substitutes a virtual drive name for a pathname.

drive:	is the virtual drive name.
pathname	is the pathname you want to replace.
/D	deletes an associated drive or pathname.

```
subst d: b:\sales\region1
```

SYS *drive*:

(External) Transfers the MS-DOS system files from the current disk to the disk in the specified *drive*.

```
sys b:
```

TIME [*time*]

(Internal) Displays or sets the time.

time	specifies the time in hours, minutes, seconds, and hundredths of a second (*hh:mm:ss.cc* format).

```
time 14:30    time 2:12:30    time
```

TREE [*drive*:] [/F]

(External) Displays all directories and subdirectories on the specified *drive*.

/F also displays the files on the specified *drive.*

```
tree b: /f
```

TYPE *pathname*

(Internal) Displays the contents of the file specified by *pathname*.

```
type b:carfile.txt
```

VER

(Internal) Displays the version number of your MS-DOS operating system.

```
ver
```

VERIFY [*switch*]

(Internal) Enables or disables disk write verify. VERIFY displays the current verify setting if you omit the switch parameter.

switch can be ON or OFF.

```
verify on
```

VOL [*drive*:]

(Internal) Displays the volume label of the disk in the current or specified *drive*.

```
vol b:          vol
```

XCOPY *source pathname* [*target pathname*] [/A] [/D:*date*] [/E] [/M] [/P] [/S] [/V] [/W]

(External) Copies files and directories, including subdirectories. You can use XCOPY to backup between two different disk drive types or media types.

source pathname	is the drive, directories, and/or file you want to copy.
target pathname	is the drive, directories, and/or file to copy to. If you omit this parameter, XCOPY copies to the current directory. The default filename is *.*.
/A	copies only files that have an archive bit set, without modifying the archive bit.
/D:*date*	copies only files modified on or after the specified date.
/E	copies any empty subdirectories. You can use this switch only with /S.
/M	copies only files that have an archive bit set, and modifies the source file by turning off the archive bit.
/P	displays a Y/N? prompt for each source file before copying it.
/S	copies directories and subdirectories, unless they are empty. If you omit /S, XCOPY works within a single directory.
/V	verifies each target file as it is written to be sure it is identical to the source file.
/W	waits to begin the XCOPY. At the message, press any key to continue, or press CTRL C to cancel XCOPY.

```
xcopy a: b: /s /e
```

CONTROL CHARACTER KEYS

Keys	Function
[BACKSPACE] or [CTRL] [H]	Backspace. Moves the cursor left one position and erases the character in that position.
[CTRL] [C]	Cancel. Stops execution of a command.
[CTRL] [N]	Echo. Toggles echoing of output to printer.
[ESC]	Escape. Voids the current line. \ appears on the screen.
[ENTER]	Execute command/carriage return. Processes the current command line and moves to the next line.
[CTRL] [J]	Line feed. Ends the current line and moves to the next line without processing the command line.
[PRINT] or [CTRL] [P]	Printer. Sends all output to the printer and to the screen. Press again to stop.
[SHIFT] [PRINT]	Print screen. Prints everything currently displayed on the screen.
[CTRL] [ALT] [DELETE]	Reset. Resets your computer.
[HOLD] or [CTRL] [S]	Stop scroll. Stops the screen from scrolling. Press [HOLD] again, or press [CTRL] [Q], to resume scrolling.

MS-DOS EDITING KEYS

Keys	Function
ENTER	Enter line. Makes the new line the new template and executes the command line.
INSERT	Insert character. Goes into the insert mode. (F3 ends the insert mode.)
DELETE	Delete character. Erases the next character from the template.
→ or F1	Copy character. Copies the next character from the template and displays it on the command line.
F2 *char*	Copy to *character*. Copies all characters up to the specified *character* and displays them on the command line.
F3	Template. Redisplays the entire template.
F4 *char*	Delete to *character*. Deletes all characters up to the specified *character* from the template.
F5	Replace template. Makes the line you type the new template, but does not execute the command line.
F6 or CTRL Z	End-of-file. Puts an end-of-file character in the template.

EDLIN COMMANDS

Append Lines

[*number*]A

Adds the specified *number* of lines from disk to memory. If you omit *number*, EDLIN appends lines until available memory is 75% full.

```
100a
```

Copy Lines

[*line1*][,*line2*],*line3*[,*count*]C

Copies all lines in the range *line1* to *line2* and places them immediately ahead of *line3* for the number of times specified by *count*.

```
3,9,12c        ,20,35c
```

Delete Lines

[*line1*][,*line2*]D

Deletes all lines in the range *line1* to *line2*. Deletes the current line if you omit *line1* and *line2*.

```
5,25d    4d      ,4d
```

Edit Line

[*line*]

Displays the specified *line* for editing.

```
4
```

End Edit

E

Ends the EDLIN program and saves the edited file.

```
e
```

Insert

[*line*]I

Inserts lines of text immediately before the specified *line*, or enters lines into a new file. Omit the *line* or include a period to use the current line. Include a number sign (#) to append the lines to the end of the file.

```
3i          .i          #i
```

List

[*line1*][,*line2*]L

Displays all lines in the range *line1* to *line2*.

```
2,5l        26l        ,10l
```

Move Lines

[*line1*][,*line2*],*line3*M

Moves all lines in the range *line1* to *line2* to the line immediately preceding *line3*.

```
23,30,100m
```

Page

[*line1*][,*line2*]P

Pages through a file 23 lines at a time or lists the specified block of lines.

```
10,15p        20p
```

Quit

Q

Quits the editing session without saving the file.

```
q
```

Replace String

[*line1*][,*line2*][?]R[*string1*] CTRL Z [*string2*]

Replaces all occurrences of *string1* with *string2* in the lines between *line1* and *line2*. ? prompts before each modification.

```
2,7?rand [CTRL] [Z] or
```

Search Text

[*line1*][,*line2*][?]S[*string*]

Searches all lines in the range *line1* to *line2* for each occurrence of the text *string*. ? prompts at each occurrence of *string*.

```
1,10sand
```

Transfer Lines

[*line*]T[*drive*:]*filename*

Inserts the contents of the file specified by *filename* immediately ahead of the specified *line* or current line of the file being edited.

```
10tb:myfile
```

Write Lines

[*number*]W

Writes a specified *number* of edited lines from memory to disk, beginning with Line 1. If you omit *number*, EDLIN writes until 25% of memory is freed.

```
100w
```

You can also use the special MS-DOS editing keys (ENTER, ESC, INSERT, DELETE, →, and F1-F5) to modify a EDLIN file. Refer to "MS-DOS Editing Keys."

DEBUG COMMAND PARAMETERS

Parameter	Description
address	An alphabetic segment register and offset. CS:0100 A segment address and offset. 04BA:0100 An offset only. (Default segment is CS for G, L, T, U, and W commands, DS for all other commands.)
byte	A one- or two-character hexadecimal value placed in or read from an address or register.
drive	A one digit value for the drive to be used for accessing or writing data, as follows: 0 = Drive A 2 = Drive C 1 = Drive B 3 = Drive D
filespec	A drive specification, filename, and extension. (The complete filespec is optional; however, you must specify at least the drive or filename.)
list	A series of strings or byte values.
portaddress	A hexadecimal value (4 characters maximum) that specifies a port number.
range	An area of memory specified by: *address1 address2* (*address2* must be an offset) *address* L *value* *value* = number of bytes to operate on (default=80) Do not use the *address* L *value* format if another hexadecimal value follows the *range* parameter.
registername	One of the following registers: AX CS BP DI BX DS IP SI CX ES SP PC DX SS F
sector	A hexadecimal value (1-3 characters) indicating the relative sector number on the disk.

sectorcount	A hexadecimal value (1-3 characters) indicating the number of disk sectors to write or load.
string	Any number of characters, enclosed by quotation marks (" or ').
value	A hexadecimal value (4 characters maximum).

DEBUG COMMANDS

Assemble

A[*address*]

Assembles statements directly into memory, starting at *address*.

```
a cs:0100
```

Compare

C *range address*

Compares the portion of memory specified by *range* to a portion of the same size beginning at the specified *address* and displays all differences.

```
c 100,1ff 300        c 100l100 300
```

Dump

D [*address*]
D [*range*]

Displays the contents of the specified memory *address* or *range*.

```
d cs:100 109
```

Enter

E *address* [*list*]

Enters byte values into memory at the specified *address*. Beginning at *address*, the Enter command replaces the contents of memory with the *list* of values.

```
e ds:100 45 a1 "abc" 0f      e cs:1004
```

Fill

F *range list*

Fills the memory locations in the specified *range* with the values in the *list*.

```
f 04ba:100 l 100 42 45 52 54 41
```

Go

G[=*address1*[*address2* . . .]]

Executes the program currently in memory. The Go command begins at *address1* and stops at each breakpoint (specified by *address2* . . .).

```
g cs:7550
```

Hex

H *value1 value2*

Displays the results of *value1* + *value2* and *value1* - *value2* (hexadecimal arithmetic).

```
h 19f 10a
```

Input

I *portaddress*

Inputs and displays one item from the specified port.

```
i 2f8
```

Load

L [*address*[*drive sector sectorcount*]]

Loads a file from the *drive* (0-3) into memory beginning at the specified *address*. Loads absolute sectors from the *drive* beginning at *sector* and continuing until the number of sectors specified by *sectorcount* are loaded.

```
l 04ba:100 2 0f 6d
```

Move

M *range address*

Moves the block of memory specified by *range* to the location beginning at *address*.

```
m cs:100 110 cs:500
```

Name

N *filespec1* [*filespec2* . . .]

Assigns program names for later LOAD or WRITE commands and assigns filespec parameters for the file being debugged.

```
n file1.exe
n file2.dat file3.dat
```

Output

O *portaddress byte*

Sends the *byte* to the specified *portaddress*.

```
o 2f8 4f
```

Proceed

P [=*address*] [*value*]

Beginning at *address*, executes the number of instructions specified by *value*. The Proceed command displays the register contents, flags, and the next instruction after it executes each instruction. If you omit all parameters, Proceed displays the registers and flags for the current instruction.

```
p               p=011a 10
```

Quit

Q

Quits the DEBUG program without saving the file.

```
q
```

Register

R [*registername*]

Displays the register or all flags specified by *registername* and lets you change the settings. The Register command displays the contents of all registers and flags if you omit the *registername* parameter.

```
r          rax          rf
```

Search

S *range list*

Searches the locations in the *range* for the *list* of bytes.

```
s cs:100 110 41
```

Trace

T[=*address*][*value*]

Executes one or more instructions (specified by *value*), beginning at *address*. The Trace command displays the register contents, flags, and the next instruction after each instruction executes.

```
t              t=011a 10
```

Unassemble

U [*address*]
U [*range*]

Disassembles instructions beginning at *address* or for the specfied *range*. The Unassemble command displays the addresses, the hexadecimal values, and the source statements that correspond to the instructions.

