HEWLETT-PACKARD

HP-85

Pocket Guide

HP-85 Pocket Guide

Contents

The HP-85 BASIC language consists of **statements**, **functions**, **operators**, and **commands**. Operators and functions are used with variables, numbers, and strings to create numeric and string **expressions**. Expressions and functions can be included in statements and executed from the keyboard. Each statement can be preceded by a statement number and stored as a program statement. Most functions, statements, and commands can be executed separately from the keyboard or used in programs; exceptions are noted.

OPERATORS

Arithmetic

+	Add
−	Subtract
*	Multiply
/	Divide
^	Exponentiate
MOD	Modulo: A MOD B=A−B*INT(A/B)
\ or DIV	Integer divide: A DIV B=IP(A/B)

Logical Evaluation

Logical expressions return the values 0 for false and 1 for true. Non-zero values are considered true; zero values are considered false.

Relational

=	Equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
<> or #	Not equal to

Non-numeric values can also be compared with relational operators. Strings are compared using decimal values, character by character, from left to right until a difference is found. If one string ends before a difference is found, the shorter string is considered the lesser.

Logical

AND
OR
EXOR
NOT

Truth Table

A	B	A AND B	A OR B	A EXOR B	NOT A	NOT B
T	T	1	1	0	0	0
T	F	0	1	1	0	1
F	T	0	1	1	1	0
F	F	0	0	0	1	1

1=True 0=False

String

& String concatenator

Math Hierarchy

( )
Functions
^
NOT
*, /, MOD, \ or DIV
+, –
Relational operators (=, >, <, >=, <=, <> or #)
AND
OR, EXOR

Performed First

Performed Last

Expressions are evaluated from left to right for operators at the same level. Operations within parentheses are performed first. Nested parentheses are evaluated inward out.

DATA PRECISION

Precision	Accuracy	Range	Maximum array size (standard memory, no program)
REAL	12 Digits	±9.99999999999E±499	1800
SHORT	5 Digits	±9.9999E±99	3600
INTEGER	5 Digits	-99999 through 99999	4800

SPECIAL CHARACTERS

@ Enables multi-statement lines.
```
100 CLEAR @ KEY LABEL
```

! Remark follows.
```
110 DISP C ! Display cost.
```

? INPUT prompt. Input items are expected.

" " String delimiters. Mark beginning and end of literal text.
```
120 PRINT "MEAN", "MODE"
```

VARIABLES

Simple numeric variables: A1, B, C3.

The name consists of a letter or a letter and one digit. Real precision is assumed unless SHORT or INTEGER type is declared.

Arrays: A1(50,5), B(20,20), C3(10).

The name consists of a letter or a letter and one digit. An array name can be the same as a simple variable name used elsewhere in the program, but a one-dimensional array cannot have the same name as a two-dimensional array. Arrays contain numeric elements only. Subscripts dimension the row or row and column in DIM, COM, or type (REAL, INTEGER, SHORT) declaration statements. The lower bound of an array subscript is 0 unless OPTION BASE 1 is specified before all array references. The default upper bound for row and column subscripts is 10.

Subscripts reference a particular array element in non-declaratory statements with three exceptions. Entire arrays (either one- or two-dimensional) may be referenced in TRACE VAR, PRINT#, and READ# statements by specifying the array name followed by a pair of parentheses and no subscripts (e.g., C3( )). A comma may be enclosed within the parentheses for documentation purposes to specify a two-dimensional array (e.g., A1(,)). This notation enables you to trace, write onto tape, or read from tape all elements of the specified array.

String variables: A1$, B$, C3$.

The name consists of a letter or a letter and one digit followed by a dollar sign. The default length is 18 characters unless otherwise specified in a COM or DIM statement. The maximum length of a string is limited only by available memory. Dimension strings in a DIM or COM statement by specifying the variable name followed by the length enclosed within brackets: A1$[25], B$[415], C3$[5].

Substrings: A1$[2,25], B$[5], C3$[3,3].

Substrings are specified by one or two numbers (or expressions) enclosed within *brackets*. One number specifies a beginning character; the substring extends to the end of the string. Two numbers separated by a comma specify beginning and ending character positions, respectively.

Strings can be compared with the relational operators and can be concatenated with the & operator. (Refer to Operators.)

SYNTAX GUIDELINES TO COMMANDS AND BASIC STATEMENTS

These terms and conventions are used in the following list of statements and commands.

color	All items in color denote system commands or BASIC statements that must appear exactly as shown. Either small or capital letters may be used to spell keywords.
[ ]	All items enclosed within square brackets are optional unless the brackets are in color.
...	Three dots indicate that the previous item can be repeated.
statement number	An integer from 1 through 9999.
numeric expression	A logical combination of variables, constants, operators, and functions (including user-defined functions), grouped within parentheses as necessary.

string expression — A logical combination of text within quotes, string variables, substrings, string concatenations, and string functions.

file name or **program name** — A program or file name can be any string expression. Any letter, number, symbol, or character except quotes or the null string may be used. Only the first six letters of the string expression are used for the name.

```
LOAD "PROG1*"
LOAD T$
```

buffer number — The number assigned to a tape data file by an ASSIGN# statement, and referenced by the PRINT# and READ# statements. Its range is 1 through 10.

