RADIO SHACK, A DIVISION OF TANDY CORPORATION

U.S.A.: FORT WORTH, TEXAS 76102
CANADA: BARRIE, ONTARIO L4M 4W5

TANDY CORPORATION

AUSTRALIA	BELGIUM	U. K.
91 KURRAJONG ROAD MOUNT DRUITT, N.S.W. 2770	PARC INDUSTRIEL DE NANINNE 5140 NANINNE	BILSTON ROAD WEDNESBURY WEST MIDLANDS WS10 7JN

874-9496 02/85-SP

Printed in U.S.A.

REFERENCE GUIDE FOR THE MODEL 2000

Reference Guide for
the Model 2000

Contents

Notes:

Loading MS-DOS

1. Turn on your Model 2000.
2. Insert the MS-DOS System Diskette into Drive A. Close the drive latch or door.
3. Press the RESET switch. MS-DOS loads into memory.
4. When the prompt Enter new date: appears, enter the current date in the *mm-dd-yyyy* format. (For example, enter 6-7-84 or 06-07-1984 for June 7, 1984.)
5. When the prompt Enter new time: appears, either press (ENTER) or enter the time in the 24-hour format, *hh:mm:ss.cc*. (For example, type 14:30 for 2:30 p.m.)

   MS-DOS displays the system prompt:

   A>

   Your Model 2000 is now ready for use.

Notes:

Notation:

UPPER CASE indicates keywords (material that you must type)

lower-case italics represent words, letters, characters, or values that you supply

[ ] (square brackets) indicate optional parameters

. . . (ellipsis) indicates that you may repeat a parameter as many times as you want

BACKUP [*pathname*] *drive* [/S] [/M] [/A] [/D:*mm/dd/yy*]

(*Hard disk only; external*) Copies one or more files from a hard disk to floppy disks.

pathname specifies the file to back up.

drive specifies the disk to receive the files.

/S copies all files in the specified directory and all directories below it.

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

/A adds the files to the diskette already in the specified drive, rather than prompting you to insert a new diskette.

/D:*mm/dd/yy* copies only those files created on or after the specified date.

```
BACKUP C:STORE1\sales.dat A:/A
```

BREAK [ON | OFF]

(*Internal*) Turns the CTRL C check on or off. Displays the current CTRL C setting if you omit ON and OFF.

```
BREAK OFF
```

CHDIR [*pathname*]
CD [*pathname*]

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

pathname specifies the directory you wish to change to.

```
CHDIR \BIN\USER          CHDIR B:\USER
```

CHKDSK [*drive*] [/F] [/V] [>*pathname*]

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

/F fixes errors (if possible) and updates disk. (Do not specify a *pathname*.)

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

pathname specifies the file to which CHKDSK is to redirect its output. (Do not use the /F parameter.)

```
CHKDSK B: \USER\TOM\errors
```

CLS
(*Internal*) Clears the screen.

```
CLS
```

COMPDUPE [/D] [/S]
(*Floppy disk only; external*) Makes a mirror-image copy of the floppy disk in Drive A onto the floppy disk in Drive B and compares the disks (duplication mode), or only compares the disks (comparison mode).
/D specifies duplication mode.
/S skips formatting of the Drive B disk during duplication.
Omitting /D and /S specifies comparison mode.

```
COMPDUPE      COMPDUPE /D
```

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 /A and /B, /B is used.
/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 file (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 /A B:corr.txt
```

COPY *target pathname* + *source pathname1* [+ *source pathname2* . . .] [/A] [/B] [/V]
(*Internal*) Adds one or more files to the end of another existing file. If you omit /A and /B, /A is used.
/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 file (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:read.dat + write.dat + print.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 /A and /B, /A is used.
/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 file (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:memos.txt + B:letters.txt B:corr.txt
```

CTTY *device*
(*Internal*) Changes the I/O device to the *device* specified.
device can be:
AUX specifies an auxiliary device.
CON specifies the console.

```
CTTY AUX
```

DATE [*mm/dd/yy*]
(*Internal*) Enters or changes the system date, or displays the current date.
mm/dd/yy specifies the month, day, and year to set as the date.

```
DATE 11/15/84
```

DEL
See ERASE.

DIR [*pathname*] [/P] [/W]
(*Internal*) Displays information about files that are in the current directory or in the directory specified by *pathname*, or information about the one file specified by *pathname*.
/P selects "page" mode.
/W selects a wide display.

```
DIR B:      DIR \USER\*.bat /P
```

DISKCOPY [*source drive*] [*target drive*]
(*External*) Copies the contents of the disk in the *source drive* to the disk in the *target drive*. The target disk must already be formatted.

```
DISKCOPY      DISKCOPY A: B:
```

ECHO [ON|OFF|*message*]
(*Internal*) Turns the batch ECHO feature on or off; displays a message; displays the current setting of ECHO if you omit all parameters.

```
ECHO OFF      ECHO  Insert disk.
```

ERASE [*pathname*]
DEL [*pathname*]
(*Internal*) Erases one or more files from the current directory or the directory specified by *pathname*. Omitting the filename or the entire *pathname* erases all files in the specified or current directory.

```
ERASE \BIN\USER\MARY\text.txt
```

EXE2BIN *source pathname [target pathname]*
(*External*) Converts an .exe file to .com file format.
source pathname specifies the .exe file.
target pathname specifies a file to receive the converted program file.

```
EXE2BIN testfile.exe B:
```

