Syntax Reference

HP250

Syntax Reference

Manual Part no. 45260-90063

Hewlett-Packard Company
8010 Foothills Blvd., Roseville, CA 95678

Printing History

First Edition .... November 1981
Second Edition ... January 1983

Table of Contents

Syntax Legend

The HP 250 BASIC language consists of **statements**, **functions**, **operators** and **commands**. Operators and functions are used with variables and numbers in creating numeric and string **expressions**. Expressions can be included in statements and executed from the keyboard. Each statement can also be preceded by a line number and stored as a program line. Commands can only be executed from the keyboard; they are not programmable.

Conventions Used In This Guide

`dot matrix` – all key words and characters in dot matrix must appear as shown.

... – an ellipsis indicates that the previous element can be duplicated.

[ ] – all elements enclosed in brackets are optional unless the brackets are in dot matrix. Several elements stacked inside a pair of brackets means the user may select any one or none of these elements. For example: $\begin{bmatrix}A\\B\end{bmatrix}$ user may select A or B or neither.

{ } – when several elements are stacked within braces, the user **must** select one of these elements. For example: $\begin{Bmatrix}A\\B\\C\end{Bmatrix}$ the user must select A or B or C.

name – a capital letter followed by up to 14 lowercase letters, digits or underscores. For example –

- variables (simple, array, numeric and string)
- labels
- subprogram names

line number – an integer from 1 thru 9999.

line label – a unique name (see **name** above) given to a program line. It follows the line number and is followed by a colon.

line id – a program line can be identified either by its line number (`GOTO 150`) or its label, if any (`GOTO Routine`).

current line – the next program line to be executed: normally the first line in memory, unless the program was suspended by HALT or PAUSE.

numeric expression – a logical combination of variables, constants, operators and functions (including user-defined functions) grouped within parentheses as necessary.

string expression – a series of characters (text) within quotes, a string variable, a substring, a string concatenation operation (`&`), a string function (including user-defined), or any combination of these items.

constant – a fixed number within the system's range, such as 2.12.

character – a letter, number, symbol or any arbitrary 8-bit byte defined by the CHR$ function.

text – a string of characters, quoted or unquoted.

variable – a name which is assigned a value and specifies a location in memory.

array identifier – a numeric variable name followed by `(*)`, indicating the use of the entire array variable.

pass parameters – includes variables, array identifiers, expressions and data file numbers (preceded by `#`).

formal parameters – includes non-subscripted variables, array identifiers and data file numbers (preceded by `#`).

subscripts – numbers within parentheses which are attached to a variable name and reference a particular variable element or boundary.

redim subscripts – numeric expressions separated by commas and enclosed in parentheses to redefine array working bounds. (The number of dimensions cannot change and the total number of elements cannot increase over the number originally dimensioned.)

file number – the number assigned to a mass storage data file by an ASSIGN statement. Its range is 1 thru 10.

file name – a one to six character string with the exception of a space, quote mark, comma, colon, NULL or CHR$(255).

device address – an expression (rounded to an integer) in the range 0 thru 20 which indicates the peripheral device address. These device addresses are reserved –
0 – standard external printer
6 – flexible disc drives
7 – built-in 5 or 10 Mb Disc or HP 7910 Fixed Disc
8 – display (standard printer at power-on)
9 – null address (allows outputting data to a "bit bucket")
10 thru 20 – peripherals connected via data comm interface.

unit spec – a string expression specifying the mass storage device. The form is –
device type [select code [device address [unit code]]]

The letters specifying the various device types are –
C – HP 7906A Removeable Disc
D – HP 7906A Fixed Disc
F – Flexible Disc (default address 6)
G – 5 Mb Disc
H – 10 Mb Disc
K – Cartridge Tape Drive
L – HP 7910 Fixed Disc (default address 7)
Q – HP 7908 Disc Drive
R – HP 7911P Disc
S – HP 7912P Disc

The device address range is from 0 thru 7: the unit code is from 0 thru 7 (codes 0 thru 4 are for flexible disc units). For example, the unit spec :F2,6,0 specifies the top flexible disc drive. The select code always defaults to 2.

volume label – a one- to eight-character string assigned to the storage medium via PRINT LABEL.

volume spec – a string specifying either a unit spec or a volume label preceded by a comma. For example,

,payrolls

file spec – a string expression of the form –

file name [volume spec]

The optional volume spec is needed when addressing a mass storage device other than the default device (see MASS STORAGE IS).

protect code – any valid string expression (except a null string) assigned via the PROTECT statement. Only the first six characters are recognized as the protect code.