COMMANDS

Non-Programmable

AUTO [**beginning statement number** [, **increment value**]]

Enables program statements to be numbered automatically by the system. If no parameters are specified, numbering begins with 10 and is incremented by 10. If one parameter is specified, numbering begins with that statement and is incremented by 10. Auto numbering can be halted by executing NORMAL or any executable statement or command without a statement number.

```
AUTO
AUTO 5
AUTO 100, 5
```

CONT [**statement number**]

Continues execution of the program at the specified statement, or where it was halted with PAUSE, without altering program conditions or modes.

```
CONT
CONT 640
```

DELETE **first statement number** [, **last statement number**]

Deletes an individual statement or the inclusive range of statements.

```
DELETE 20
DELETE 110, 150
```

INIT Allocates all variables and initializes them to undefined values. Resets the program pointer to the first line of the program.

LOAD program name

Loads into memory a copy of the specified program. The program must have been previously stored on tape with a STORE command. Computer memory is scratched before the program is loaded.

```
LOAD "Ac#3"
LOAD D$
```

REN [first statement number [, increment value]]

When parameters are not specified, renumbers all statements in the current program beginning with 10, in intervals of 10. The selected beginning number and interval may be specified. If renumbering will cause statement numbers to be larger than 9999, REN automatically renumbers the entire program beginning with 1, and incrementing by 1.

```
REN
REN 500
REN 2010, 2
```

RUN [statement number]

Starts execution of the current program at the first statement, or at the specified statement.

```
RUN
RUN 100
```

SCRATCH Deletes the current program and all variables from computer memory.

STORE program name

Stores the current program in memory into a program file on a tape cartridge.

```
STORE "TRADE"
STORE J5$
```

UNSECURE file name, security code, secure type number

Removes the security from a file secured with the SECURE command. The file name, first two characters of the security code, and secure type number must match those specified in the SECURE command.

Number	Secure type
0	LIST, PLIST, and edit.
1	STORE (duplication), LIST, PLIST, and edit.
2	STORE (overwriting), PRINT#, STORE BIN.
3	CAT (blank name in directory).

```
UNSECURE "DECANT", "J5", 1
UNSECURE "DATA/Q", "**", 3
```

Programmable

The following system commands may also be used in a program as BASIC statements.

CAT Produces a catalog listing of information about tape files.

COPY Produces a printed copy of the information on the current display.

CTAPE Conditions the tape by rewinding in both directions.

ERASETAPE Erases and initializes the current tape cartridge.

FLIP Toggles back and forth between BASIC mode (capital letters unshifted, small letters shifted) and "typewriter" mode (small letters unshifted, capital letters shifted).

LIST [beginning statement number [, ending statement number]]
Lists, on the current display, all or part of the current program from the lowest numbered statement to the highest numbered statement. If one statement number is specified, listing begins with that statement and continues for one screen. If two statement numbers are specified, that section of statements is listed. Following the list of the last program line, displays the remaining number of memory locations.

```
LIST
LIST 200
LIST 1050, 1200
```

PLIST [beginning statement number [, ending statement number]]
Same as **LIST** except that the listing appears on the current printer and all of the program will be listed at once, unless interrupted by any key.

PRINT ALL Produces a printed copy of all user activity. Any information appearing on the display is printed on the printer.

REWIND Rewinds the tape cartridge.

SECURE file name, security code, secure type number
Secures a program file against being listed, edited, stored, rewritten, or prevents the name of a program or data file from being listed in the directory. The file name must exist in the tape directory before it can be secured. The first two characters of the second parameter are used as the security code. Secure types 0 and 1 may be used with program files only. Secure types 2 and 3 may be used with program or data files.

Number	Secure type
0	LIST, PLIST, and edit.
1	STORE (duplication), LIST, PLIST, and edit.
2	STORE (overwriting), PRINT#, STORE BIN.
3	CAT (blank name in directory).

```
SECURE "DECANT", "J5", 1
SECURE "DATA/Q", "**", 3
```

BASIC STATEMENTS

Most BASIC statements can also be executed as commands without statement numbers. The exceptions are COM, DATA, DEF FN, DIM, DISP USING statement number, FN END, FOR, GOSUB, GOTO, IMAGE, INPUT, NEXT, ON, ON ERROR, ON KEY#, ON TIMER#, PRINT USING statement number, READ, and RETURN. You *can* execute a FOR-NEXT loop without statement numbers if the entire loop is concatenated with @ symbols.