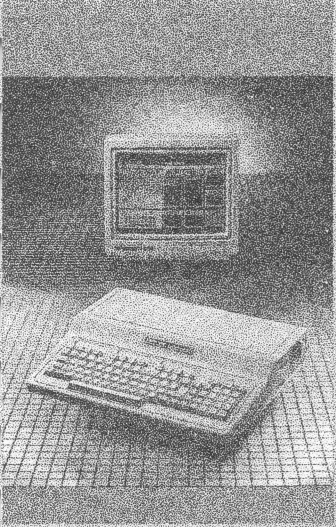
```
u 04Ba:100 l 10
```

Write

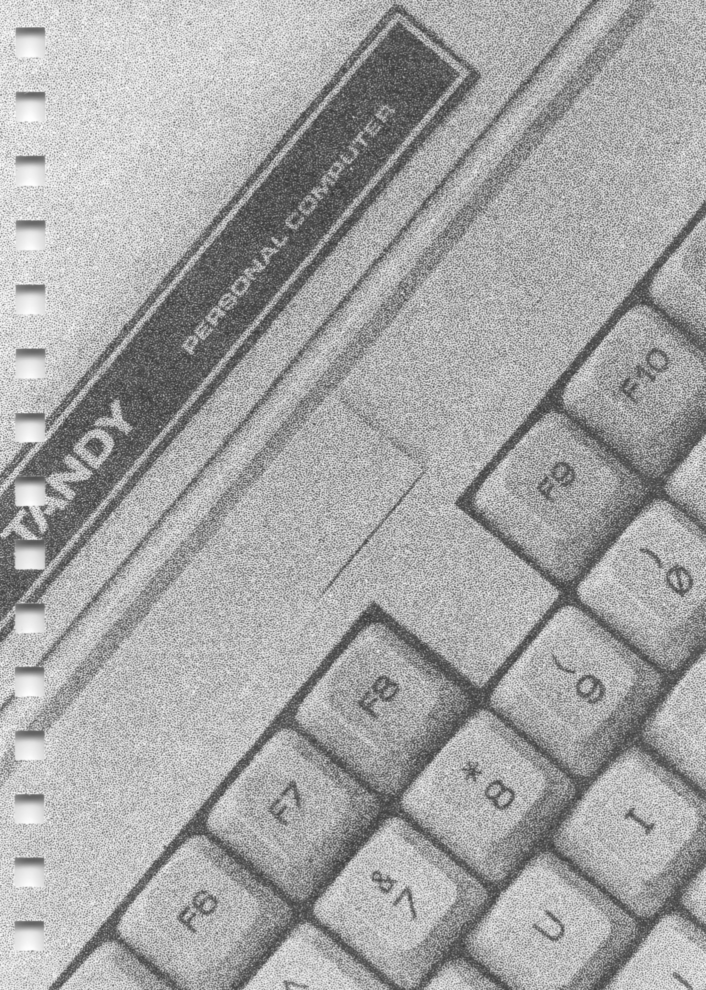
W [*address* [*drive sector sectorcount*]]

Writes the data being debugged to a disk file on the specified *drive* (0-3) beginning at the specfied *address*. Writes absolute sectors to the specified *drive* beginning at *sector* and continuing until the number of sectors specified by *sectorcount* are written.

```
w cs:100 1 37 2b
```



BASIC Quick Reference



CONTENTS

LOADING BASIC

Use the following syntax to load BASIC at the MS-DOS system prompt:

BASIC [*"pathname"*][<*input file*][>[>]*output file*]
[/F:*files*][/M:*memory location,block size*]
[/C:*buffer size*][/S:*record length*][/D][/I]

pathname loads and executes the specified BASIC program file.

<*input file* inputs data from the file specified instead of from the keyboard.

>*output file* outputs data to the file specified instead of to the video display. Use > to overwrite the existing *output file* or >> to append to it.

/F:*files* specifies the maximum number of data files that can be open at one time including those BASIC reserves for internal use. The maximum number of simultaneous opens is 15. The default is three files reserved for your use. Use a FILES command in your CONFIG.SYS file if you want a number other than the default. You **must** use the /I option along with the /F parameter.

/M:*memory location,block size* loads BASIC with the amount of reserved memory specified by *block size* (block size x 16). BASIC uses memory up to *memory location*, and memory above is reserved for machine language routines. The default is 64K bytes for BASIC. You must set the /M parameter if you plan to use the SHELL statement.

/C:*buffer size* sets the size of the RS232 receive buffer. The default is 256 bytes. (The RS232 transmit buffer is always set to 128 bytes.)

/S:*record length* sets the direct access record length. The default is 128 bytes. You **must** use the /I option along with the /S parameter.

/D loads BASIC with the double-precision transcendental math package.

/I tells BASIC not to dynamically allocate space during file operations.

BASIC COMMANDS AND STATEMENTS

Notation:

BOLD UPPERCASE indicates a command or statement. (Type commands and statements exactly as they appear.)

lowercase italics represent variable words, letters, characters, or values.

UPPERCASE indicates information you type exactly as it appears.

[ ] (square brackets) indicate optional parameters.

... (ellipsis) indicates that you can repeat a parameter.

Also, certain commands are valid only for a specific version of BASIC. These commands are indicated as follows:

2.02	BASIC V 2.02 command	3.20	BASIC V 3.20 command

ABS(*number*)

Computes the absolute value of *number*.

```
PRINT ABS(-44)    X = ABS(Y)
```

ASC(*string*)

Returns the ASCII code (a decimal number) for the first character of *string*.

```
PRINT ASC("A")    N=ASC(B$)
```

ATN(*number*)

Computes the arctangent of *number* in *radians*.

```
PRINT ATN(7)
X = ATN(Y/3) * 57.29578
```

AUTO [*line*][,*increment*]

Automatically generates a line number when you press ENTER. If *line* already exists in memory, BASIC displays an asterisk after the number. To turn off AUTO, press BREAK. *Line* is the starting line number. Default = Line 10. *Increment* is the increment to use when generating line numbers. Default = 10.

```
AUTO                        AUTO 100,50
```

BEEP [*switch*]

Produces a sound at 800 Hz for ¼ second.

Use BEEP with SOUND to direct sound to the computer's speaker or an external speaker (or both).

`BEEP ON: SOUND ON`
directs sound to both speakers

`BEEP OFF: SOUND OFF`
turns off sound to both speakers

`BEEP ON: SOUND OFF`
directs sound to the internal speaker only

`BEEP OFF: SOUND ON`
directs sound to the external speaker only

```
IF X > 20 THEN BEEP
```

BLOAD *"pathname"*[,*offset*]

Loads a memory image file into memory. *Offset* is the number of bytes into the current segment where BASIC loads the image. Must be in the range 0 to 65535. Default = value set by BSAVE.

```
BLOAD "PROG1.BAS"
BLOAD "PROG2.BAS",0
```

BSAVE *"pathname",offset,length*

Saves the contents of an area of memory as a disk file (memory image file). *Offset* is the number of bytes into the current segment where BASIC starts saving. Must be in the range 0 to 65535. *Length* is the length in bytes of the memory image file to be saved. Must be in the range 1 to 65535.

```
BSAVE "PROG1.BAS"
BSAVE "PROG2.BAS",50
```

CALL *variable* [(*parameter list*)]

Transfers program control to an assembly-language subroutine stored at *variable*. *Variable* contains the offset into the current segment where the subroutine starts in memory. The offset must be on a 16-byte boundary. *Parameter list* is the variables that are passed to the external subroutine.

```
CALL C                  CALL C (A$,Z,X)
```

CALLS *variable* [(*parameter list*)]

Transfers program control to an MS®-FORTRAN or MS-PASCAL routine stored at *variable. Variable* contains the offset into the current segment where the subroutine starts in memory. The offset must be on a 16-byte boundary. *Parameter list* is the variables that are passed to the external subroutine.

```
CALLS X                 CALLS X (S$)
```

CDBL(*number*)

Converts *number* to double-precision.

```
PRINT CDBL(465.342)     Z=CDBL(A)
```

CHAIN [MERGE] *"pathname"* [,[*line*]][,[ALL]] [,DELETE *line-line*]

Lets the current program load and execute the program specified by *pathname*, which must be saved in ASCII format. Commas in the syntax are significant and must be entered even if you omit the option. *Line* is the line number at which execution begins in the chained program. Default=first program line of the chained program.

ALL tells BASIC to pass every variable in the current program to the chained program. If you omit ALL, the current program must contain a COMMON statement to pass variables to the chained program. MERGE overlays the lines of the chained program with the current program. DELETE deletes lines in the overlay so that you can merge in a new overlay.

```
CHAIN "PROG2"
CHAIN "SUBPROG.BAS",,ALL
```

CHDIR *"pathname"*

Changes the current directory to the directory specified by *pathname*.

```
CHDIR "B:\ACCTS\RECVBLE"
CHDIR".."
```

CHR$(*code*)

Returns the character corresponding to an ASCII or control *code*.

```
PRINT CHR$(35)     C$=CHR$(32)
```

CINT(*number*)

Converts *number* to integer by rounding the fraction portion of *number. Number* must be in the range -32768 to 32767.

```
PRINT CINT(1.6)     Z=CINT(-1.67)
```

CIRCLE [STEP] (*x,y*),*radius* [,*color* [,*start,end*[,*aspect*]]]

Graphics. Draws an ellipse on the screen, the center of which is (*x,y*). STEP designates (*x,y*) as relative coordinates. *Radius* is the major axis of the ellipse. *Start,end* are the beginning and ending angles in radians. Must be in the range -6.283186 to 6.283186, or -2 * pi to 2 * pi. *Aspect* is the ratio of the x-radius to the y-radius in terms of coordinates. If *aspect* is less than 1, *radius* is the x-radius and is measured in points in the horizontal direction. If *aspect* is greater than 1, *radius* is the y-radius and is measured in points in the vertical direction.

```
CIRCLE (150,100),50
```

CLEAR [,*memory location*] [,*stack space*] [,*video memory*]

Frees memory for data without erasing the program currently in memory. CLEAR erases all arrays, sets numeric variables to zero and string variables to null, and erases any information set using a DEF statement, such as DEF SEG and DEF FN. CLEAR also turns off the SOUND, PEN, and STRIG functions and resets the music background.