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

```
EXIT
```

FC [*/number*] [/B] [/W] [/C] *pathname1 pathname2* [*>target pathname*]
(*Utility*) Compares the contents of two files, *pathname1* and *pathname2*, and sends the output to the screen or to the file specified by *target pathname*.
/B forces a binary comparison of the files.
number specifies the number of lines that must match for the file to be considered as matching again after FC finds a difference (1 through 9; default = 3); use in source file comparisons only.
/W compresses tabs and spaces; use in source file comparisons only.
/C interprets all letters in the file as upper case; use in source file comparisons only.

```
FC /B test1.src test2.src >test3.src
```

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* among the lines on the current screen display if you omit *pathname*.
/V displays all lines that do not contain the *string*.
/C displays only the number of lines in each file that contain the *string*.
/N displays each line's relative line number in the file; do not use with /C.

```
FIND /N "mispell" *.txt
```

FOR %f IN (*set*) DO *command* %f
(regular command)
FOR %%f IN (*set*) DO *command* %%f
(batch file command)
(*Internal*) Executes the specifed *command* for each item in the *set*.
set is a list of items separated by spaces, or one wildcard item.

```
FOR %f IN (taxfile autofile homefile) DO DEL %f
```


FORMAT [*drive*] [/S] [/V] [/P]
(*Floppy disk only; external*) Prepares the blank floppy disk in the specified *drive* for use.
drive can be A: or B:.
/S copies the system files to the disk.
/V prompts for a volume label.
/P prompts for the skew and interleave factors.

```
FORMAT      FORMAT B: /S /V
```

GOTO *label*
(*Internal*) Transfers control to the line in the batch file that follows the one that contains *:label*.
label is a character string.

```
:G
REM looping . . .
GOTO G
```

HFORMAT [*drive*] [/S] [/V] [/P]
(*Hard disk only; external*) Prepares a hard disk for use.
drive can be C: or greater.
/S copies the system files to the disk.
/V prompts for a volume label.
/P prompts for the interleave factor and the number of cylinders and heads.

```
HFORMAT /S      HFORMAT D: /P /V
```

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 the file specified by *filename* exists.

```
IF EXIST All.lst GOTO G
```

MKDIR *pathname*
MD *pathname*
(*Internal*) Makes a new directory.
pathname tells MS-DOS under which directory to create the new directory and specifies the name to give it.

```
MKDIR \USER      MD B:LETTERS
```

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
```

PATH [*pathname*[;*pathname*] . . .]
(*Internal*) Sets a command path, which tells MS-DOS the directories or drives in which to search for external commands. Displays the current path setting if you omit *pathname*.
pathname specifies a directory or an entire drive.

```
PATH \BIN\USER\JOE
```

PAUSE [*comment*]
(*Internal*) Suspends execution of the batch file.
comment is a message to be displayed when the file pauses.

```
PAUSE Insert disk
```

PRINT [*pathname* [/T][/C][/P]] . . .
(*External*) Puts up to 10 files in the print queue for background printing.
/T deletes (terminates) all files from the print queue.
/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.

```
PRINT /T
PRINT temp1.tst/C temp2.tst/P temp3.tst
```

PROMPT [*prompt-text*]
(*Internal*) Changes the system prompt to *prompt-text*. Sets the prompt to the current drive specification if you omit *prompt-text*.
prompt-text is a string of characters to set as the prompt.

```
PROMPT $n$g
```

RECOVER *pathname*
RECOVER *drive*
(*External*) Recovers a file that contains bad sectors or recovers all files on a disk that contains bad sectors in its directory.
pathname specifies the file to recover.
drive specifies the disk to recover.

```
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:\USER\GL1.dat GL2.dat
```

RESTORE *drive* [*pathname*] [/S] [/P]
(*Hard disk only; external*) Restores one or more files from diskettes to a hard disk. The files must have been stored on diskette using the BACKUP command.
drive specifies the drive containing the backup files.
pathname specifies the hard disk file you want to restore.
/S restores all files in the specified directory and all directories below it.
/P prompts before restoring files that have been changed since the last backup and all read-only files.

```
RESTORE A: C:*.dat/P
```

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

```
RMDIR \BIN\USER\JIM
```

SET [*parameter*=[*replacement parameter*]]
(*Internal*) Sets the *parameter* equal to the *replacement parameter* for use in later programs and batch files. If you omit *parameter* = *replacement parameter*, SET returns the values that are set.
parameter is any character string (except the numbers 0 through 9).
replacement parameter is a specific parameter, which can be any character string (except the numbers 0 through 9). If omitted, MS-DOS voids the current setting of *parameter*.

```
SET drive=B:      SET drive=
```

SHIFT
(*Internal*) Shifts all parameters that replace the batch file parameters %0 through %9 down one place.

```
SHIFT
```

SORT [/R] [/+*n*] [<*input pathname*] [>*output pathname*]
(*External*) Reads input from the keyboard or a file, sorts the data, and writes it to the display or a file.
<*input pathname* specifies the file to be sorted. If omitted, keyboard input is sorted.
>*output pathname* specifies the file to receive the sorted information. If omitted, output is sent to the display.

/R reverses the sort (sorts from Z to A).
/+*n* begins the sort at Column *n*. If omitted, the sort begins at Column 1.

```
SORT /R <unsort.txt >sort.txt
```

SYS *drive*
(*External*) Transfers the system files IO.SYS and MSDOS.SYS from the current disk to the disk in the specified *drive*.

```
SYS B:
```

TIME [*hh:mm:ss.cc*]
(*Internal*) Displays or sets the time.
hh:mm:ss.cc specifies the time to set. If you omit the time, the current system time is displayed.