ASSIGN# buffer number TO file name

Opens a data file by assigning a buffer number (1 through 10) to the file name.

```
100 ASSIGN#1 TO "DATA"
110 ASSIGN#7 TO "ZIP F8"
120 ASSIGN#3 TO R$
```

ASSIGN# buffer number TO *

An asterisk in place of the file name closes the specified file by writing the remaining contents of the buffer into the file.

```
130 ASSIGN#1 TO *
```

BEEP [tone, duration]

Produces an audible sound. If no parameters are specified, the frequency is approximately 2000 hertz and the duration is 100 milliseconds. By specifying parameters, the user can change the period and duration. Both parameters are integer numbers in the range of 1 through 99999. Tones produced with a first parameter less than 1000 are more distinct than tones produced with a first parameter greater than 1000.

```
140 BEEP
150 BEEP 40, 150
```

CHAIN file name

Loads and runs the designated program. The current program is deleted from memory except for variables declared in "common" (see COM) and buffers opened by the ASSIGN# statement. Chaining allows programs of unlimited size to be run by breaking up the program into smaller segments.

```
160 CHAIN "KITE 2"
170 CHAIN A3$
```

CLEAR Clears the alphanumeric screen.

```
180 CLEAR
```

COM common variable list

Declares variables in common. Primarily used with the CHAIN statement to pass variables between programs. The common list is one or more variables, array declarations, or string declarations separated by commas. The variables in common must agree in type and size between programs that are CHAINed.

```
190 COM X,Y, A$, INTEGER M(10, 16)
200 COM R2$,N$[40], T(3, 6), SHORT G(20, 2)
```

CRT IS output code number

Specifies the device that will display normal system messages.

Code	Device
1	CRT (display)
2	Printer

```
210 CRT IS 2
```

CREATE file name, number of records [, number of bytes per record]

Establishes a data file of the specified size. Assigns 256 bytes per record if not otherwise specified.

```
220 CREATE "DATA*", 20
230 CREATE "#NAMES", 60, 75
```

DATA data list

Provides numeric constants and quoted or unquoted strings from which the READ statement can obtain values for numeric and string variables. Data is read from left to right. DATA statements may be located anywhere in the program and are considered a continuous list. DATA cannot be executed as a command from the keyboard.

```
240 DATA 3, 76.5, "HENRY", "T204ABC"
```

DEFAULT OFF Cancels the use of default values for math errors and sets the computer to normal error processing (see DEFAULT ON).

DEFAULT ON Prevents the following math errors from halting program execution by providing default values for out-of-range results that occur in computations or assignments. A warning message will be displayed when a default value is used and the program will continue. The system powers on in DEFAULT ON mode. The errors and default values are:

Error Number	Default Value
Underflow (1)	0
Integer precision overflow (2)	+ or −99999
Short precision overflow (2)	+ or −9.9999E99
Real precision overflow (2)	+ or −9.99999999999E499
COT or CSC of $n*180°$; n = integer (3)	9.99999999999E499
SEC or TAN of $n*90°$; n = odd integer (4)	9.99999999999E499
Zero ∧ negative power (5)	9.99999999999E499
Zero ∧ zero (6)	1
Null data: uninitialized numeric variable (7)	0
Null data: uninitialized string variable (7)	""
Division by zero (8)	+ or −9.99999999999E499

DEF FN numeric variable name [(parameter)] [=numeric expression]

DEF FN string variable name [(parameter)] [=string expression]

Defines special functions within a program. The function name must be a simple variable; array names are not allowed. The parameter is a dummy variable used only in the definition; it is replaced by the actual variable when called. DEF FN defines a single line function or, with FN END, defines a multiple-line function. In multiple-line functions, the returned value is set with an assignment statement.

String parameters in user-defined functions are defaulted to local strings of less than or equal to 18 characters in length.

```
250 DEF FNA(X) = SQR(X^2+3 *X+ 1)
260 DEF FNC$(A$[100]) = C$ & A$ & B$
270 DEF FNT(Z)
280 W = ABS(4 *Z-82)
290 X = (W - ABS(Z))
300 Y = (W-X)^2
310 FNT = ABS(W+X- Y)
320 FN END
```

DEG Sets degrees mode for results and arguments of trigonometric functions. There are 360 degrees in a circle.