Memory location specifies the highest memory location available for BASIC. *Stack space* specifies the amount of memory to set aside for temporarily storing internal data and addresses during subroutine calls and during FOR/NEXT loops. Default=768 bytes or one-eighth of the memory available, whichever is smaller. *Video memory* specifies the amount of memory to be set aside as video memory. Default=16K (16384).

```
CLEAR  CLEAR, 45000  CLEAR,,,32768
```

CLOSE [*buffer*[,...]]

Closes access to a disk file or communications channel. If you omit *buffer*, BASIC closes all open files.

```
CLOSE                 CLOSE 1, 2, 8
```

CLS

Clears the screen (or active viewport) and returns the cursor to the home position. Home is Row 0, Column 0, or in other words, the upper left corner of the screen.

```
CLS
```

COLOR [*background*] [,[*palette*]]
(Screen Mode 1)

COLOR [*foreground*][,[*background*]]
(Screen Modes 3-6)

Graphics. Selects the background color and either the palette or foreground colors, depending on the current screen mode. *Palette* specifies which palette to use in Screen Mode 1 and can be 0 or 1.

```
COLOR 9,0                  COLOR ,3
```

COLOR [*foreground*][,[*background*][,*border*]]

Text Mode Only. Selects the display colors for the foreground, background, and border for Screen Mode 0. COLOR can use any of the colors in the 16-Color Set as foreground and border. Specify *color*+16 as *foreground* to get a blinking foreground. *Background* can be Colors 0 to 15 of the 16-Color Set. If you select blinking foreground, *background* can be Colors 0 to 7.

```
COLOR 0,7                 COLOR 1,0
```

COM(*channel*) *action*

Turns on, turns off, or temporarily halts the trapping of activity on the communications channel. *Channel* specifies communications channel 1 or 2.

COM( ) ON	enables communications trapping.
COM( ) OFF	disables communications trapping.
COM( ) STOP	temporarily suspends communications trapping.

COMMON *variable*[,*variable,...*]

Passes *variables* to a chained program. Both programs in the chain should contain a COMMON statement.

```
COMMON A, B$, C, D(),G$()
```

CONT

Resumes program execution when stopped by the [BREAK] key or execution of a STOP or an END statement.

```
CONT
```

COS(*number*)

Computes the cosine of *number*.

```
PRINT COS(5.8)    Y = COS(X*.0174533)
```

CSNG(*number*)

Converts number to single-precision. BASIC rounds the number when converting it to single-precision.

```
PRINT CSNG(.1453885509)  Z=CSNG(a#)
```

CSRLIN

Returns the current row position of the cursor.

```
PRINT CSRLIN              A=CSRLIN
```

CVD(*8-byte string*)

Converts an 8-byte string to a double-precision number. Use to restore data to numeric form after it is read from the disk.

```
A# = CVD(GROSSPAY$)   D#=CVD(TOTAL$)
```

CVI(*2-byte string*)

Converts a 2-byte string to an integer. Use to restore data to numeric form after it is read from the disk.

```
A%=CVI(INVTRY$)    I=CVI(QTY$)
```

CVS(*4-byte string*)

Converts a 4-byte string to a single-precision number. Use to restore data to numeric form after it is read from the disk.

```
A!=CVS(TOTAL$)    S=CVS(DOLLR$)
```

DATA *constant* [,*constant*,...]

Stores numeric and string constants to be accessed by a READ statement. String constants containing delimiters, such as leading or trailing blanks, colons, or commas, must be enclosed in quotation marks when used in DATA statements.

```
DATA NEW YORK, CHICAGO, LOS ANGELES
```

DATE$[=*string*]

Sets the date or retrieves the current date. *String* is a literal, enclosed in quotation marks, that sets the date by assigning its value to DATE$. Month may be any number 01-12, day may be 01-31, and year may be 01-99 or 1980-2099. If you omit *string*, BASIC retrieves the current date.

```
DATE$="04/17/85"    TODAY$=DATE$
```

DEFDBL *letter* [,*letter*,...]

Defines any variables beginning with *letter(s)* as double-precision variables.

```
DEFDBL A       DEFDBL J-O
```

DEFINT *letter* [,*letter*,...]

Defines any variables beginning with *letter(s)* as integer variables.

```
DEFINT L        DEFINT A-G
```

DEFSNG *letter*[,*letter*,...]

Defines any variables beginning with *letter(s)* as single-precision variables.

```
DEFSNG T        DEFSNG Q-Z
```

DEFSTR *letter*[,*letter*,...]

Defines any variables beginning with *letter(s)* as string variables.

```
DEFSTR A        DEFSTR G-M
```

DEF FN*name* [(*argument list*)] = *expression*

Defines *name* as a function according to *expression. Name* is a valid variable name. *Argument list* is a list of dummy variables used in *expression*. They are replaced on a one-to-one basis with the variables or values given when the function is called. *Expression* defines the operation to be performed.

```
DEF FNR = RND(1)*69+10
DEF FNW# (A#,B#)=(A#-B#)^2
```

DEF SEG[= *address*]

Assigns the current segment address. The segment address is used by BLOAD, BSAVE, CALL, PEEK, POKE, and USR.

Address is an integer in the range 0 to 65535. *Address* must be on a 16-byte boundary. *Default* = BASIC's data segment (DS).

```
DEF SEG        DEF SEG=&HB800
```

DEF USR[*number*] = *offset*

Defines the user number and segment offset of a subroutine to be called by the USR function. *Number* may be an integer in the range 0 to 9.
Default = USR0. *Offset* is the number of bytes from the current segment address where the subroutine begins. Must be an integer in the range 0 to 65535.

```
DEFUSR = 0     DEF USR3 = &H0020
```

DELETE *line1-line2*

Deletes *line1* through *line2* of the program in memory. If you omit *line1*, BASIC deletes from the beginning of the program. If you omit *line2*, BASIC deletes to the end of the program. Use a period (.) to indicate the current line.

```
DELETE 70        DELETE .-110
```

DIM *array*(*dimension*)[,*array*(*dimension*),...]

Sets aside storage for *arrays* with the *dimensions* you specify. *Array* is the variable name of a string, integer, single-precision, or double-precision variable name. *Dimension* is one or more integer numbers separated by commas that define the dimensions of the array.

```
DIM AR(100)      DIM L1%(8,25)
```

DRAW *string*

Graphics. Draws an image on the screen. *String* specifies one or more of the movement commands listed below.

Movement Commands

Movement commands begin movement from the current graphics position, which is the coordinate of the last graphics point plotted with another graphics command. Current position defaults to the center of the screen if no previous graphics command has been executed.

U[*n*]	Moves up *n* points.
D[*n*]	Moves down *n* points.
L[*n*]	Moves left *n* points.
R[*n*]	Moves right *n* points.
E[*n*]	Moves diagonally up and right *n* points.
F[*n*]	Moves diagonally down and right *n* points.

G[*n*]	Moves diagonally down and left *n* points.
H[*n*]	Moves diagonally up and left *n* points.
M*x,y*	moves to point *x,y*. If you precede *x* with a plus (+) or minus (−) sign, DRAW assumes it is a relative position. Otherwise, it is an absolute position.

Prefix Commands

Prefix commands can precede the movement commands. They must be enclosed in quotation marks.

B	plots no points after move.
N	returns to original position when move is complete.
A*angle*	sets angle of move. *Angle* may be 0 to 3 (0 = 0 degrees, 1 = 90 degrees, 2 = 180 degrees, and 3 = 270 degrees.
Ccolor	sets color.
P*color,border*	paints using *color* and *border.*
Sfactor	sets scale factor. *Factor* is an integer in the range 1 to 255. The scale factor is *factor* divided by 4. Default=4 (scale of 1).
TA*angle*	moves at the specified angle. *Angle* is in the range -360 to +360. If *angle* is positive, movement is counterclockwise. If *angle* is negative, movement is clockwise.
X*variable*;	executes a substring. The X command lets you execute a second substring from the first string, much like the GOSUB statement. *Variable* is a string variable in your program that contains the substring you want to execute. The semicolon after *variable* is required.

```
DRAW "U30;"+"D30;"+"L40;"+"R40;"
```

EDIT *line*

Enters the Edit mode. BASIC displays *line* for editing. Use a period (.) to indicate the current line.

```
EDIT 100        EDIT.
```

END

Ends program execution and closes all files.

```
END
```

ENVIRON *"parameter id=text"* [;*"parameter id= text"*...]

Advanced Statement. Lets you modify BASIC's Environment String Table, such as to change the PATH parameter for a child process or to pass parameters to a child process. BASIC's Environment String Table is initially empty.

Parameter id is the name of the parameter. *Text* is the new parameter text. It must be separated from *parameter id* by an equal sign (=) or a space. If you omit *text*, or specify a null string or a semicolon (;), BASIC removes the parameter from the Environment String Table and compresses the table. *Parameter id=text* must be enclosed in quotation marks and be entered in uppercase characters.

```
ENVIRON "PATH=A:\"
ENVIRON "SALES=MYSALES"
```

ENVIRON$ [(*"parameter id"*)] [(*number*)]

Advanced Function. Returns the specified environment string from BASIC's Environment String Table.

Parameter id is the parameter for which to search and must be enclosed in quotation marks. *Number* specifies which parameter to return by its position within the table. *Number* and *parameter id* are mutually exclusive; only one may be specified on the command line.

```
PRINT ENVIRON$("PATH")
```

EOF(*buffer*)

Detects the end of a file. *Buffer* is the number assigned to the file when you opened it.

Sequential files: EOF returns 0 (false), when the end-of-file record has not been read yet, and -1 (true), when it has been read.

Direct access files: EOF returns -1 (true) if the last executed GET statement was unable to read an entire record because of an attempt to read beyond the physical end of the file.

```
IF EOF(1) THEN GOTO 1540
```

EOF(*buffer*)

Communications. Detects an empty input queue for communications files. *Buffer* is the number assigned to the file when you opened it.

ASCII mode: EOF returns -1 (true) if a CONTROL-Z is received. EOF remains true until the device is closed.

Binary mode: EOF returns -1 (true) when the input queue is empty. EOF becomes false when the input queue is not empty.

```
IF EOF(3) THEN RETURN
```

ERASE *array*[,*array*,...]