```
TIME 14:30
```

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

```
TYPE B:carfile
```

VER
(*Internal*) Displays the number of the MS-DOS version that you are using.

```
VER
```

VERIFY [ON|OFF]
(*Internal*) Turns the verify switch on or off (VERIFY ON verifies all disk writes). Displays the current VERIFY setting if you omit ON and OFF.

```
VERIFY OFF
```

VOL [*drive*]
(*Internal*) Displays the volume label of the disk in the specified *drive*, or the current drive if you omit *drive*.

```
VOL B:     VOL
```


MS-DOS Editing Keys

Function	Key(s)	Description
Copy *char*	→	Copies one character from template to command line and displays it.
Delete *char*	DELETE	Deletes a character from template.
Copy to *char*	F2 *char*	Copies and displays all characters up to specified character.
Delete to *char*	F4 *char*	Deletes all characters up to specified character from template.
Copy all	F3	Copies remaining characters and displays command line.
Insert	INSERT	Enters insert mode. (F3 ends insert mode.)
Replace template	F5	Makes the new line the new template; does not send it to the requesting program.
Void line	F8 or CTRL X	Voids current input, leaves template unchanged.
Enter line	ENTER	Makes the new line the new template; sends it to the requesting program.
End-of-file	F6 or CTRL Z	Puts an end-of-file character in the new or CTRL Z template.

Notes:

Control Character Keys

Key(s)	Function
CTRL C	Stops execution of a command.
SHIFT PRINT	Sends the current screen contents to the printer.
PRINT or CTRL P	Sends all output to the printer and the screen.
CTRL N	Toggles echoing of output to printer.
CTRL H or BACKSPACE	Removes last character from command line.
CTRL J	Inserts a physical end-of-line.
HOLD or CTRL S	Suspends the screen. Press HOLD or CTRL Q to continue scrolling.
F8 or CTRL X	Voids the current line and empties the command line.

Notes:

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 ranging from *line1* to *line2*, placing 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 within the range *line1* to *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*; enters lines into a new file.

3I .I #I

List
[*line1*][,*line2*]L
Displays all lines in the range *line1* to *line2*.

2,5L .,26L

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)

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* into the file being edited, just ahead of the specified *line* or current line.

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 lines until 25% of memory is freed.

100W

EDLIN Editing Keys

Function	Key(s)	Description
Copy char	(→)	Copies one character to the new line.
Copy to *char*	(F2)*char*	Copies all characters up to the specified character to the new line.
Copy all	(F3)	Copies all remaining characters in the template to the new line.
Delete char	(DELETE)	Deletes a character in the template.
Delete to *char*	(F4)*char*	Deletes all characters up to the character specified.
Void line	(F8) or (CTRL)(X)	Voids the current input.
Insert	(INSERT)	Enters/exits the insert mode.
Replace template	(F5)	Replaces the template with the characters displayed to allow further editing.
Enter line	(ENTER)	Makes the new line the new template and sends it to the requesting program.

Assemble

A [*address*]

Assembles statements directly into memory.

address is the starting address at which the instructions are to be assembled in memory.

A CS:0100

Compare

C *range address*

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

C 100,1FF 300 C 100L100 300

Dump

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

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

D CS:100 109

Enter

E *address* [*list*]

Enters byte values into memory at the specified *address*; replaces the contents of memory beginning at *address* 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, beginning at *address1* and stopping at each breakpoint (specified by the optional *addresses*).

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 byte from the specified port.

I 2F8

Load
L [*address*[*drive sector sectorcount*]]
Loads a file from the *drive* into memory, beginning at the specified *address*; loads absolute sectors from the *drive*, beginning at *sector* and continuing until *sectorcount* number of sectors have been 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 the filespec specified by *filespec1* for a later Load or Write command; assigns filespec parameters to 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

Quit
Q
Ends the DEBUG program without saving the file.

Q

Register
R [*registername*]
Displays the contents of all registers and flags; displays a single register and lets you change the contents; displays the flags and lets you change the settings.

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 (indicated by *value*), beginning at *address*, and 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 specified *range*), and displays their addresses, their hexadecimal values, and the source statements that correspond to them.

U04BA:100 L 10

Write
W [*address*[*drive sector sectorcount*]]
Writes the data being debugged, starting at the *address*, to a disk file; writes absolute sectors to the specified *drive*, beginning at *sector* and continuing until *sectorcount* number of sectors have been written.

W CS:100 1 37 2B

Notes:

DEBUG Command Parameters

Parameter	Description
address	• Alphabetic segment register + offset CS:0100 • Segment address + offset 04BA:0100 • Offset only (default segment is used)
byte	1- or 2-character hex value
drive	1-digit value: 0 = Drive A 1 = Drive B 2 = Drive C 3 = Drive D
filespec	File specification (drive specification, filename, and filename extension). You must specify at least the drive or filename.
list	A series of strings or byte values. CS:100 FF 42 "XXX" 1A 3
portaddress	A hex value of up to four characters.
range	• *address1 address2* *address2* must be an offset. CS:100 110 • *address* L *value* Do not use if another hex value follows *range.* *value* = number of bytes to operate on. If L *value* is omitted, 80 bytes is assumed. CS:100 L 10 CS:100
registername	One of the following: AX SP DS IP BX BP ES PC CX SI SS F DX DI CS
sector *sectorcount*	1- to 3-character hex values. *sector* = relative sector number. *sectorcount* = number of sectors to write or load.
string	Any number of characters enclosed in quotation marks.
value	A hex value of up to 4 characters.

Notes:

Loading BASIC

At the MS-DOS system prompt A>, load BASIC by typing:

BASIC (ENTER)

When the BASIC prompt Ok appears on your screen, you may begin using BASIC.

You can specify options with either form of BASIC. Use the following format:

BASIC [*filename*] [/F:*number of files*] [/C:*buffer size*] [/M:*highest memory location*] [/S:*record length*]

filename	a program to run after BASIC is loaded
number of files	the maximum number of data files that may be open at any one time (0-15; default = 3 files)
buffer size	the size of the receive buffer for RS-232 communication (default = 256 bytes)
highest memory location	the highest memory location for BASIC to use (default = 64K bytes allocated for BASIC)
record length	the maximum size for direct access files (default = 128 bytes)

Notes:

BASIC Commands and Functions

ABS(*number*)
Computes the absolute value of *number*.