DIM dimension list

Declares the maximum subscript values for one- or two-dimensional REAL precision arrays and the maximum length for strings. A string variable name must be followed by the length in brackets; an array name must be followed by the maximum subscript values in parentheses.

```
330 DIM L(2, 3), M(10, 20), N$[100], O(5)
```

DISP [display list]

Causes items in the display list to be displayed on the CRT (or CRT IS device). Except for the output device, same as the PRINT statement. Items in the display list must be separated by commas or semicolons. A comma advances an item to the next display zone; a semicolon suppresses the advance.

```
340 DISP "X EQUALS"; X, "A EQUALS"; A
350 DISP "***WAIT A MINUTE***"
360 DISP SIN(X)^3+5, G$
```

DISP USING image format string [; disp using list]

DISP USING statement number [; disp using list]

The image format string is a string expression that determines the exact form of displayed output. The disp using list can contain variables and expressions separated by commas or semicolons. Each item in the disp using list must correspond to an appropriate field specifier in the image format string. The statement number must refer to an IMAGE statement. (Refer to IMAGE.)

```
370 DISP USING 50; "NAME", A$, I(J)
380 DISP USING S1$; C, B, M, J
390 DISP USING "3A, 2X, 4D.DD"; T$, K
```

END Terminates program execution.

FN END Last line of a multiple line function definition.

FOR loop counter = initial value TO final value [STEP increment value]

The FOR statement is used with NEXT and defines the number of times a FOR-NEXT loop is to be executed. The loop counter must be a simple numeric variable. Starting with a specified initial value, the loop moves to the final value by the optionally specified step. The default step is 1. The loop counter is set and tested before the loop is entered. It is incremented and then checked at the end of the loop.

```
400 FOR A = 1 TO 5
410 FOR B = -1 TO -3 STEP -.3*A
420 PRINT B
430 NEXT B
440 NEXT A
```

GOSUB statement number

Transfers program control to the subroutine beginning at the specified statement number. RETURN in a subroutine returns control to the statement following GOSUB.

```
450 GOSUB 760
```

GOTO statement number

Transfers program control to the specified statement number.

```
460 GOTO 310
```

GRAD Sets grads mode for all results and arguments of trigonometric functions. There are 400 grads in a circle.

IF numeric expression THEN statement number (or **executable statement**) **[ELSE statement number** (or **executable statement**)**]**

Provides conditional branching. If the numeric expression is evaluated as true, execution is transferred to the specified line or the statement is executed. If false and ELSE is not specified, execution continues with the next sequential statement. If false and ELSE is specified, execution is transferred to the specified ELSE line, or the specified ELSE statement is executed. Non-zero values are considered true, zero values are false.

The following statements cannot follow **THEN** or **ELSE**:

COM, DATA, DEF FN, DIM, FN END, FOR, IF, IMAGE, INTEGER, NEXT, OPTION BASE, REAL, SHORT.

```
470 IF X THEN 950
480 IF X+4=Y THEN PRINT "x="; X
490 IF B>78 THEN STOP ELSE DISP B
500 IF X> 9 OR Y>32 THEN 100 ELSE 500
```

IMAGE image format string

Used with the **PRINT USING** or **DISP USING** statements to specify output format: numeric and string field specifiers, blanks, and carriage control.

Field specifiers must be separated by a comma or slash (/).

The following is a list of symbols that are combined to make up field specifiers.

n(...)	parentheses allow field replication n times
A	string character
Z	digit position or leading zero
*	digit position or leading asterisk (*)
D	digit position or leading blank
.	decimal point (.)
S	sign (+ or −)
M	minus sign or blank
E	exponent in form ESDDD
K	use default format
X	blanks
R	comma radix (,)
C	comma (,)
P	period (.)
/	carriage return/line feed
""	literal

```
510 IMAGE 4D.DD,2X,**Z.DD,2X,ZZZRDD
520 IMAGE "COST=",5X,"$",2D.DD,2/,"DISCOUNT=",Z.DD,
"%"
530 IMAGE 6("HELLO")
532 PRINT USING 510; 1473, 25.39, 76.5
1473.00  *25.39  076,50
```

```
534 C=43.17 @ D=.07
536 PRINT USING 520, C,D*100
COST=        $43.17

DISCOUNT=7.00%

538 PRINT USING 530
HELLOHELLOHELLOHELLOHELLOHELLO
```

INPUT variable name$_1$ [, variable name$_2$ ...]

Allows variable assignments to be entered from the keyboard during program execution. INPUT can only be executed in a running program. Any numeric expression is valid for numeric input; any string expression, quoted or unquoted, is valid for string input (as long as an unquoted string contains no commas). Null is allowed with string input but not with numeric input. Separate input values with commas.

```
540 INPUT A, B
550 INPUT Z, H$
```

INTEGER numeric variable$_1$ [(subscripts)] [, numeric variable [(subscripts)] ...]