Erases one or more *arrays* from memory. Lets you either redimension arrays or use their previously allocated space in memory for other purposes.

```
ERASE C        ERASE G, H, I, Z$
```

ERDEV

Advanced Function. Returns the value of a device error within MS-DOS as set by the Interrupt 24 handler. The lower 8 bits of ERDEV contain the Interrupt 24 error code.

```
ERDEV
```

ERDEV$

Advanced Function. Returns the name of the device (as set by the Interrupt 24 handler) when a device error occurs. If the error occurred on a character device, ERDEV$ returns the 8-byte character device name. If the error does not occur on a character device, ERDEV$ returns the 2-character block device name.

```
ERDEV$
```

ERL

Returns the number of the line in which an error has occurred. If no error has occurred, ERL returns 0. If the error occurs while you are entering something at the prompt, ERL returns 65535 (the largest number that can be represented in 2 bytes).

```
PRINT ERL        E = ERL
```

ERR

Returns the error code if an error has occurred.

```
IF ERR = 7 THEN 1000 ELSE 2000
```

ERROR *code*

Simulates a specified error during program execution. *Code* is an integer expression in the range 0 to 255 specifying one of BASIC's error codes.

```
ERROR 1
```

EXP(*number*)

Computes the natural exponent of *number*, that is, *e* (base of natural logarithms) to the power of *number*. *Number* must be less than or equal to 88.02968.

```
PRINT EXP(-2)        A=EXP(-6)
```

FIELD *buffer, length* AS *variable* [,*length* AS *variable*...]

Divides a direct access buffer into fields so that you can send data from memory to disk and from disk to memory. Each field is identified by a string *variable* and is the *length* you specify. *Length* must be an integer in the range 1 to 255.

```
FIELD 3, 128 AS A$, 128 AS B$
```

FILES [*"pathname"*]

Displays the names of the files and directories on a disk.

If you specify *pathname*, BASIC lists all files that match that pathname. If you omit the filename when specifying *pathname*, BASIC lists all files and directories in the specified directory. Default = all files and directories in the current directory on the current drive.

```
FILES                    FILES "\BOOKS\"
```

FIX(*number*)

Returns the truncated integer of *number*.

```
PRINT FIX(2.6)        Z=FIX(B)
```

FOR *variable* = *initial value* TO *final value* [STEP *increment*]
NEXT [*variable*]

Establishes a program loop that allows a series of program statements to be executed a specified number of times. *Variable* must be either integer or single precision. *Increment* is the number BASIC adds to *initial value* each time the loop is executed. Default = 1.

```
FOR I=1 TO I + 5:PRINT I:NEXT
```

FRE(*dummy argument*)

Returns the number of bytes in memory not being used by BASIC. If you specify a numeric argument, BASIC returns the amount of memory available. If you specify a string argument, BASIC compresses the data before returning the amount of memory available. BASIC automatically compresses data if it runs out of workspace.

```
PRINT FRE("44")    PRINT FRE(44)
```

GET [#]*buffer*[,*record*]

Reads a record from a direct access disk file and places it in the specified *buffer*. The number sign (#) is not required. *Record* is an integer in the range 0 to 16,777,215. Default = the next sequential record (after the last GET).

```
GET 1          GET 1,25
```

GET *[#]buffer,number*

Communications. Transfers data from the communications line to the communications buffer. The number sign (#) is not required. *Number* is the number of bytes to transfer.

```
GET 1,8
```

GET *(x1,y1)-(x2,y2),array*

Graphics. Transfers points from an area on the display to an array.

(x1,y1) are the coordinates at which the image begins. *(x2,y2)* are the coordinates at which the image ends. *Array* is a numeric array to hold the image.

```
GET (0,0)- (100,100),Z
```

GOSUB *line*

Branches to the subroutine, beginning at *line*. Every subroutine must end with a RETURN statement.

```
GOSUB 1000
```

GOTO *line*

Branches to the specified *line*.

```
GOTO 100     IF R=13 THEN GOTO 80
```

HEX$(*number*)

Computes the hexadecimal value of *number*.

```
PRINT HEX$(30)     Y$=HEX$(X/16)
```

IF *expression* **THEN** *statement(s)*[**ELSE** *statement(s)*]

Tests a conditional expression and makes a decision regarding program flow. If *expression* is true, BASIC executes the THEN *statement*. If *expression* is false, BASIC executes the matching ELSE *statement* or the next program line.

```
IF A = B THEN PRINT "A = B"
   ELSE PRINT "A <> B"
```

INKEY$

Returns a one-character string from the keyboard without pressing ENTER. If no key is pressed, BASIC returns a null string (length zero). INKEY$ does not echo the character to the display.

```
A$ = INKEY$:IF A$ =""THEN 10
```

INP(*port*)

Returns the byte read from *port*. *Port* may be any integer from 0 to 65535.

```
PRINT INP(255)     A=INP(255)
```

INPUT[;] [*"prompt";*]*variable*[,*variable*,...]

Accepts data from the keyboard and stores it in one or more variables. BASIC stops execution and displays *prompt* followed by a question mark to indicate that the program is waiting for input. If you do not want BASIC to display the question mark, type a comma instead of a semicolon after *prompt.*

If INPUT is immediately followed by a semicolon (;), BASIC does not echo the ENTER key when you press it as part of a response.

```
INPUT Y%
INPUT "ENTER YOUR NAME AND
  AGE"; N$,A
```

INPUT# *buffer, variable*[,*variable*...]

Accepts data from a sequential device or file and stores it in a program *variable. Buffer* is the number assigned to the file when you opened it.

```
INPUT#1, A,B    INPUT#4, A$, B$, C$
```

INPUT$(*number* [,[#]*buffer*])

Inputs a string of characters from either the keyboard or a sequential access file. *Number* specifies the number of characters to be input and may be in the range 1 to 255.

If you include *buffer*, BASIC inputs the string from a sequential access file. If you omit *buffer,* BASIC inputs the string from the keyboard. The number sign (#) is not required.

```
A$ = INPUT$(5)     A$ = INPUT$(11,3)
```

INSTR([*number*,]*string1*,*string2*)

Searches for the first occurrence of *string2* in *string1* and returns the position at which the match is found. *Number* specifies the position in *string1* to begin searching for *string2* and must be an integer in the range 1 to 255. Default = first character in *string1*.

```
PRINT INSTR (3, "1232123", "12")
A$ = "LINCOLN":P=INSTR(A$,"INC")
```

INT(*number*)

Converts *number* to the largest integer that is less than or equal to *number*. *Number* is not limited to the integer range.

```
PRINT INT(79.89)  PRINT INT (-12.11)
```

IOCTL [#]*buffer*,*string*

Advanced Statement. Sends a control data string to a device driver. *Buffer* is the number assigned to the driver when you opened it. The number sign (#) is not required.

String is a string expression containing a series of commands called "control data." The commands are generally 2 to 3 characters long and may be followed by an alphanumeric argument. The commands are separated by semicolons (;). *String* can be a maximum of 255 bytes.

```
IOCTL 1,"PL56"
```

IOCTL$([#]*buffer*)

Advanced Function. Returns the control data string from a device driver that you have opened previously. *Buffer* is the number assigned to the driver when you opened it. The number sign (#) is not required.

```
IF IOCTL$(1) ="NR" THEN PRINT
  "PRINTER NOT READY"
```

KEY *number*,*string*

Assigns or displays function key values. *Number* indicates the function key (1-12) or the user key (17-20) being defined. See KEY (*number*) *action*. *String* is the string expression assigned to the key and may contain a maximum of 15 characters.

KEY ON

Displays the function key assignment values on Line 25 of the screen. BASIC shows only the first 5 characters of the string. CTRL T is the same as KEY ON.

KEY OFF

KEY OFF erases the soft key assignments from Line 25. The assignments are still active, but the screen does not display them.

KEY LIST

KEY LIST displays all 15 characters of all 12 soft key assignments on the screen.

KEY(*number*) *action*

Turns on, turns off, or temporarily halts key trapping for a specified key.

KEY() ON	enables key trapping
KEY() OFF	disables key trapping
KEY() STOP	temporarily suspends key trapping

Number may be a number in the range 1 to 20, indicating the number of the key to trap. Function keys use their corresponding function key number (1-12). The cursor directions keys are:

↑	13
←	14
→	15
↓	16

User-defined keys are 17-20. Use the following syntax to define your own user keys:

KEY *number,* CHR$(*key*) + CHR$(*scan*)

Key is one of the following:

&H40	CAPS lock key
&H20	NUM LOCK key
&H08	ALT key
&H04	CTRL key
&H02	Left SHIFT key
&H01	Right SHIFT key

Scan is the scan code for a physical key on the keyboard.

KILL *"pathname"*

Kills (deletes) *pathname* from disk.

```
KILL "FILE.BAS"
KILL "A:\REPORT\DATA"
```

LCOPY

Copies all text data on the screen to the printer.

```
LCOPY
```

LEFT$(*string,number*)

Returns the specified number of characters from the left portion of *string*. *Number* must be in the range 1 to 255.

```
PRINT LEFT$("BATTLESHIPS",6)
```

LEN(*string*)

Returns the number of characters in *string*. Blanks are counted.

```
PRINT LEN("DOG") + LEN("TERRIER")
X = LEN(SENTENCE$)
```

LET *variable = expression*

Assigns the value of *expression* to *variable*. BASIC does not require assignment statements to begin with LET.

```
LET A$ = "A ROSE IS A ROSE"
B1 = 1.23
```

LINE [[STEP](*x1,y1*)]-[STEP](*x2,y2*),[*color*][,B[F]] [,*style*]

Graphics. Draws a line or a box on the video display.

STEP designates (*x,y*) as relative coordinates.
(*x1,y1*) are the coordinates at which the line begins.
Default = last point referenced on the screen.
(*x2,y2*) are the coordinates at which the line ends.

With the B option, BASIC draws a box. The points that you specify are opposite corners. If you specify both the B and F options, BASIC draws a box and fills the box in with *color*. *Style* is a 16-bit integer that lets you select the line-style used when drawing normal lines and unfilled boxes. Each bit represents a point in the line. If the bit equals 1, then the point is drawn. If the bit equals zero, then the point is not drawn.

```
LINE (0,0)-(319,199)
LINE -(319,199),BF
```

LINE INPUT[;][*"prompt";*] *string variable*

Accepts an entire line (a maximum of 254 characters) from the keyboard, including delimiters (commas, quotation marks, etc.). BASIC stops execution and displays *prompt* to indicate that the program is waiting for input.