```
X = ABS(Y)
```

ASC(*string*)
Returns the ASCII code for the first character of *string*.

```
PRINT ASC("A")
```

ATN(*number*)
Computes the arctangent of *number* in radians.

```
X = ATN(Y/3)
```

AUTO [*line*][,*increment*]
Automatically generates a line number every time you press (ENTER).

```
AUTO        AUTO 100, 50
```

BEEP
Produces a sound from the computer speaker.

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

BLOAD *filespec*[,*offset*]
Loads a memory image file into memory.

BSAVE *filespec, offset, length*
Saves the contents of an area of memory as a disk file.

CALL *variable* [(*parameter list*)]
Transfers control to an assembly-language subroutine stored at *variable*.

```
CALL MYROUT(I,J,K)
```

CDBL (*number*)
Converts *number* to double precision.

```
PRINT CDBL(A)
```

CHAIN [MERGE] *filespec* [,*line1*] [,ALL] [,DELETE *line2-line3*]
Loads a BASIC program named *filespec*, chains it to a "main" program, and begins running it.

```
CHAIN "PROG2.BAS"
```

CHR$(*code*)
Returns the character corresponding to an ASCII or control *code*.

```
PRINT CHR$(35)
```

CINT(*number*)
Converts *number* to integer representation.

```
PRINT CINT(17.65)
```

CIRCLE [STEP] (*x-coordinate, y-coordinate*),*radius* [,*color,start,end,aspect*]
Draws an ellipse with the specified center and radius.

CIRCLE (200,200),50

CLEAR [,*memory location*] [,*stack space*]
Clears the value of all variables and CLOSEs all open files.

CLEAR CLEAR, 45000

CLOSE [*buffer, . . .*]
Closes access to a file.

CLOSE CLOSE 3,4

CLS
Clears the screen.

CLS

COLOR/Text
COLOR [*foreground*][,*background*][,*border*]
Selects the display colors for the foreground, background, and border.

COLOR 0,7

COLOR/Graphic
COLOR [*background*][,*palette*]
Selects the *palette* of colors to be used by subsequent graphics statements.

COLOR 9,0

COM(1) *action*
Turns on, turns off, or temporarily halts the trapping of activity on the communications channel.

COM(1) ON COM(1) OFF COM(1) STOP

COMMON *variable, . . .*
Passes *variables* to a CHAINed program.

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

CONT
Resumes program execution.

CONT

COS (*number*)
Computes the cosine of *number*.

Y = COS(X*.01745329)

CSNG (*number*)
Converts *number* to single precision.

PRINT CSNG(.8944737829)

[*variable*] = **CSRLIN**
Returns the current row position of the cursor.

R = CSRLIN

CVD (*eight-byte string*)
CVS (*four-byte string*)
CVI (*two-byte string*)
Convert string values to numeric values.

A# = CVD(GROSSPAY$)

DATA *constant, . . .*
Stores numeric and string *constants* to be accessed by a READ statement.

DATA 2, 3, 0, 57.2

variable = **DATE$**
DATE$ = *string*
Sets or retrieves the current date.

DATE$ = "1/10/84" X$ = DATE$

DEFDBL *letter, . . .*
DEFINT *letter, . . .*
DEFSNG *letter, . . .*
DEFSTR *letter, . . .*
Defines any variables beginning with *letter*(s) as: (DBL) double precision, (INT) integer, (SNG) single precision, or (STR) string.

DEFDBL L-P DEFSTR A

DEF FN*function name* [(*argument, . . .*)] = *function definition*
Defines *function name* according to your *function definition*.

DEF FNR = RND(90) + 9

DEF SEG[= *address*]
Assigns the current segment address.

DEF SEG = &HB800 DEF SEG

DEF USR[*digit*] = *offset*
Defines the segment offset and user number of a subroutine to be called by the USR function.

DEF USR3 = &H0020

DELETE *line1-line2*
Deletes from *line1* through *line2* of a program in memory.

DELETE 50-110 DELETE .

DIM *array (dimension(s)), array (dimension(s)), . . .*
Sets aside storage for *arrays* with the *dimensions* you specify.

```
DIM AR(100)
```

DRAW "*direction [number] . . .*"
Draws an object on the video display.

```
DRAW "L40 E20 F20"
```

EDIT *line*
Enters the edit mode so that you can edit *line*.

```
EDIT 100
```

END
Ends program execution and closes all files.

```
END
```

EOF(*buffer*)
Detects the end of a file.

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

EOF(*variable*)
Detects an empty input queue for communications files.

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

ERASE *array, . . .*
Erases one or more *arrays* from a program.

```
ERASE C,F
```

ERL
Returns the line number in which an error occurred.

```
PRINT ERL
```

ERR
Returns the error code.

```
IF ERR = 7 THEN 1000
```

ERROR *code*
Simulates a specified error during program execution.