Dimensions and reserves storage space for INTEGER precision variables—simple and array.

```
560 INTEGER A(2,2), B
570 INTEGER C, D, E, F(2,23)
```

KEY LABEL Displays labels associated with the user-defined special function keys on the bottom three lines of the CRT display. The labels are created with the ON KEY# statement. Also returns the cursor to the home position.

```
580 KEY LABEL
```

[LET] numeric variable$_1$ [, numeric variable$_2$ ...] = numeric expression

[LET] string variable$_1$ [, string variable$_2$ ...] = string expression

Assigns a value to a variable or variables. The word LET is optional.

```
590 X = Y ∧ 2
600 LET A, B, C, D, E = 100
610 A$, B$ = C$[20]
620 X$ = "PRICES"
```

[LET] FN variable name = expression

Assigns the function a value. Necessary statement in multiple line function definition. (Refer to DEF FN.)

LOAD BIN file name

Loads the specified binary file into memory. At most one binary program is allowed in computer memory at any time.

```
630 LOAD BIN "ROUTIN"
```

NEXT loop counter

Used with the FOR statement, defines the last statement of a FOR-NEXT loop and causes the loop counter to be incremented and tested.

NORMAL Cancels all tracing operations, returns the system to normal print mode from print all mode, and stops auto statement numbering.

OFF ERROR Cancels any ON ERROR condition currently active.

OFF KEY# key number

Deactivates a corresponding ON KEY# statement so that pressing the key no longer has any effect on program control.

```
640 OFF KEY#3
```

OFF TIMER# timer number

Deactivates the corresponding ON TIMER#. No further interrupts from that timer will occur until reactivated.

ON ERROR GOSUB statement number

ON ERROR GOTO statement number

Prevents recoverable program execution errors from halting execution by branching when an error occurs and suppressing the normal error reporting process. ON ERROR overrides DEFAULT ON. The RETURN from ON ERROR GOSUB returns to the statement after the one in which the error occurred.

```
650 ON ERROR GOTO 2000
660 ON ERROR GOSUB 2040
```

ON numeric expression GOSUB statement number list

ON numeric expression GOTO statement number list

Transfers program control to one of one or more subroutines or statements in the current program based on the rounded, integer value of the numeric expression. A value of 1 corresponds to the first statement number in the list, 2 to the second, etc. A subroutine returns program control to the statement following the ON... GOSUB statement that referenced it.

```
670 ON X GOSUB 800,850,900
680 ON A4+3 GOTO 100,200,300,400
```

ON KEY# key number [, key label] GOSUB statement number

ON KEY# key number [, key label] GOTO statement number

Allows any user-defined special function key to be used for running program control. When the user-defined key is pressed during a program and an ON KEY# statement has been declared for it, the specified branching occurs. With ON KEY# ... GOSUB, control returns to the statement following the one that was executing when the key was pressed. The optional key label is a quoted string of at most eight characters. (Refer to KEY LABEL.)

```
690 ON KEY#1, "HELP" GO TO 1550
700 ON KEY#3 GO TO 10
710 ON KEY#4, "FORTUNE" GOSUB 1080
720 ON KEY#8 GOSUB 1150
```

ON TIMER# timer number, milliseconds GOSUB statement number

ON TIMER# timer number, milliseconds GOTO statement number

Specifies interrupt intervals for three different timers. When an interrupt occurs, the specified branching occurs. With ON TIMER# ... GOSUB, control returns to the statement following the one that was executing when the interrupt occurred.

```
730 ON TIMER#1, 32000 GOTO 3010
740 ON TIMER#3, 40 GOSUB 5000
```

OPTION BASE 1 or 0

OPTION BASE 1 specifies that the lower bound of all arrays is 1 rather than the default bound 0. OPTION BASE 0 need only be declared for documentation purposes since 0 is the default lower bound. The OPTION BASE statement must be declared before any array reference. Only one OPTION BASE statement is allowed in a program.

PAUSE Halts execution of a program without affecting the program pointer. The program may be resumed with a CONT command.

PRINT [print list]

Prints a list of items on the system printer (or current PRINTER IS device). The print list may contain variables, arrays, expressions (including multiple line user-defined functions), or TAB. Items must be separated by commas or semicolons. A comma advances an item to the beginning of the next print zone. A semicolon suppresses the advance.

```
750 PRINT
760 PRINT A; TAB(10); B; TAB(24); C
```

```
770 PRINT "KEY IN A NUMBER";
780 PRINT A, B, C, D
```

PRINT# buffer number ; [print# list]

PRINT# buffer number, record number [; [print# list]]

Records values in the print# list into the specified file on tape. In serial access mode (no record number specified), recording starts at the beginning of the file or after the last data item accessed. In random access mode (with the record number specified), recording starts at the beginning of the record. If there is no print# list, repositions the data pointer to the beginning of the specified record. The print# list can include variables, constants, and literals, separated by commas.

```
790 PRINT# 1; A, B(), C(,), D$
800 PRINT# 2, 6; E8(,)
810 PRINT# 5, 11
```

PRINT USING image format string [; print using list]

PRINT USING statement number [; print using list]

The image format string is a string expression that determines the exact form of printed output. The print using list can contain variables and expressions separated by commas or semicolons. Each item must correspond to an appropriate field specifier in the image format string. The statement number must refer to an IMAGE statement.

```
820 PRINT USING 100; J, K, L, C$
830 PRINT USING A$ ; C$[10,10]
840 PRINT USING "K, X, K, 2X, DDD.DD,K"; C$, "EARNED"
,X, "DOLLARS"
```

PRINTER IS output code number

Redefines the device that is used for PRINT statements.