The only way to terminate the string input is to press [ENTER]. However, if LINE INPUT is immediately followed by a semicolon, pressing [ENTER] does not echo a carriage return to the display.

```
LINE INPUT A$
LINE INPUT "LAST, FIRST NAME?"; N$
```

LINE INPUT#*buffer, variable*

Accepts an entire line of data from a sequential access file, including delimiters (commas, quotation marks, etc.). *Buffer* is the number assigned to the file when you opened it.

```
LINE INPUT#1, A$
```

LIST *startline-endline* [,*"device:"*]

Lists a program in memory to the display. *Startline* specifies the first line to be listed. Default = first line in the program. *Endline* specifies the last line to be listed. Default = last line in the program. *Device*: can be either SCRN: (screen) or LPT1: (printer). Default = screen (SCRN:).

```
LIST     LIST 50-100, "LPT1:"
```

LLIST *startline-endline*

Lists program lines in memory to the printer. LLIST assumes a 132-character-wide printer. You may change this by using the WIDTH statement. *Startline* and *endline* are described in LIST.

```
LLIST       LLIST 68-90
```

LOAD *"pathname"* [,R]

Loads a BASIC program from disk into memory. The R option tells BASIC to run the program.

```
LOAD "A:PROG1.BAS"
LOAD "PROG1.BAS",R
```

LOC(*buffer*)

Returns the current record position within a file. *Buffer* is the number assigned to the file when you opened it.

Direct access files: LOC returns the record number accessed by the last GET or PUT statement.

Sequential access files: LOC returns the number of 128-byte records that have been read or written.

```
A=LOC(2)     IF LOC(1)>55 THEN END
```

LOC(*buffer*)

Communications. Returns the number of characters in the input queue. *Buffer* is the number assigned to the file when you opened it

If more than 255 characters are in the input queue, LOC always returns 255. If fewer are there, LOC returns the actual number of characters waiting to be read.

```
IF LOC(X)>0 THEN 1000
```

LOCATE *[row][,[column][,[cursor][,[start][,stop]]]]*

Positions the cursor on the screen at the position indicated by *row* and *column. Cursor* indicates whether the cursor is visible or invisible. 1 = visible and 0 = invisible. *Start* is the first scan line of the cursor. *Stop* is the last scan line of the cursor. *Start* and *stop* can be in the range 0 to 7.

```
LOCATE 10,20,1,4    LOCATE 24,1,1,3
```

LOCK *[#]buffer[,record]*
UNLOCK *[#]buffer [,record]*

Controls access by other processes to all or part of an opened file, specified by *buffer*. LOCK and UNLOCK are used only by the compiler. *Record* is the record or the range of records to lock or unlock.

```
LOCK 1, 1 TO 4    UNLOCK 1, 1 TO 4
```

LOF*(buffer)*

Returns the length of the file in bytes. *Buffer* is the number assigned to the file when you opened it.

```
Y = LOF(5)
```

LOF*(buffer)*

Communications. Returns the amount of free space in the input queue. You can use LOF to determine when an input queue is getting full so that transmission is stopped.

```
IF LOF(X) <20 GOTO 1000
```

LOG*(number)*

Computes the natural logarithm of *number. Number* must be greater than zero.

```
PRINT LOG(3.14159)
Z = 10 * LOG(P5/P1)
```

LPOS*(number)*

Returns the logical position of the print head within the printer's buffer. *Number* can be 0 or 1 to indicate LPT1:.

```
IF LPOS (X)>60 THEN LPRINT
```

LPRINT [USING *format;*] *data*[,*data*,...]

Prints *data* on the printer. LPRINT and LPRINT USING assume a print width of 132 characters. You may change the width with the WIDTH statement.

See PRINT and PRINT USING for more information on formatting the output.

```
LPRINT (A * 2)/3
LPRINT USING "#####.#"; 2.17
```

LSET *field name* = *data*

Moves *data* to the direct access buffer and places it in *field name*, in preparation for a PUT statement. *Field name* is a string variable defined in a FIELD statement. You must have used FIELD to set up buffer fields before using LSET.

Any numeric value that is placed in a direct access file buffer with an LSET statement must be converted to a string. See MKS$, MKD$, and MKI$.

```
LSET AD$ = "2000 EAST PECAN ST."
LSET TD$=D$
```

MERGE *"pathname"*

Loads a BASIC program and merges it with the program currently in memory. Program lines in *pathname* are inserted into the resident program in sequential order. The file must be in ASCII format; that is, it must have been saved with the A option.

If line numbers in *pathname* coincide with line numbers in the resident program, *pathname*'s program lines replace the resident program's lines.

```
MERGE "PROG2.TXT"
```

MID$(*oldstring,start*[,*length*]) = *newstring*

Replaces a portion of *oldstring* with *newstring. Start* specifies the position of the first character you want to change. *Length* is the number of characters you want to replace.

```
A$=MID$ ("ABCDEFGHIJ",3,4)
A$=MID$(Z$,4,5)
```

MID$(*string, start* [,*length*])

Returns a substring of *string. Length* is the number of characters in the substring. It must be in the range 1 to 255. *Start* specifies the position in the string from which to get the substring.

```
PRINT MID$("WEATHERFORD",3,2)
A$=MID$(T$,4,5)
```

MKDIR *"pathname"*

Creates the directory specified by *pathname*.

```
MKDIR "A:\ACCTS\PAYABLE"
MKDIR "\ADDRESS"
```

MKD$(*double-precision expression*)

Converts a numeric value to an 8-byte string value. This is the inverse function of CVD. Any numeric value that is placed in a direct access file buffer by an LSET or RSET statement must be converted to a string.

```
LSET YTD$=MKD$(564.33)
RSET DAY$=MKS$(DAY)
```

MKI$(*integer expression*)

Converts a numeric value to a 2-byte string value. This is the inverse function of CVI. Any numeric value that is placed in a direct access file buffer by an LSET or RSET statement must be converted to a string.

```
LSET TOT$=MKI$(TOT)
RSET QTY$=MKI$(NUM)
```

MKS$(*single-precision expression*)

Converts a numeric value to an 4-byte string value. This is the inverse function of CVS. Any numeric value that is placed in a direct access file buffer by an LSET or RSET statement must be converted to a string.

```
LSET AVG$=MKS$(0.123)
RSET MIX$=MKS$(A)
```

NAME *"old filename"* AS *"new filename"*

Renames *old filename* as *new filename*. You cannot change directory names.

```
NAME "OLDFILE.BAS" AS
  "NEWFILE.BAS"
```

NEW

Deletes the program currently in memory and clears all variables.

```
NEW
```

NOISE *source,volume,duration*

Generates noise through a TV monitor's speaker (external speaker). You must execute a SOUND ON statement before using NOISE. *Source* selects the type of noise and may be an integer in the range 0 to 7. 0-3 selects periodic noise and 4-7 selects white noise. *Volume* is an integer in the range 0 to 15 where 0 is the quietest and 15 is the loudest. Default = 8. *Duration* may be in the range 0 to 65536. A *duration* of 18.2 equals 1 second.

```
NOISE 0,15,20
```

OCT$(*number*)

Returns a string that represents the octal value of a decimal *number*.

```
PRINT OCT$(30)        S$=OCT$(90)
```

ON COM(*channel*) **GOSUB** *line*

Transfers program control to a subroutine beginning at *line* when activity occurs on the specified communications channel. *Channel* specifies communications channel 1 or 2. *Line* is the subroutine line at which execution begins when activity occurs on the communications channel. Specifying Line 0 turns off communications trapping.

```
ON COM(1) GOSUB 1000
```

ON ERROR GOTO *line*

Transfers control to *line* if an error occurs. You must execute an ON ERROR GOTO before the error occurs. Specifying Line 0 turns off error trapping.

```
ON ERROR GOTO 1500
```

ON *n* **GOSUB** *line*[,*line*,...]

Looks at *n* and transfers program control to the subroutine indicated by the *n*th line listed. If *n* equals 1, BASIC branches to the first line listed. If *n* equals 2, BASIC branches to the second line listed, and so on. *N* must be in the range 0 to 255.

```
ON Y GOSUB 1000, 2000, 3000
```

ON *n* **GOTO** *line*[,*line*...]

Looks at *n* and transfers program control to the *n*th line listed. If *n* equals 1, BASIC branches to the first line listed. If *n* equals 2, BASIC branches to the second line listed, and so on. *N* must be in the range 0 to 255.

```
ON MI GOTO 150, 160, 170, 150, 180
```

ON KEY(*number*) **GOSUB** *line*

Transfers program control to a subroutine, beginning at *line* when you press the specified key. *Number* indicates the number of the key to trap. Function keys are 1 to 12. The cursor direction keys are numbered:

$\boxed{\uparrow}$ 13
$\boxed{\leftarrow}$ 14
$\boxed{\rightarrow}$ 15
$\boxed{\downarrow}$ 16

User keys are numbered 17 through 20. User keys are defined with the KEY statement. Specifying Line 0 turns off key trapping for the specified key.

```
ON KEY(13) GOSUB 500
```

ON PEN GOSUB *line*

Transfers program control to the subroutine at *line* when you activate the light pen. Specifying Line 0 turns off pen trapping.

```
ON PEN GOSUB 1000
```

ON PLAY(*number*) **GOSUB** *line*

Transfers program control to the subroutine, beginning at *line* when the number of notes in the background music buffer is less than *number*. *Number* indicates that control should transfer to *line* when the number of notes left in the music buffer is less than *number*. *Number* must be in the range 1 to 32. Specifying Line 0 turns off play trapping.

```
ON PLAY(30) GOSUB 200
```

ON STRIG(*number*) **GOSUB** *line*

Transfers program control to the subroutine at *line* when you press one of the joystick's buttons. *Number* specifies the button pressed and is one of the following:

0	left joystick, button 1
2	right joystick, button 1
4	left joystick, button 2
6	right joystick, button 2

Specifying Line 0 turns off joystick trapping.

```
ON STRIG(0) GOSUB 1000
```

ON TIMER(*number*) **GOSUB** *line*

Transfers program control to the subroutine, beginning at *line* when the specified time has elapsed. *Number* indicates the number of seconds. It may be a value in the range 1 to 86400 (86400 seconds = 24 hours).

```
ON TIMER(3600) GOSUB 500
```

OPEN *"mode",[#]buffer,["pathname"]["device:"]* *[,record length]*
OPEN *["pathname"]["device:"]* [FOR *mode*] AS [#]*buffer* [LEN=*record length*]

Establishes an input/output path for a file or device. *Buffer* specifies the I/O buffer in memory to use when accessing the file and may be in the range 1 to 255. The number sign (#) is not required. If you do not specify *pathname*, you must specify *device:*. *Record length* sets the record length for direct access files and may be in the range 1 to 32768. Default = 128 bytes. *Mode* specifies any of the following:

O or OUTPUT	sequential output mode
I or INPUT	sequential input mode
A or APPEND	sequential extension of an existing file
R or RANDOM	direct input/output mode

In the first form of the syntax, you must use the abbreviated form of *mode* and enclose it in quotation marks.