```
ERROR 1
```

EXP(*number*)
Calculates the natural exponent of *number*.

```
PRINT EXP(-2)
```


FIELD *buffer, length* AS *field name, . . .*
Divides a direct-access *buffer* into one or more fields. Each field is identified by *field name* and is the *length* you specify.

```
FIELD 3, 50 AS A$, 50 AS B$
```

FILES [*filespec*]
Displays the names of the files on a diskette.

```
FILES          FILES"*.BAS"
```

FIX(*number*)
Returns the truncated integer of *number*.

```
PRINT FIX(2.6)
```

FOR/NEXT
FOR *variable = initial value* TO *final value* [STEP *increment*]
NEXT [*variable*]
Establishes a program loop.

```
20 FOR H=1 TO 10 STEP 2
30      PRINT H
40 NEXT H
```

FRE (*dummy number*) or (*dummy string*)
Returns the number of bytes in memory not being used by BASIC.

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

GET *buffer*[, *record*]
Reads a record from a direct-access disk file and places it in a *buffer*.

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

GET *buffer, integer*
Transfers data from the communications buffer to the file buffer.

GET [STEP] (*x-coordinate1, y-coordinate1*) - (*x-coordinate2, y-coordinate2*), *array*
Transfers points from an area on the display to an array.

GOSUB *line*
Branches to a subroutine, beginning at *line*.

```
GOSUB 1000
```

GOTO *line*
Branches to the specified *line*.

```
GOTO 100
```

HEX$(*number*)
Calculates the hexadecimal value of *number*.

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

IF . . . THEN . . .ELSE
IF *expression* THEN *statement(s)* or *line* [ELSE *statement(s)* or *line*]
Tests a conditional expression and makes a decision regarding program flow.

```
IF X>127 THEN PRINT "OUT OF RANGE"
```

INKEY$
Returns a keyboard character.

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

INP(*port*)
Returns the byte read from a *port*.

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

INPUT[;][*"prompt string"*;] *variable1, variable2, . . .*
Inputs data from the keyboard into one or more *variables*.

```
INPUT Y%
```

INPUT#*buffer, variable, . . .*
Inputs data from a sequential disk file and stores it in a program *variable*.

```
INPUT#1, A,B
```

INPUT$(*number1*[,*number2*])
Inputs a string of characters from either the keyboard or a sequential disk file.

```
A$ = INPUT$(5)
```

INSTR([*integer,*] *string1, string2*)
Searches for the first occurrence of *string2* in *string1*, and returns the position at which the match is found. *Integer* is a position in *string1* to begin searching.

```
INSTR(A$, "INC")
```

INT(*number*)
Converts *number* to integer value.

```
PRINT INT(79,89)
```

KEY/Set/Display
KEY *integer, string*
Assigns or displays function key values. *Integer* is the number of the function key being defined. *String* is the string expression assigned to the key.

KEY ON
Displays the function key assignment values.

KEY OFF
Erases the soft key assignments from the display.

KEY LIST
Displays the complete soft key assignments on the screen.

KEY/Trap
KEY (*number*) *action*
Turns on, turns off, or temporarily halts key trapping for a specified function key or cursor direction key.

```
KEY (3)ON     KEY (2)OFF     KEY ( )STOP
```

KILL *filespec*
Deletes *filespec* from the disk.

```
KILL "A:DATA"
```

LEFT$(*string,integer*)
Returns the leftmost *integer* characters of *string*.

```
B$ = LEFT$(A$,3)
```

LEN(*string*)
Returns the number of characters in *string*.

```
x = LEN(S$)
```

[**LET**] *variable = expression*
Assigns the value of *expression* to *variable*.

```
LET B = 3
```

LINE [STEP] [(*x-coordinate1, y-coordinate1*)]-(*x-coordinate2, y-coordinate2*)[,*color*][,B[F]]
Draws a line or a box on the video display.

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

LINE INPUT[;][*"prompt message"*;]*string variable*
Inputs an entire line from the keyboard.

```
LINE INPUT "LAST NAME?";N$
```

LINE INPUT#*buffer, variable*
Inputs an entire line of data from a sequential disk file to a string *variable*.

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

LIST [*startline*]-[*endline*][,*device*]
Lists a program in memory to the display.

```
LIST          LIST 227-,"LPT1:"
```

LLIST [*startline*]-[*endline*]
Lists program lines in memory to the printer.

```
LLIST
```

LOAD *filespec*[,R]
Loads a BASIC program into memory. R tells BASIC to run the program.

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

LOC(*buffer*)
Returns the current record number.

```
IF LOC(1)>55 THEN END
```

LOC(*variable*)
Returns the number of characters in the input queue.

```
LOC(X)
```

LOCATE [*row*][,*column*][,*cursor*] [,*start*][,*stop*]
Positions the cursor on the screen.

```
LOCATE 10,20,1,4
```

LOF(*buffer*)
Returns the length of the file in bytes.

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

LOF(*variable*)
Returns the amount of free space in the input queue.

LOG(*number*)
Computes the natural logarithm of *number*.

```
PRINT LOG(3.14159)
```

LPOS(*number*)
Returns the logical position of the print head within the line printer's buffer.

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

LPRINT *data, . . .*
LPRINT USING *format; data, . . .*
Prints *data* on the printer.

```
LPRINT TAB(50) "TABBED 50"
```

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

```
LSET NM$ = "PAT"
```


MERGE *filespec*
Loads a BASIC program and merges it with the program currently in memory.

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

MID$(*oldstring, position*[,*length*]) = *replacement string*
Replaces a portion of an *oldstring* with *replacement string*.