Code number	Device
1	CRT (display)
2	Printer

PURGE file name [, purge code number]

Erases the specified file from the tape cartridge and creates a NULL file, enabling you to store a new program or create a new data file there. If the purge code is 0, purges the specified file and all subsequent files; makes the tape available for new files without old file sizes (no NULL files are created). If the purge code is not 0, the statement is the same as PURGE file name.

```
850 PURGE "5/6/78"
860 PURGE "PLOT",0
```

RAD Sets radians mode for all results and arguments of trigonometric functions. There are 2π radians in a circle. **RAD** is the default mode at power on.

RANDOMIZE [**numeric expression**]

Regenerates the random number seed. With no parameter, the seed is generated using the system timer. Specifying a parameter enables you to repeat the pseudo-random number sequence. A parameter of zero generates a constant sequence of zeros.

```
870 RANDOMIZE
880 RANDOMIZE PI/10
```

READ variable name$_1$ [**, variable name$_2$ ...**]

Reads numeric or string items from **DATA** statements and assigns them to the specified variables.

```
890 READ A, B, C, D(1,2) , E$[5,10]
900 READ X, Y, Z(5,3), C$, D$[1,2]
```

READ# buffer number ; variable list

READ# buffer number, record number [**; variable list**]

Retrieves values for variables from the specified buffer. In serial access mode (no record number specified), reading starts at the beginning of the file or after the last data item accessed. In random access mode, with the record number specified, reading starts at the beginning of the record. **READ#** can be used to reposition the data pointer by omitting the variable list in random mode. The variables in the variable list must be separated by commas.

```
910 READ#3; A, B(,) , C(1,2), D$[5]
920 READ#4, 7; X, Y, Z(,) , J$
930 READ#2 , 5
```

REAL numeric variable [**(subscripts)**] [**, numeric variable** [**(subscripts)**] **...**]

Dimensions and reserves storage space for simple and array variables and declares them full (real) precision.

REM [**any combination of characters**]

Places remarks in a program listing to provide documentation and make the program easier to follow. An exclamation mark may be used in place of **REM**.

```
940 REM Output the data.
950 ! *****HISTOGRAMS *****
```

RENAME old file name TO new file name

Enables any file to be given a new name.

```
960 RENAME "5/6/78" TO "5/6/79"
970 RENAME "SKIP" TO "HOP"
```

RESTORE[statement number]

Resets the data pointer to the beginning of the specified DATA statement, or to the lowest numbered DATA statement in the current program if no statement is specified, so that values can be reused.

RETURN Last line of a subroutine. Returns control from a GOSUB branch to the statement immediately following the GOSUB statement that referenced it.

SET TIME seconds since midnight, Julian day in form yyddd

Sets time and date into the system timer. Default value for both parameters at power on is 0.

```
980 SETTIME 43200,79114
990 SETTIME 8*3600+32*60+20, 80137
```

SHORT numeric variable [(subscripts)] [, numeric variable [(subscripts)] ...]

Dimensions and reserves storage space for simple and array variables and declares them short precision.

```
1000 SHORT H, J(10,4), K(2,2), L
```

STOP Terminates program execution and sets the program pointer to the lowest numbered statement.

STORE BIN file name

Stores current binary program in memory into a binary program file on a tape cartridge.

```
1010 STORE BIN "ROUTIN"
```

TRACE Used to follow the order of statement execution in all or part of a program. Any branching causes a trace output to be printed, designating where the branching was from and which line it was going to. Cancel trace operations with NORMAL.

```
1020 TRACE
```

TRACE ALL Used primarily with the STEP key to trace all statements and variable assignments in a running program. The TRACE ALL output indicates the previously executed statement, current statement number, and any variable changes. Cancelled with NORMAL.

```
1030 TRACE ALL
```

TRACE VAR variable$_1$ [, variable$_2$ ...]

Monitors and reports value changes of selected variables in a running program. The trace output indicates the statement number in which the assignment occurred and the name and new value of simple variables; the name, row or row and column, and new value of array elements; the name only of string variables; and the name and new first value of entire arrays. Cancelled with NORMAL.

```
1040 TRACE VAR A, B(,), C$, D( )
1050 TRACE VAR X,Y,H,J,K
1060 TRACE VAR C( ), Z( ), J(,)
```

WAIT number of milliseconds.