SFK – special function keys defined via ON KEY# and EDIT KEY#.

standard printer – the output device selected by the PRINTER IS statement.

system printer – the output device selected by the SYSTEM PRINTER IS statement.

default device – the mass storage device specified in the configuration file as the default. MASS STORAGE IS is used to change the default device.

Operators

Arithmetic Operators

		Examples
+	add	10 + 5 = 15
-	subtract, negate	10 - 5 = 5 -2
*	multiply	10 * 5 = 50
/	floating point divide	15 / 10 = 1.5
^ or **	exponentiate	8^3 = 512
DIV	integer divide	15 DIV 10 = 1 -15 DIV 10 = -1
MOD	modulo; A MOD B = A−(B * INT(A / B))	20 MOD 6 = 2 -13 MOD 2 = 1 -13 MOD -2 = −1

Relational Operators

		Examples
<	less than	A < B
>	greater than	A > B
<=	less than or equal to	A <= B
>=	greater than or equal to	A >= B
=	equals	A = B
<> or #	not equal to	A <> B

Logical Operators

AND	logical AND	A AND B
OR	inclusive OR	A OR B
EXOR	exclusive OR	A EXOR B
NOT	logical NOT	NOT A

Truth Table

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

Operational Hierarchy

( ) (parenthesis) (highest)
^ (exponentiation)
NOT, unary +, −
*, /, MOD, DIV
+, −
=, <, >, <=, >=, < > or # (relational)
AND
OR, EXOR (lowest)

Real : 10
Short : 6
Integer : 4
String : 6 + length

A

ABS (numeric expression)

This function returns the absolute value of the numeric expression.

ASSIGN { file spec TO# file number | #file number TO file spec } [, return variable [, protect code] [; class list]]

ASSIGN { * TO# file number | #file number TO* }

Opens a data file by assigning it a number (first syntax). It also closes a data file by using * in place of the file spec (second syntax). The return variable can be any simple numeric variable. The protect code is required when the file is protected. The class list is a series of keywords separated by commas which specify the type of file access.

Value of Return Variable	Meaning
0	File available.
1	No such file found.
2	File is protected.
3	Wrong file type.
4	Access violation.
5	Other error.

Class	Word	Meaning
1	EXCLUSIVE UPDATE READ ONLY	Type of access to assign one keyword only
2	BUFFERED	Assign buffer
3	CHECKREAD	Specify automatic checkread

AUTO [beginning line number [, increment value]]

This command allows program lines to be numbered automatically as lines are stored. If no parameters are specified, numbering begins with the last line number in memory plus 10, and is incremented by 10.

AVAIL

This function returns the total number of available records onthe medium in the current default mass storage device.

B

BEEP

Outputs an audible beep.

BUFFER# file number

Assigns a 256-byte buffer from user read/write memory to the specified file to reduce mass storage device transfers.

C

CALL subprogram name [(pass parameter list)]

Transfers control to a subprogram. A pass-parameter list is not allowed in calls initiated from ON . . . branching statements.

CASE case item [, case item [, ...]]

$$\text{case item - }\left\{\begin{matrix}\text{constant}\\ \text{"string"}\end{matrix}\right\}\left[\text{"TO"}\quad\left\{\begin{matrix}\text{constant}\\ \text{"string"}\end{matrix}\right\}\right]$$

A number of CASE statements may be included in a structured SELECT block. If the SELECT expression is within the range of a case item, the following block of statements are executed. If not, execution resumes after the END SELECT or CASE ELSE statement.