```
MID$(A$, 1, 3) = "***"
```

MID$(*string, integer*[,*number*])
Returns a substring of a *string*. *Integer* is the position in the string to begin returning characters from. *Number* is the number of characters in the substring.

```
F$ = MID$(A$,3)
```

MKI$(*integer expression*)
MKS$(*single-precision expression*)
MKD$(*double-precision expression*)
Convert numeric values to string values.

```
LSET AVG$ = MKS$(0.123)
```

NAME *old filespec* AS *new filespec*
Renames *old filespec* as *new filespec*.

```
NAME "FILE.BAS" AS "FILE.OLD"
```

NEW
Deletes the program currently in memory and clears all variables.

```
NEW
```

OCT$(*number*)
Computes the octal value of *number*.

```
Y$ = OCT$(X/84)
```

ON COM(1) GOSUB *line number*
Transfers program control to a subroutine beginning at *line number* when activity occurs on the communication channel.

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

ON ERROR GOTO *line*
Transfers control to *line* if an error occurs.

```
ON ERROR GOTO 1500
```

ON . . . GOSUB
ON *number* GOSUB *line1, line2, . . .*
Branches to a subroutine at the *line* specified by the value of *number*.

```
ON Y GOSUB 10, 20, 30
```

ON . . . GOTO
ON *number* GOTO *line1, line2, . . .*
Goes to the *line* specified by the value of *number*.

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

ON KEY GOSUB
ON KEY(*number*) GOSUB *line number*
Transfers program control to a subroutine when you press a function key or a cursor direction key.

```
ON KEY(1) GOSUB 1000
```

ON STRIG (*integer*) GOSUB *line number*
Branches to a subroutine when you press the specified mouse button.

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

OPEN *mode, buffer, filespec*[*, record length*]
OPEN *filespec* [FOR *mode*] AS *buffer* [LEN=*record length*]
Establishes an input/output path for a file or device.

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

OPEN "COM1:[*speed*][,*parity*][,*data*][,*stop*][,RS][,CS[*seconds*]][,DS[*seconds*]][,CD[*seconds*]][,*mode*][,LF]" AS [*buffer*][LEN = *number*]
Opens a file and allocates a buffer for RS-232C communication.

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

OPTION BASE *n*
Sets *n* as the minimum value for an array subscript.

```
OPTION BASE 1
```

OUT *port, data byte*
Sends a *data byte* to a machine output *port*.

```
OUT 32,100
```

PAINT [STEP] (*x-coordinate,y-coordinate*) [*color*[,*border*]]
Fills in an area on the display with a selected color.

PALETTE [*position number, new color*]
Changes one of the colors in the current palette.

```
PALETTE 1,5
```

PALETTE USING *array name* (*subscript*)
Changes more than one of the color numbers in the current palette.

```
PALETTE USING A(4)
```

PEEK (*memory location*)
Returns a byte from *memory location*.

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

PLAY *string*
Plays musical notes specified by *string*.

```
PLAY "C4F4.C8F8.C16F8.G16A2F2".
```

POINT (*x-coordinate, y-coordinate*) = *variable*
Returns the color number of a point on the screen.

```
IF POINT(1,1)<>0 THEN PRESET (1,1)
```

POKE *memory location, data byte*
Writes *data byte* into *memory location*.

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

POS(*number*)
Returns the position of the cursor. (*Number* is a dummy argument.)

```
PRINT TAB(40) POS(0)
```

PRINT *data, . . .*
Prints numeric or string *data* on the display.

```
PRINT "HELLO"     PRINT N
```

PRINT USING *format*; *data item, . . .*
Prints *data items* using a *format* specified by you.

```
PRINT USING "!"; "PERSONNEL"
```

PRINT TAB(*n*)
Moves the cursor to the *n* position on the current line.

```
PRINT TAB(5) "TABBED 5"
```

PRINT# *buffer, item1, item2, . . .*
Prints data *items* in a sequential disk file.

```
PRINT# 1,A
```

PSET [STEP] (*x-coordinate, y-coordinate*) [,*color*]

PRESET [STEP] (*x-coordinate, y-coordinate*) [,*color*]
Draws a point on the display.

```
PSET(1,10)
```

PUT *buffer* [,*record*]
Puts a *record* in a direct-access disk file.

```
PUT 1,25
```

PUT *buffer, integer*
Transfers data from the file buffer to the communications buffer.

```
PUT 4,80
```

PUT/Graphics
PUT (*x-coordinate, y-coordinate*),*array* [,*action*]
Transfers an image stored in an array onto the screen. *Action* may be PSET, PRESET, AND, OR, or XOR.

RANDOMIZE
Reseeds the random number generator.

```
RANDOMIZE
```

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

```
READ T
```

REM
Inserts a remark line in a program.

```
REM CALCULATE SUM
```

RENUM [*new line*] [,*line*] [,*increment*]
Renumbers a program, starting at *line*, using *new line* as the first new line and *increment* for the new sequence.