Delays program execution by the approximate number of milliseconds specified before it continues. The range of the wait is 0 to 27 minutes (1,666,650 milliseconds).

```
1070 WAIT 250
1080 WAIT X
```

Graphics Statements

ALPHA Sets the display to alphanumeric mode.

BPLOT character string, number of characters per line

Plots a group of dots on the graphics display as specified by the character string.

DRAW *x*-coordinate, *y*-coordinate

Draws a line from the current pen position on the graphics screen to the *x*,*y* coordinate position specified.

GCLEAR [*y*]

Clears the graphics screen from the specified *y* value to the bottom of the screen, or the entire screen if no parameter is specified, in current pen color.

GRAPH Sets the display to graphics mode.

IDRAW *x*-increment, *y*-increment

Incremental draw. Draws a line from the current pen position to the position determined by incrementing the current pen coordinates by the specified increment values.

IMOVE ***x*-increment, *y*-increment**

Incremental move. Moves the pen from the current pen position to the position determined by incrementing the current pen coordinates by the specified increment values.

LABEL character string

Writes a character string on the graphics display at the current pen position.

LDIR numeric expression

Label direction. Specifies the direction for labels in graphics mode. Horizontal labels are specified by values less than 45. Vertical labels are specified by values greater than or equal to 45. Default label direction is horizontal.

MOVE ***x*-coordinate, *y*-coordinate**

Moves the pen to the specified coordinate position without drawing a line on the graphics display.

PEN numeric expression

Specifies whether the plotting is done with white dots or black dots. When the expression is positive, a white dot is specified; when it is negative, a black dot is specified.

PENUP Lifts the pen; inhibits line generation.

PLOT ***x*-coordinate, *y*-coordinate**

Moves the pen from the current point to the specified location, drops the pen, and makes a dot. If the pen was down, draws a line from current point to specified point.

SCALE ***x* min, *x* max, *y* min, *y* max**

Scales the graphics display in user-defined units. Default values are 0, 100, 0, 100.

XAXIS ***y*-intercept [, tic spacing [, *x* min, *x* max]]**

Draws a horizontal axis on the graphics display. Tic marks and initial and final x values can be specified. Positive tic parameters specify the left side of the screen as a reference, negative tics specify the right side as a reference.

YAXIS ***x*-intercept [, tic spacing [, *y* min, *y* max]]**

Draws a vertical axis on the graphics display. Tic marks and initial and final y values can be specified. Positive tic parameters specify the bottom of the screen as a reference, negative tics specify the top of the screen as a reference.

Predefined Functions

ABS(X)	Absolute value of X.
ACS(X)	Arccosine of X, in 1st or 2nd quadrant.
ASN(X)	Arcsine of X, in 1st or 4th quadrant.
ATN(X)	Arctangent of X, in 1st or 4th quadrant.
ATN2(Y,X)	Arctangent of Y/X, in proper quadrant.
CEIL(X)	Smallest integer >=X.
CHR$(X)	Character whose decimal character code is X, 0<=X<=255.
COS(X)	Cosine of X.
COT(X)	Cotangent of X.
CSC(X)	Cosecant of X.
DATE	Julian date in format yyddd (assumes system timer has been set properly).
DTR(X)	Degree to radian conversion.
EPS	Smallest machine number (1E-499).
ERRL	Line number of latest error.
ERRN	Number of latest error.
EXP(X)	e^x
FLOOR(X)	Same as INT(X) (relates to CEIL).
FP(X)	Fractional part of X.
INF	Largest machine number (9.99999999999E499).
INT(X)	Largest integer <=X.
IP(X)	Integer part of X.
LEN(S$)	Length of string S$.

LGT(X)	Log to the base 10 of X, X>0.
LOG(X)	Natural logarithm, X>0.
MAX(X,Y)	If X>Y then X, else Y.
MIN(X,Y)	If X<Y then X, else Y.
NUM(S$)	Decimal character code of first character of S$.
PI	3.14159265359
POS(S1$, S2$)	Searches string S1$ for the first occurrence of string S2$. Returns starting index if found, otherwise returns 0.
RMD(X,Y)	Remainder of X/Y: X − Y * IP(X/Y)
RND	Next number, X, in a sequence of pseudo-random numbers, 0<=X<1.
RTD(X)	Radian to degree conversion.
SEC(X)	Secant of X.
SGN(X)	The sign of X: −1 if X<0, 0 if X=0, and +1 if X>0.
SIN(X)	Sine of X.
SQR(X)	Positive square root of X.
TAB(N)	Skips to specified column.
TAN(X)	Tangent of X.
TIME	Time in seconds since midnight (assumes system timer has been set properly) or since power on.
UPC$(S$)	Converts all lower-case alphabetic characters in S$ to upper-case.
VAL(S$)	Returns the numeric equivalent of the string S$.
VAL$(X)	String equivalent of X.