In the second form of the syntax, you must specify the complete word for *mode*. You cannot specify RANDOM. If you want to use direct access in the second form of the syntax, omit *mode*.

```
OPEN "R",2,"TEST.DAT"
OPEN "LPT1:"FOR OUTPUT AS #2
```

OPEN *"mode", buffer,["pathname"]["device:"]* *[,record length]*
OPEN *["pathname"]["device:"]* [FOR mode] [*access*] AS *buffer* [LEN=*record length*]

Establishes an input/output path for a file or device. *Buffer* specifies the I/O buffer in memory to use when accessing the file. It can be in the range 1 to 16. If you do not specify *pathname*, you must specify *device:*. *Record length* sets the record length for direct access files. It can be in the range 1 to 32768. Default = 128 bytes. *Mode* specifies any of the following:

O or OPEN	sequential output mode

I or INPUT	sequential input mode
A or APPEND	sequential extension of an existing file
R or RANDOM	direct intput/output mode

In the first form of the syntax, you must use the abbreviated form of *mode*, and must enclose it in quotation marks.

In the second form of the syntax, you must specify the complete word for *mode*. You cannot specify RANDOM. If you want to use direct access in the second form of the syntax, omit *mode*.

Access controls the processes that can access the file and the degree to which they do so. *Access* can be SHARED, LOCK READ, LOCK WRITE, or LOCK READ WRITE.

```
OPEN "R",2,"TEST.DAT"
OPEN "LPT1:" FOR OUTPUT AS 2
```

OPEN "COM *channel:* [*speed*] [,*parity*] [,*data*][,*stop*][,RS] [,CS[*seconds*]][,DS[*seconds*]] [,CD[*seconds*]][,*mode*][,PE] [,LF]" [FOR *mode*] AS [#][*buffer*][LEN =*number*]

2.02

Communications. Opens a file and allocates a buffer for RS-232C (Asynchronous Communications Adapter) communication. *Channel* can be 1 or 2 to select the communications channel to be opened. *Speed* specifies the baud rate. It can be 75, 110, 150, 300, 600, 1200, 2400, 4800, or 9600. Default = 300. *Parity* can be E for EVEN, O for ODD, M for MARK, S for SPACE or N for NO. Default = E. *Data* specifies the number of bits. It can be 5, 6, 7 or 8. Default = 7. *Stop* can be either 1 or 2 to indicate the number of stop bits. Default = 2 for baud rates of 75 and 100, and 1 for all other baud rates. *Mode* is either OUTPUT or INPUT for sequential access. Default = random input/output. *Buffer* indicates the buffer that accesses the file. It can be in the range 1 to 255. *Number* specifies the maximum number of bytes that can be accessed in the communications buffer by GET and PUT statements. Default = 128 bytes.

```
OPEN "COM1:" AS 1
OPEN "COM1:9600,N,8,1,BIN" AS 2
```

OPEN "COM *channel:* [*speed*] [,*parity*] [,*data*][,*stop*][,RS] [,CS[*seconds*]][,DS[*seconds*]] [,CD[*seconds*]][,*mode*][,PE] [,LF]" [FOR *mode*] AS [#][*buffer*][LEN =*number*]

Communications. Opens a file and allocates a buffer for RS-232C (Asynchronous Communications Adapter) communication. *Channel* can be 1 or 2 to select the communications channel to be opened. *Speed* specifies the baud rate. It can be 75, 110, 150, 300, 600, 1200, 2400, 4800, or 9600. Default = 300. *Parity* can be E for EVEN, O for ODD, M for MARK, S for SPACE or N for NO. Default = E. *Data* specifies the number of bits. It can be 5, 6, 7 or 8. Default = 7. *Stop* can be either 1 or 2 to indicate the number of stop bits. Default = 2 for baud rates of 75 and 100, and 1 for all other baud rates. *Mode* is either OUTPUT or INPUT for sequential access. Default = random input/output. *Buffer* indicates the buffer that accesses the file. It can be in the range 1 to 16. *Number* specifies the maximum number of bytes that can be accessed in the communications buffer by GET and PUT statements. Default = 128 bytes.

```
OPEN "COM1:" AS 1
OPEN "COM1:9600,N,8,1,BIN" as 2
```

OPTION BASE *value*

Sets *value* as the minimum value for an array subscript. This statement must precede the DIM statement. *Value* may be 1 or 0. Default = 0.

```
OPTION BASE 1
```

OUT *port, data byte*

Sends a *data byte* to a machine output *port.* A port is an input/output location in memory. *Port* is an integer in the range 0 to 65535 and *data byte* is an integer in the range 0 to 255.

```
OUT 32,100
```

PAINT *(x,y)* *[color[,border][,background]]*

Graphics. Fills in an area on the display with a selected color or pattern. *(x,y)* are the coordinates at which painting begins. *Color* can be either a number or a string expression. If *color* is a number it specifies a color number available in the current screen mode. If *color* is a string expression, it specifies the mask to be used for tiling in the form:

```
CHR$(&Hnn)+CHR$(&Hnn)+CHR$(&Hnn)...
```

Border is the color at which to stop painting. *Background* is the color to skip when checking for borders while paint tiling.

PALETTE *[color,display color]*

Graphics. Changes the color associated with a particular color number in the current palette. *Color* specifies the color in the current palette you want to change. *Display color* specifies the new color you want BASIC to display when *color* is specified.

```
PALETTE 3,7
```

PALETTE USING *array(subscript)*

Graphics. Changes the colors associated with more than 1 of the color numbers in the current palette. *Array* is the name of an integer array in which you can define the order of colors to be put in the current palette. *Subscript* is the position in the array that contains the value of the first position for the palette.

```
PALETTE USING A(0)
PALETTE USING A(2)
```

PCOPY *source page, destination page*

Copies the *source* video page to the *destination* video page.

```
PCOPY 3,5       PCOPY 6,4
```

PEEK(*memory location*)

Returns a byte from *memory location. Memory location* must be in the range -32768 to 65535. The value returned is an integer in the range 0 to 255.

```
A = PEEK (&H5A00)
```

PEN(*number*)

Returns the light pen's coordinates.

Number is a number in the range 1 to 9 that tells BASIC what to return.

0 Returns a -1 if pen button has been pressed since last poll. Returns a 0 if not.

1 Returns the x-coordinate (horizontal) where the pen was last activated.

2 Returns the y-coordinate (vertical) where the pen was last activated.

3 Returns a -1 if the pen button is pressed. Returns a 0 if it is not.

4 Returns the last known valid x-coordinate (horizontal).

5 Returns the last known valid y-coordinate (vertical).

6 Returns the character row position where the pen was last activated.

7 Returns the character column position where the pen was last activated.

8 Returns the last known character row position.

9 Returns the last known character column position.

```
A = PEN(1)
```

PEN *action*

Turns on, turns off, or temporarily halts light pen event trapping.

PEN ON	enables event trapping.
PEN OFF	disables event trapping.
PEN STOP	temporarily suspends event trapping.

PLAY *string*

Plays the musical notes specified by *string. String* is a string expression consisting of 1 or more single-character music commands.

Single character music commands:

A - G	plays notes A through G of 1 musical scale. You may include an optional number sign (#) or plus sign (+) to indicate a sharp note or a minus sign (−) to indicate a flat note.
L*n*	sets the duration of the notes that follow. *n* may be a value in the range 1 to 64 where: 1 indicates a whole note. 2 indicates a half note. 4 indicates a quarter note. 8 indicates an eighth note. 16 indicates a sixteenth note.
O*n*	sets the current octave. There are 7 octaves, 0 through 6. Octave 3 starts with middle C. Default = Octave 4.
N*n*	plays a note. *N* may be in the range 0 to 84.
P*n*	rests. *N* may be in the range 1 to 64.
T*n*	sets the number of quarter notes in 1 minute. *N* may be in the range of 32 to 255. Default = 120 quarter notes in 1 minute.
•	plays as a dotted note. BASIC plays the note one-half its length longer.
MF	plays the music in the foreground. Default = MB.
MB	plays the music in the background. A maximum of 32 notes and/or rests can play in background at a time. Default = MB.
MN	sets "music normal"; each note plays 7/8 of the duration as set by the L option. Default = MN.
ML	sets "music legato"; each note plays the full duration as set by the L option. Default = MN.
MS	sets "music staccato"; each note plays 3/4 of the duration as set by the L option. Default = MN.

X *variable;*	executes a substring. You can have 1 string execute another, which executes a third, and so on.
V*n*	sets the volume. *N* must be in the range 0 to 15. You must execute a SOUND ON statement to use this option. Default = 8.

```
PLAY "C4F.C8F8.C16F8.G16A2F2"
```

PLAY(*number*)

Returns the number of notes currently in the background music queue. *Number* is a dummy argument when SOUND is OFF. If you execute a SOUND ON, then *number* may be one of the following (Default = 0):

0 returns the number of notes left to play on voice channel 0.

1 returns the number of notes left to play on voice channel 1.

2 returns the number of notes left to play on voice channel 2.

```
X=PLAY(0)        X=PLAY(2)
```

PLAY *action*

Turns on, turns off, or temporarily halts background music event trapping.

PLAY ON	enables play event trapping.
PLAY OFF	disables play event trapping.
PLAY STOP	temporarily suspends play event trapping.

PMAP(*coordinate,action*)

Returns the physical or world coordinate for the specified coordinate. *Coordinate* is any x- or y-coordinate. *Action* is one of the following:

0 returns the physical x-coordinate for the specified world coordinate.

1 returns the physical y-coordinate for the specified world coordinate.

2 returns the world x-coordinate for the specified physical coordinate.

3 returns the world y-coordinate for the specified physical coordinate.

```
X=PMAP(200,3)          Z=PMAP(50,0)
```

POINT *(x,y)*
POINT *(action)*

Graphics. Returns the color number of a point on the screen or returns the current physical or world coordinates. *(x,y)* are the coordinates of the point. *Action* is one of the following:

0 returns the current physical x-coordinate (horizontal).

1 returns the current physical y-coordinate (vertical).

2 returns the world x-coordinate if WINDOW is active. Otherwise, returns the physical x-coordinate.