```
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.

```
RESTORE
```

RESUME [*line*]
RESUME NEXT
Resumes program execution after an error-handling routine.

```
RESUME 10
```


RETURN [*line number*]
Returns control to the line immediately following the most recently executed GOSUB.

```
RETURN 50
```

RIGHT$(*string, number*)
Returns the rightmost *number* characters of *string*.

```
PRINT RIGHT$("ABCDEFG",3)
```

RND(*number*)
Generates a pseudorandom number.

```
PRINT RND(0)
```

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

```
RSET NM$ = "PAT"
```

RUN [*line*]
RUN *filespec*[,R]
Executes a program. R leaves all OPEN files open.

```
RUN      RUN "PROG"
```

SAVE *filespec* [,A][,P]
Saves a program in a disk file under *filespec*.

```
SAVE "A:FILE1.BAS"
```

variable = **SCREEN** (*row, column*)[,1]
Returns the ASCII code or the color attribute for the character at the specified row and column.

```
A = SCREEN (20,20)
```

SCREEN *mode*[, *burst*]
Sets the screen, attributes to be used by all other graphics statements.

```
SCREEN 0,1
```

SGN(*number*)
Determines *number's* sign.

```
Y = SGN(A*B)
```

SIN(*number*)
Computes the sine of *number*.

```
PRINT SIN(7.96)
```

SOUND *tone, duration*
Generates a sound with the *tone* and *duration* specified.

```
SOUND 37,2
```

SPACE$(*number*)
Returns a string of *number* spaces.

```
PRINT SPACE$(20) "HELLO"
```

SPC(*number*)
Prints *number* blanks.

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

SQR(*number*)
Calculates the square root of *number*.

```
Z = SQR(R)
```

STICK (*integer*) = *variable*
Returns the number of points moved along the x axis (*integer* = 0) or y axis (*integer* = 1) since the last STICK statement.

```
STICK(0) = XMOVE
```

STOP
Stops program execution.

```
STOP
```

STR$(*number*)
Converts *number* into a string.

```
S$ = STR$(X)
```

STRIG/Function Enable
STRIG ON
STRIG OFF
Enables and disables the STRIG/Function statements, which return the status of the mouse buttons.

STRIG/Trap Enable
STRIG(*integer*) ON
STRIG(*integer*) OFF
STRIG(*integer*) STOP
Turns on, turns off, or temporarily halts mouse button trapping.

```
STRIG(0) ON
```

variable = **STRIG** *integer*
Returns the status of mouse buttons.

```
S1 = STRIG(1)
```

STRING$(*number, character*)
Returns a string of *number* of characters.

```
B$ = STRING$(25,"X")
```


SWAP *variable1, variable2*
Exchanges the values of two variables.

```
SWAP A$, B$
```

SYSTEM
Returns you to MS-DOS level.

```
SYSTEM
```

TAB(*number*)
Spaces to position *number* on the display.

```
PRINT A$ TAB(25) B$
```

TAN(*number*)
Computes the tangent of *number*.

```
PRINT TAN(7.96)
```

variable = **TIME$**
TIME$ = *string*
Sets or retrieves the current time.

```
TIME$ = "14:15"
```

TROFF
TRON
Turn the trace function on and off.

```
TRON    TROFF
```

USR[*digit*](*argument*)
Calls a user's assembly-language subroutine identified with *digit* and passes *argument* to that subroutine.

VAL(*string*)
Calculates the numerical value of *string*.

```
PRINT VAL("100DOLLARS")
```

VARPTR(*variable*)
VARPTR(*#buffer*)
Returns the offset into BASIC's data segment of a variable or the file control block.

VARPTR$(*variable*)
Returns a character form of the address of a *variable* in memory.

```
DRAW "X" + VARPTR$(A$)
```

WAIT *port, integer1*[,*integer2*]
Suspends program execution until a machine input *port* develops a specified bit pattern.

```
WAIT 32,2
```

WHILE . . . WEND
WHILE *expression*
.
.
.
{loop statements}
.
WEND
Execute a series of statements in a loop as long as a given condition is true.

WIDTH [LPRINT] *size*
WIDTH *buffer, size*
WIDTH *device, size*
Sets the line width in number of characters for the display, line printer, or communication channel.

```
WIDTH #1, 40      WIDTH "SCRN:",40
```

WRITE [*data, . . .*]
Writes *data* on the display.

```
WRITE D, B, V$
```

WRITE# *buffer, data, . . .*
Writes *data* to a sequential-access file.

```
WRITE#1, A$,B$
```


BASIC Special Keys

Command Mode

BACKSPACE or CTRL H	Backspaces and erases.
SPACEBAR	Types a space.
BREAK or CTRL C	Starts new line.
CTRL J	Line feed.
CAPS	Switches to upper case or upper/lower case.
ENTER or CTRL M	Ends current logical line.
CTRL U or ESC	Erases the current line.
CTRL ← or ←	Moves cursor one position left.
CTRL → or →	Moves cursor one position right.
CTRL F or CTRL →	Moves cursor to next word.
CTRL B or CTRL ←	Moves cursor to previous word.
CTRL R or INSERT	Turns insert mode on and off.
DELETE	Deletes character at cursor position.
CTRL W	Deletes next word.
CTRL T	Displays soft key values of Function Keys.
END or CTRL N	Moves cursor to end of logical line.
CTRL END or CTRL E	Deletes to end of line.
CTRL I or TAB	Advances cursor to next tab position.
CTRL G	Rings the bell.

Execution Mode

HOLD or CTRL S	Pauses execution.
BREAK	Terminates execution, returns to command mode.
ENTER or CTRL M	Interprets data from keyboard as a response to INPUT statement.