CASE ELSE

This statement is optionally added to a structured SELECT block. Execution resumes here, rather than at the END SELECT statement when the SELECT expression does not match any CASE list.

CAT[ALOG][catalog spec][; file type]

The catalog spec can be followed by a volume spec.

Outputs the file name, type, and physical specifications for each user file on either the default mass storage device (omit volume spec) or the specified device. The optional catalog spec is a string expression of one thru six characters; when specified, only file names beginning with that combination of characters are output. The optional file type is a four-character string specifying to list only that type of files. The catalog is output to the system printer.

The available file types are –

DATA	–	Data file
PROG	–	Program file
KEYS	–	Special function keys file
OTHR	–	Not created by HP 250
BPRG	–	Binary program file (programs supplied by HP)
FORM	–	FORM software file
DROM	–	Dynamic Relocatable Option Module (software) file
COMM	—	Configuration file for INP
DSET ROOT	}	IMAGE software files
SYST	–	Main operating system file
BKUP	—	Backup file

CHECKREAD [OFF] [# file number]

Verifies that the information being stored on a storage medium is identical to that in memory. The optional file number specifies to checkread data to that file only.

CHECK READ also forces the output of data if it is not buffered (see BUFFER). Including OFF deactivates a previous corresponding CHECK READ statement, either general or for a specified file.

CHR$ (numeric expression)

The character function converts a numeric value between −32768 and 32767 into a string character. Any number outside the range of 0 thru 255 is converted modulus 256 to that range.

COL operand array

This function returns the number of columns in the array.

COM item [, item...]

Dimensions and reserves memory space for simple and array variables and data file numbers in a "common" memory area, allowing values or file status to be passed to subprograms or to other programs. Each item in the list can be –

simple numeric
numeric array (subscripts)
simple string [(number of characters)]
string array (subscripts) [(number of characters)]
# file number

COM cannot be executed from the keyboard.

COMMAND string expression

This statement executes the statement composed in the string expression. COMMAND cannot be executed from the keyboard.

$\left\{\begin{matrix}\text{CONT}\\\text{CONTINUE}\end{matrix}\right\}$ [line id]

This command resumes execution of a program, either at the specified line or where it was suspended (no line id), without altering program conditions and modes.

COPY **source file spec [** TO **destination file spec**
[; protect code]]

Copies a file from one medium to another. The protect code is needed only if the source file is protected. Omitting the destination file spec outputs the source file contents to the standard printer (the source file is assumed to be a spool file).

[F] CREATE **file spec ; number of defined records**
[; record length]

Establishes a data file of the specified size and places an EOF mark in each word of every record. The number of records and record length (in bytes) can range from 1 thru 65534. Using the FCREATE keyword speeds up the operation by omitting file initialization with EOFs.

CURSOR item list

Controls the display cursor and displayed character fields. The control items available are –

([X position][;Y position])	Set cursor position
IV field length	Set inverse video
BL field length	Blinking characters
UL field length	Underline field
HB field length	Set half bright display
RE field length	Reset
PL no. of lines	Protect lines
PALL	Protect entire page
UPL no. of lines	Unprotect lines
UPALL	Unprotect entire display buffer
IF field length	Specify input field
OF field length	Specify output field

RIF field length	Reset input field
ROF field length	Reset output field

Each parameter can be an integer expression greater than 0.

Other control items are allowed with the FORMS system software; refer to the FORMS/250 section for details.

CURKEY numeric variable

This function returns an integer number indicating the source of an ON...KEY# interrupt (without TIO/250 DROM) or an ON...condition interrupt (with TIO/250).

return value	meaning
0	No interrupt
1 thru 24	SFKs 1 thru 24
① ② ③	
25, 26, 27	Port 1 (device address 11)
28, 29, 30	Port 2 (device address 12)
31, 32, 33	Port 3 (device address 13)
34, 35, 36	Port 4 (device address 14)
37, 38, 39	Port 5 (device address 15)
55	ON DELAY

① Indicates an ON INPUT# or ON OUTPUT# interrupt.

② Indicates an ON BREAK# interrupt.