3 returns the world y-coordinate if WINDOW is active. Otherwise, returns the physical y-coordinate.

```
IF POINT(1,1) <>0 THEN PRESET (1,1)
  ELSE PSET (1,1) X=POINT(1)
```

POKE *memory location, data byte*

Writes *data byte* into *memory location.* Both *memory location* and *data byte* must be integers. *Memory location* must be in the range -32768 to 65535.

```
POKE &H5A00,&HFF
```

POS(*number*)

Returns the current column position of the cursor. *Number* is a dummy argument.

```
IF POS(X) >70 THEN IF A$ = CHR$(32)
  THEN A$ = CHR$(13)
```

PRINT *data*[,*data*...]

Prints numeric or string *data* on the display. You can substitute a question mark (?) in place of the word PRINT. If you use commas, the cursor automatically advances to the next tab position before printing the next item. If you use semicolons or spaces to separate the data items, PRINT prints the items without any spaces between them.

```
PRINT "DO"; "NOT"; "LEAVE";
   "SPACES"
PRINT "THE TOTAL IS", TOT
```

PRINT USING *format; data*[,*data*...]

Prints data using a format you specified. This statement is especially useful for printing report headings, accounting reports, checks, or any other documents that require a specific format. *Format* consists of 1 or more field specifier(s), or any alphanumeric character. *Format* must be enclosed in quotation marks. *Data* may be a string and/or numeric value(s).

Specifiers for String Fields:

!	prints only the first character in the string.
\spaces\	prints 2 + *n* characters from the string. (*N* is the number of spaces between the slashes.)
&	prints the string without modifications.

Specifiers for Numeric Fields:

#	prints the same number of digit positions as number signs (#). You may insert a decimal point at any position.
+	prints the sign of the number. The plus sign may be typed at the beginning or at the end of the format string.
–	prints a negative sign *after* negative numbers and a space after positive numbers.
**	fills leading spaces with asterisks.
$$	prints a dollar sign immediately before the number. You may not use exponential format with $$.

****$**	fills leading spaces with asterisks and prints a dollar sign immediately before the number.
,	prints a comma before every third digit to the left of the decimal point.
^^^^	prints in exponential format. The four exponent signs are placed after the digit position characters. You may specify any decimal point position.
_	prints the next character as a literal character.

```
PRINT USING ".####^^^^"; 888888
PRINT USING "**$###,.##"; 1234.5
PRINT USING "###2.#-"; -768.660
PRINT USING "###.##"; 876.567
```

PRINT# *buffer,*[USING *format*] *data*[,*data*,...]

Writes data items to a sequential access file. PRINT# does not compress the data before writing it to disk. It writes an ASCII-coded image of the data.

See PRINT USING for information about the *format* parameter.

```
PRINT# 1,A        PRINT# 1, B$, T$
```

PSET [STEP] (*x,y*)[,*color*]
PRESET [STEP] (*x,y*) [,*color*]

Graphics. Draws a point on the display. If you use PSET, *color* defaults to the foreground color. If you use PRESET, *color* defaults to the background color. (*x,y*) are the coordinates of the point. STEP designates (*x,y*) as relative coordinates.

```
PSET (1,1)        PRESET (1,1),0
```

PUT [#]*buffer*[,*record*]

Puts a *record* in a direct access file. The number sign (#) is not required. *Record* is the number of the record to be written to the file and may be in the range 1 to 16,777,215. Default = the next sequential record (after the last PUT).

```
PUT 1         PUT 1,25
```

PUT [#]*buffer,number*

Communications. Transfers data from the communications buffer to the communications line. The number sign (#) is not required. *Number* is the number of bytes to transfer.

```
PUT 2,80
```

PUT (*x,y*),*array*[*,action*]

Graphics. Transfers an image stored in an array to the screen. (*x,y*) are the coordinates at which the image begins (the upper left corner of the image). Default = last point referenced. *Array* is the array variable name that holds the image. *Action* sets the type of interaction between the transferred image and the image already on the screen. *Action* may be PSET, PRESET, AND, OR, or XOR. Default = PRESET.

```
PUT(200,100),A
```

RANDOMIZE[*number*]

Reseeds the random number generator. *Number* may be an integer, or single-or double precision number. If you omit *number*, BASIC suspends program execution and prompts you for a number before executing RANDOMIZE.

```
RANDOMIZE               RANDOMIZE 300
RANDOMIZE TIMER
```

READ *variable*[*,variable,...*]

Reads values from a DATA statement and assigns them to *variables.*

```
READ T                      READ N$, D$
```

REM

inserts a remark line in a program. You may use an apostrophe (') as an abbreviation for REM.

```
REM AVERAGE VELOCITY      'TOTALS
```

RENUM *[new line][,[line][,increment]]*

Renumbers the program currently in memory. RENUM also changes all line number references appearing after GOTO, GOSUB, THEN, ON/GOTO, ON/GOSUB, ON ERROR GOTO, RESUME, and ERL. *Line* is the line in the program at which BASIC starts renumbering. Default = first line. *New line* is the new line number assigned to *line*. Default = Line 10. *Increment* tells BASIC how to number the successive lines. Default = 10.

```
RENUM       RENUM 600, 5000, 100
```

RESET

Closes all open files on all drives.

```
RESET
```

RESTORE *[line]*

Restores a program's access to previously read DATA statements. *Line* specifies the DATA statement to be accessed at the next READ statement. Default = first DATA statement.

```
RESTORE
```

RESUME *[line]*
RESUME NEXT

Resumes program execution after an error-handling routine. RESUME *line* branches to the specified line number. Default = line in which the error occurred. RESUME NEXT branches to the statement following the point at which the error occurred.

```
RESUME     RESUME 10     RESUME NEXT
```

RETURN *[line]*

Returns control from a subroutine executed by a GOSUB to the specified *line*. Default = line immediately following the GOSUB.

```
RETURN     RETURN 40
```

RIGHT$(*string,number*)

Returns the specified number of characters from the far right portion of *string. Number* must be an integer in the range 1 to 255.

```
PRINT RIGHT$("WATERMELON",5)
PRINT RIGHT$("PUPPY",25)
```

RMDIR *"pathname"*

Removes (deletes) the directory specified by *pathname*. The directory being deleted must be empty except for the "." and ".." symbols. Use the MS-DOS ERASE command or the KILL statement to remove files from the directory.

```
RMDIR "NAMES"
RMDIR "A:\ACCTS\PAYABLE"
```

RND[(*number*)]

Returns a random number between 0 and 1. If *number* is negative, RND starts the sequence of random numbers at the beginning. If *number* is 0, RND repeats the last number generated.

```
PRINT RND(1)      A = RND(0)
```

RSET *field name* = *data*

Sets *data* in a direct access buffer *field name* in preparation for a PUT statement.

```
RSET A$ = CVI(Z)
```

RUN [*line*]
RUN *"pathname"*[,R]

Executes a program. *Line* is the program line at which BASIC begins execution. Default = first line. If you specify the R option, BASIC does not close the open files before loading the new program into memory. If you omit the R option, BASIC closes all open files before loading the program.

```
RUN     RUN 100     RUN "PROGRAM.A"
```

SAVE *"pathname"* [,A]
SAVE *"pathname"* [,P]

Saves a program on disk with the specified name. The A option saves the program in ASCII format. Default = compressed format. The P option saves the file in an encoded binary format. The only operations that can be performed on the file are RUN, LOAD, and CHAIN.

```
SAVE "A:FILE1.BAS"
SAVE "\EDUC\MATHPAK.TXT", A
```

SCREEN (*row, column,*[*number*])

Returns the ASCII code for the character at the specified row and column. *Row* is an integer in the range 1 to 25. *Column* is an integer in the range 1 to 40 or 1 to 80, depending on the screen width. If *number* is specified and is non-zero, BASIC returns the color number in the range 1 to 16 instead of the ASCII code of the character.

```
A = SCREEN(20,20)
PRINT SCREEN(10,10,1)
```

SCREEN [*mode*][,[*burst*][,[*active page*]
[,*display page*]] [,*erase*]]

Sets the screen attributes to be used by all other graphics statements. *Mode* is an integer in the range 0 to 6. *Burst* enables or disables color. In Screen Mode 0 (text mode), set *burst* to 0 to disable color or 1 to enable color. In Screen Modes 1 and 4, set *burst* to 0 to enable color or 1 to disable color. *Burst* has no effect in screen Modes 3, 5, and 6 where color is always enabled or in Screen Mode 2, which is black and white. *Active page* selects the video page to which BASIC will write. All output statements to the screen go to the selected *active page*. Default = Page 0 or current active page. *Display page* selects the video page for BASIC to display. Default = *active page*. *Erase* tells BASIC how much video memory to erase. *Erase* can be one of the following:

0	Do not erase video memory, even if the screen mode changes.
1	Erase the union of the new page and old page if *mode* or *burst* change. Default = 1.
2	Erase all video memory if *mode* or *burst* changes.

```
SCREEN 0,0        SCREEN 2
```

SGN(*number*)

Determines *number's* sign. If *number* is a negative number, SGN returns -1. If *number* is a positive number, SGN returns 1. If *number* is zero, SGN returns 0.

```
PRINT SGN(-55)     Y=SGN(A*B)
```

SHELL [*command*]

Advanced Statement. Loads and executes another program (*.EXE* or *.COM*) as a child process to the original program. After the child process ends, control returns to the BASIC program at the statement following the SHELL statement. *Command* is a string expression containing the name of the program you want to run.

```
SHELL
```

SIN(*number*)

Computes the sine of *number*. *Number* must be in radians.

```
PRINT SIN(7.96)     S=SIN(T)
```

SOUND *tone,duration*[,[*volume*][,[*voice*]]]
SOUND ON
SOUND OFF

Generates a sound with the *tone* and *duration* specified. While a SOUND statement is producing sound, the program continues to execute. *Tone* is an integer in the range 1 to 1023, indicating the frequency in hertz.

tone = CINT (3579545/32 *frequency)

Note	Frequency	Note	Frequency
Middle C	523.25	G	783.99
D	587.33	A	880.00
E	659.26	B	987.77
F	698.46	C	1046.50

Duration is an integer in the range 1 to 65535, specifying the duration in clock ticks. Clock ticks occur 18.2 times per second. *Volume* is an integer in the range 0 to 15, where 0 is the lowest volume and 15 is the highest volume. Default = 8. *Voice* is an integer in the range 0 to 2. Default = 0. SOUND ON enables the external speaker that supports multivoice sounds using the PLAY or SOUND statements. SOUND OFF disables the external speaker.