HP-85 CHARACTER CODES

Character	Control Key	Decimal Code	Character	Decimal Code
◀	@ᶜ	0	space	32
¿	Aᶜ	1	!	33
X̄	Bᶜ	2	"	34
N̄	Cᶜ	3	#	35
α	Dᶜ	4	$	36
β	Eᶜ	5	%	37
Γ	Fᶜ	6	&	38
n̄	Gᶜ	7	'	39
Δ	Hᶜ	8	(	40
σ	Iᶜ	9	)	41
↑	Jᶜ	10	*	42
λ	Kᶜ	11	+	43
μ	Lᶜ	12	,	44
	Mᶜ	13	–	45
τ	Nᶜ	14	.	46
Φ	Oᶜ	15	/	47
Θ	Pᶜ	16	0	48
Ω	Qᶜ	17	1	49
δ	Rᶜ	18	2	50
Ä	Sᶜ	19	3	51
ä	Tᶜ	20	4	52
Ä	Uᶜ	21	5	53
ä	Vᶜ	22	6	54
Ö	Wᶜ	23	7	55
ö	Xᶜ	24	8	56
Ü	Yᶜ	25	9	57
ü	Zᶜ	26	:	58
Æ	[ᶜ	27	;	59
œ	\ᶜ	28	<	60
²	]ᶜ	29	=	61
£	^ᶜ	30	>	62
■	_ᶜ	31	?	63

HP-85 CHARACTER CODES

Character	Decimal Code	Character	Shift Key	Decimal Code
@	64	`	(KEY LABEL)S	96
A	65	a		97
B	66	b		98
C	67	c		99
D	68	d		100
E	69	e		101
F	70	f		102
G	71	g		103
H	72	h		104
I	73	i		105
J	74	j		106
K	75	k		107
L	76	l		108
M	77	m		109
N	78	n		110
O	79	o		111
P	80	p		112
Q	81	q		113
R	82	r		114
S	83	s		115
T	84	t		116
U	85	u		117
V	86	v		118
W	87	w		119
X	88	x		120
Y	89	y		121
Z	90	z		122
[	91	π	/S	123
\	92	|		124
]	93	→	-S	125
^	94	Σ	•S	126
_	95	⊢	+S	127

ERROR MESSAGES

No.	Error Condition
	Default Math Errors
1	UNDERFLOW
2	OVERFLOW
3	COT/CSC=INF
4	TAN/SEC=INF
5	0∧NEG
6	0∧0
7	NULL DATA
8	/ZERO
	Non-Default Math Errors
9	NEG∧NON-INT
10	SQR(−)
11	ARG OUT OF RANGE
12	LOG(0)
13	LOG(−)
14	Not used.
	System Errors
15	SYSTEM
16	CONTINUE BEFORE RUN
17	FOR NESTING
18	GOSUB NESTING
19	MEM OVF
20	Not used.
21	ROM MISSING
22	SECURED
23	SELF TEST
24	> 14 ROMS
25	TWO BIN PROGS
26-29	Not used.

No.	Error Condition
	Program Errors
30	OPTION BASE
31	CHAIN
32	COM MISMATCH
33	DATA TYPE
34	NO DATA
35	DIM EXIST VRBL
36	DIM ILLEGAL
37	DUP FN
38	NO FN END
39	FN MISSING
40	FN PARAM
41	FN=
42	RECURSIVE FN CALL
43	NUMERIC INPUT
44	TOO FEW INPUTS
45	TOO MANY INPUTS
46	NEXT MISSING
47	NO MATCHING FOR
48	END
49	NULL DATA
50	BIN PROG MISG
51	RETURN W/O GOSUB
52	IMAGE
53	PRINT USING
54	TAB
55	SUBSCRIPT
56	STRING OVF
57	MISSING LINE
58-59	Not used.

No.	Error Condition
	Tape Errors
60	WRITE PROTECT
61	>42 FILES
62	CARTRIDGE OUT
63	DUP NAME
64	EMPTY FILE
65	END OF TAPE
66	FILE CLOSED
67	FILE NAME
68	FILE TYPE
69	RANDOM OVF
70	READ
71	EOF
72	RECORD
73	SEARCH
74	STALL
75	NOT HP-85 FILE
76-79	**Not used.**
	Syntax Errors
80	) EXPECTED
81	BAD EXPRESSION
82	STRING EXPR
83	"," MISSING
84	EXCESS CHARS
85	EXPR TOO BIG
86	ILLEGAL AFTER THEN
87	BAD DIM
88	BAD STMT
89	INVALID PARAM
90	LINE >9999
91	MISSING PARAM
92	SYNTAX

00085-90040 Rev. C 7/80

Printed in U.S.A.