Edit Mode

(ENTER) or (CTRL)(M)	Records changes to current line.
(←) or (CTRL)(J)	Moves cursor one position left.
(→) or (CTRL)(\)	Moves cursor one position right.
(SPACEBAR)	Changes character to a blank.
(CTRL)(→) or (CTRL)(F)	Moves cursor to next word.
(CTRL)(←) or (CTRL)(B)	Moves cursor to previous word.
(END) or (CTRL)(N)	Moves cursor to end of logical line.
(DELETE)	Erases character at cursor position.
(BACKSPACE) or (CTRL)(H)	Erases character to the left of cursor.
(CTRL)(END) or (CTRL)(E)	Erases to end of logical line.
(CTRL)(U) or (ESC)	Erases entire logical line from the screen and cancels changes to current line.
(TAB) or (CTRL)(I)	Moves cursor to next tab position, printing tabbed-over characters.
(BREAK) or (CTRL)(C)	Returns to direct mode and cancels changes to current line.
(CTRL)(K) or (HOME)	Moves cursor to upper left corner.
(CTRL)(L)	Clears screen and moves cursor to upper left corner.
(INSERT)	Inserts characters within line.
(CTRL)(Z)	Clears to end of screen.

BASIC Error Numbers and Messages

Number	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	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

Disk Errors

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 Protect
71	Disk not Ready
72	Disk Media Error
74	Rename across disks

Notes:

Keyboard ASCII and Scan Codes

x Values preceded by "x" are extended ASCII codes (codes preceded by an ASCII NUL, 00).

-- No ASCII code is generated.

** No ASCII code is generated, but the special function described in parentheses is performed.

Scan Code	Keyboard Legend	ASCII Codes Normal	SHIFT	CTRL	ALT
01	ESC	1B	1B	1B	x8B
02	! 1	31	21	--	x78
03	@ 2	32	40	x03	x79
04	# 3	33	23	--	x7A
05	$ 4	34	24	--	x7B
06	% 5	35	25	--	x7C
07	^ 6	36	5E	1E	x7D
08	& 7	37	26	--	x7E
09	* 8	38	2A	--	x7F
0A	(9	39	28	--	x80
0B	) 0	30	29	--	x81
0C	_ -	2D	5F	1F	x82
0D	+ =	3D	2B	--	x83
0E	BACK SPACE	08	08	7F	x8C
0F	TAB	09	x0F	x8D	x8E
10	Q	71	51	11	x10
11	W	77	57	17	x11
12	E	65	45	05	x12
13	R	72	52	12	x13
14	T	74	54	14	x14
15	Y	79	59	19	x15
16	U	75	55	15	x16
17	I	69	49	09	x17
18	O	6F	4F	0F	x18
19	P	70	50	10	x19
1A	{ [	5B	7B	1B	--
1B	}]	5D	7D	1D	--
1C	ENTER (main keyboard)	0D	0D	0A	X8F
1D	CTRL (control mode)	**	**	**	**
1E	A	61	41	01	x1E
1F	S	73	53	13	x1F
20	D	64	44	04	x20
21	F	66	46	06	x21
22	G	67	47	07	x22
23	H	68	48	08	x23
24	J	6A	4A	0A	x24
25	K	6B	4B	0B	x25
26	L	6C	4C	0C	x26
27	: ;	3B	3A	--	--
28	" '	27	22	--	--
29	ⓓ	x48	x85	x90	x91
2A	SHIFT (left SHIFT)	**	**	**	**

2B	←	x4B	x87	x73	x92
2C	Z	7A	5A	1A	x2C
2D	X	78	58	18	x2D
2E	C	63	43	03	x2E
2F	V	76	56	16	x2F
30	B	62	42	02	x30
31	N	6E	4E	0E	x31
32	M	6D	4D	0D	x32
33	< ,	2C	3C	--	--
34	> .	2E	3E	--	--
35	? /	2F	3F	--	--
36	SHIFT (right SHIFT)	**	**	**	**
37	PRINT (print screen toggle)	10	**	x72	x46
38	ALT (alternate mode)	**	**	**	**
39	(space bar)	20	20	20	20
3A	CAPS (caps lock)	**	**	**	**
3B	F1	x3B	x54	x5E	x68
3C	F2	x3C	x55	x5F	x69
3D	F3	x3D	x56	x60	x6A
3E	F4	x3E	x57	x61	x6B
3F	F5	x3F	x58	x62	x6C
40	F6	x40	x59	x63	x6D
41	F7	x41	x5A	x64	x6E
42	F8	x42	x5B	x65	x6F
43	F9	x43	x5C	x66	x70
44	F10	x44	x5D	x67	x71
45	NUM LOCK (number lock)	**	**	**	**
46	HOLD (freeze display)	**	**	**	**
47	\ 7	37	5C	x93	** †
48	↑ 8	38	7E	x94	** †
49	PG UP 9	39	x49	x84	** †
4A	↓	x50	x86	x96	x97
4B	← 4	34	7C	x95	** †
4C	5	35	--	--	** †
4D	6	36	--	--	** †
4E	→ (smooth scroll toggle)	x4D	x88	x74	**
4F	END 1	31	x4F	x75	** †
50	` 2	32	60	x9A	** †
51	PG DN 3	33	x51	x76	** †
52	0	30	x9B	x9C	** †

† To generate the ASCII code of a decimal number, hold down the (ALT) key while you type *on the numeric keypad* any decimal number between 1 and 255. When you release (ALT), the ASCII code of the number typed is generated and displayed.
Note: When the NUM LOCK light is off, the Normal and SHIFT columns for these keys should be reversed.

53	DELETE	x53	x8A	x9D	x9E
54	BREAK (BREAK routine - INT 1BH)	x00	x00	**	x00
55	INSERT	x52	x89	x9F	xA0
56	(numeric keypad)	2E	xA1	xA4	xA5
57	ENTER (numeric keypad)	0D	0D	0A	x8F
58	HOME	x47	x4A	x77	xA6
59	F11	x98	xA2	xAC	xB6
5A	F12	x99	xA3	xAD	xB7