See also BEEP.

```
SOUND 20, 500, 6
```

SPACE$(*number*)

Returns a string of *number* spaces. *Number* must be in the range 0 to 255.

```
PRINT "COST" SPACE$(4) "QUANTITY"
 SPACE$(9) "TOTAL"
```

SPC(*number*)

Prints *number* blanks. *Number* is in the range 0 to 255.

```
PRINT "HELLO" SPC(15) "THERE"
```

SQR(*number*)

Returns the square root of *number. Number* must be greater than zero.

```
PRINT SQR(155.7)
```

STICK(*action*)

Returns the coordinates of the joysticks. *Action* may be one of the following:

0 returns the horizontal (x) coordinate for Joystick A.

1 returns the vertical (y) coordinate for Joystick A.

2 returns the horizontal (x) coordinate for Joystick B.

3 returns the vertical (y) coordinate for Joystick B.

```
STICK (2)      STICK (0)
```

STOP

Stops program execution.

```
STOP
```

STR$(*number*)

Converts *number* to a string.

```
S$ = STR$(x)     PRINT STR$(-234)
```

STRIG ON
STRIG OFF

Enables the STRIG function. STRIG ON lets you execute STRIG function statements to return the status of the joystick buttons. If you execute a STRIG OFF statement, you cannot execute STRIG function.

STRIG(*number*)

Returns the status of joystick buttons. *Number* is a number in the range 0 to 7 to test the status of the joystick buttons.

0 Tests to see if Trigger A1 has been pressed and released since the last STRIG(0) function was executed. BASIC returns a -1 if it has been pressed and a 0 if not.

1 Tests to see if you are currently pressing Trigger A1. BASIC returns a -1 if you are pressing it and a 0 if not.

2 Tests to see if Trigger B1 has been pressed and released since the last STRIG(2) function was executed. BASIC returns a -1 if it has been pressed and a 0 if not.

3 Tests to see if you are currently pressing Trigger B1. BASIC returns a -1 if you are pressing it and a 0 if not.

4 Tests to see if Trigger A2 has been pressed and released since the last STRIG(4) function was executed. BASIC returns a -1 if it has been pressed and a 0 if not.

5 Tests to see if you are currently pressing Trigger A2. BASIC returns a -1 if you are pressing it and a 0 if not.

6 Tests to see if Trigger B2 has been pressed and released since the last STRIG(6) function was executed. BASIC returns a -1 if it has been pressed and a 0 if not.

7 Tests to see if you are currently pressing Trigger B2. BASIC returns a -1 if you are pressing it and a 0 if not.

```
A = STRIG(0)      Z = STRIG(4)
```

STRIG(*number*) *action*

Turns on, turns off, or temporarily halts joystick trapping.

STRIG ON	enables joystick trapping.
STRIG OFF	disables joystick trapping.
STRIG STOP	temporarily halts joystick trapping.

Number is a value of 0, 2, 4 or 6 to indicate the joystick button you are trapping:

0 indicates Trigger A1.

2 indicates Trigger B1.

4 indicates Trigger A2.

6 indicates Trigger B2.

```
STRIG(0) ON        STRIG(6) OFF
```

STRING$(*number,character*)

Returns a string containing the specified number of *character. Number* must be in the range 0 to 255. *Character* is a string or an ASCII code.

```
B$ = STRING$(25,"X")
PRINT STRING$(50,10)
```

SWAP *variable1,variable2*

Exchanges the values of 2 variables of the same type.

```
SWAP F1#, F2#
```

SYSTEM

Returns you to the MS-DOS command level.

```
SYSTEM
```

TAB(*number*)

Spaces to position *number* on the display.

Number must be in the range 1 to 255.

```
PRINT "NAME" TAB(25) "AMOUNT":PRINT
```

TAN(*number*)

Returns the tangent of *number*. *Number* must be in radians.

```
PRINT TAN(7.96)     S = TAN(X)
```

TIME$[= *string*]

Sets or retrieves the current time. BASIC uses a 24-hour clock. *String* is a literal, enclosed in quotation marks, that sets the time by assigning its value to TIME$. If you omit *string*, BASIC retrieves the current time.

```
TIME$ ="14:15"   TIME$ ="3:3:3"
A$=TIME$         PRINT TIME$
```

TIMER

Returns the number of seconds since midnight or since the last system reset. You can use TIMER as the argument for the RANDOMIZE statement to reseed the random number generator.

```
PRINT TIMER      A = TIMER
```

TIMER *action*

Turns on, turns off, or temporarily halts timer event trapping.

TIMER ON	enables timer event trapping.
TIMER OFF	disables timer event trapping.
TIMER STOP	temporarily suspends timer event trapping.

TROFF
TRON

Turns the trace function on/off. The tracer lets you follow program flow. TRON turns on the tracer and TROFF turns it off.

```
TRON          TROFF
```

UNLOCK

3.20 See LOCK.

USR[*number*](*argument*)

Calls a user's assembly-language subroutine identified by *number* and passes *argument* to that subroutine. The *number* you specify must be the same as the corresponding DEF USR statement for that routine. Default = 0.

VAL(*string*)

Calculates the numerical value of *string*.

```
PRINT VAL("100")
PRINT VAL("1234E5")
```

VARPTR (*variable*)
VARPTR ([#]*buffer*)

Returns the offset into BASIC's data segment of a variable or a disk buffer. When used with *variable*, VARPTR returns the address of the first byte of data identified with *variable.* When used with *buffer*, VARPTR returns the address of the file's control block. The number sign (#) is not required.

```
PRINT VARPTR(3)     A = VARPTR(A$)
```

VARPTR$(*variable*)

Returns a 3-byte string representing a memory address of a variable:

Byte 0	=	*type*
Byte 1	=	*low byte of address*
Byte 2	=	*high byte of address*

Type is 2 for integer variables, 3 for string variables, 4 for single-precision variables, and 8 for double-precision variables.

```
A$ = VARPTR$(A!)
```

VIEW [SCREEN] [(*x1,y1*)-(*x2,y2*)[,[*color*][,[*border*]]]]

Graphics. Creates a rectangular viewport that redefines the screen parameters. This defined area, a window, becomes the only place in which you can draw graphics displays. (*x1,y1*) specifies the upper-left corner of the viewport. (*x2,y2*) specifies the lower-right corner of the viewport. SCREEN specifies that all coordinates used in drawing are absolute to point 0,0 on the screen. If you omit SCREEN, all coordinates specified are relative to the viewport coordinates.

```
VIEW (10,10)-(100,100)
VIEW SCREEN (20,25)-(100,150)
```

VIEW PRINT *top line* TO *bottom line*

Creates a text viewport that redefines the text screen parameters. *Top line* specifies the first line of the text viewport. It may be in the range 1 to 24, but must be less than *bottom line*. Default = Line 1. *Bottom line* specifies the last line of the text viewport. It may be in the range 1 to 24, but must be greater than *top line*. Default = Line 24.

```
VIEW 1 TO 15
```

WAIT *port, number1* [,*number2*]

Suspends program execution until a machine input *port* develops a specified bit pattern. *Number1* and *number2* are integers in the range 0 to 255.

```
WAIT 32,2
```

WHILE *expression*
WEND

Executes a series of statements in a loop as long as a given condition is true. If *expression* is true, BASIC executes the statements after the WHILE statement until it encounters a WEND statement. Then BASIC returns to the WHILE statement and checks *expression*. If it is still true, BASIC repeats the process. If it is not true, execution resumes with the statement following the WEND statement.

```
WHILE NUM
. . .
WEND
```

WIDTH [LPRINT] *size*
WIDTH *buffer, size*
WIDTH *"device:", size*

Sets the line width in number of characters for the display, printer, or communications channel. *Buffer* is the number assigned to the file in the OPEN statement. *Device*: is a valid device, enclosed in quotation marks, that specifies the device for which you are setting the width. It may be SCRN:, LPT1:, COM1:, or COM2:. *Size* may be an integer in the range 0 to 255 that specifies the number of characters in a line. For the screen, *size* may be only 40 or 80.

```
WIDTH 40          WIDTH LPRINT 100
WIDTH "SCRN:", 40
```

WINDOW [SCREEN] [(*x1*,*y1*)-(*x2*,*y2*)]

Lets you change the physical coordinates of the screen (or current viewport) by defining *world coordinates*. (*x1*,*y1*) are the world coordinates for the upper-left corner of the screen. (*x2*,*y2*) are the world coordinates for the lower-left corner of the screen. The SCREEN option tells BASIC to set the coordinates similar to the screen display in that the lesser y-coordinate is in the upper-left corner of the screen. If you omit SCREEN, BASIC inverts the y-coordinates to show a true Cartesian coordinate system. That is, the lesser y-coordinate is in the lower-left corner of the screen. WINDOW lets you plot points outside the normal screen coordinate limits by setting new world coordinates to the screen.

```
WINDOW (1984,100000)-(1987,300000)
```

WRITE *data*[*,data,...*]

Writes data to the screen.

```
WRITE D, B, V$
```

WRITE#*buffer, data*[*,data,...*]

Writes data to a sequential-access disk file.

```
WRITE#1, A$,B$
```

BASIC ERROR CODES AND MESSAGES

Error Number	Error Message
1	NEXT without FOR
2	Syntax error
3	Return without GOSUB
4	Out of DATA
5	Illegal function call
6	Overflow
7	Out of memory
8	Undefined line number
9	Subscript out of range
10	Redimensioned Array/Duplicate Definition
11	Division by zero
12	Illegal direct
13	Type mismatch
14	Out of string space
15	String too long
16	String formula too complex
17	Can't continue
18	Undefined user function
19	No RESUME
20	RESUME without error
21	Unprintable error
22	Missing operand
23	Line buffer overflow
24	Device Timeout
25	Device Fault
26	FOR without NEXT
27	Out of paper
29	WHILE without WEND
30	WEND without WHILE
50	FIELD overflow
51	Internal error
52	Bad file number
53	File not found
54	Bad file mode
55	File already open
57	Device I/O Error
58	File already exists
61	Disk full
62	Input past end
63	Bad record number
64	Bad file name
66	Direct statement in file
67	Too many files
68	Device Unavailable
69	Communication buffer overflow
70	Disk write protected
71	Disk not Ready

72	Disk media error
73	Advanced Feature
74	Rename across disks
75	Path/file Access Error
76	Path not found
77	Dead lock

RADIO SHACK
A Division of Tandy Corporation
Fort Worth, Texas 76102

08/86-BCo 8749624 Printed in U.S.A.