③ Indicates an ON CONNECT# or ON DISCONNECT# interrupt.

D

DATA constant or text [, constant or text...]

This statement provides constants and quoted or unquoted text from which READ obtains values for numeric and string variables. DATA cannot be executed from the keyboard.

DATE$

This function returns the current system date, if one has been set. The date is returned in either European or US format, depending upon the format in which it was set.

DEFAULT $\left\{\begin{matrix}\text{ON}\\\text{OFF}\end{matrix}\right\}$

Specifying ON prevents the following math errors from halting program execution by providing default values for out-of-range results which occur in computations or assignments. The default values allow a program to execute completely, using the default values, rather than stopping due to any of these math errors –

Error (Number)	Default Value
Integer precision overflow (20)	32767 or −32768
Short precision overflow (21)	+ or − 9.99999E63
Real precision overflow (22)	+ or − 9.99999999999E99
Intermediate result overflow (23)	+ or −9.99999999999E511
TAN(N*PI/2), N:odd integer (24)	9.99999999999E511
Zero to negative power (26)	9.99999999999E511
LGT or LOG of 0 (29)	−9.99999999999E511
Division by 0 (31)	+ or −9.99999999999E511
X MOD Y, Y=0	0

Specifying OFF cancels any previous DEFAULT ON.

DEF FN $\left\{\begin{matrix}\text{function name}\\\text{function name\$}\end{matrix}\right\}$ [(formal parameter list)] = expression

DEF FN $\left\{\begin{matrix}\text{function name}\\\text{function name\$}\end{matrix}\right\}$ [(formal parameter list)]

Defines a single-line function (first syntax), or a multiple-line function subprogram (second syntax). DEF FN cannot be executed from the keyboard.

DEL first line id [- second line id]

This statement deletes a line or section of a program. If only one line id is specified, just that line is deleted. If two line ids are specified, the entire block of lines is deleted.

DEL SUB subprogram name [TO END]

Deletes the named subprogram from memory. If TO END is specified, all successive subprograms and user-defined function subprograms are also deleted.

DEL FN function name [$][TO END]

Deletes the named user-defined function subprograms from memory. If TO END is specified, all successive subprograms and function subprograms are also deleted.

DET [operand matrix]

This function returns the determinant of either the specified matrix or the last matrix inverted using fetch MAT...INV.

DIM item [, item...]

Declares the number of dimensions and the maximum number of elements in each dimension for real-precision array variables and initializes all elements to 0. The DIM statement is also used to define the maximum length of all string variables, declare the number of dimensions and maximum number of elements in each dimension and initialize all strings to the null string. Each item in the list can be –

numeric array (subscripts)
simple string [number of characters]
string array (subscripts) [[number of characters]]

DIM cannot be executed from the keyboard.

DIRECT

Dumps all pending data from disc to cartridge tape. Causes all subsequent requests to come through the memory buffer, not the disc buffer.

DIRECT NOUPDATE

Starts memory buffered operation without dumping any buffers. This may cause loss of data. It is intended to be used when the disc is having problems which require diagnosis.

DISABLE

Deactivates any ON KEY# interrupt declarative so that pressing that key has no effect on current program control. The interrupt is still recorded.

DISP [display list]

Causes the items in the list to be displayed. The items can be variables, expressions, array identifiers, and output functions (SPA, TAB, LIN, and PAGE). Each item is separated by a comma or semicolon.

DISP USING $\begin{Bmatrix}\text{image format string}\\ \text{line id}\end{Bmatrix}$ [; print using list]

This statement is similar to PRINT USING, except that the output is always directed to the display. See PRINT USING and IMAGE.

DOOR LOCK volume spec

Locks the door of the specified flexible disc drive. DOOR LOCK cannot be executed from the keyboard.

DOOR UNLOCK [volume spec]

Unlocks the door of either the specified flexible disc drive (volume spec) or all locked devices.

DOT (vector$_1$, vector$_2$)

This function returns the inner (dot) product of two vectors.

DROUND (numeric expression , number of significant digits)

The digit round function returns the numeric expression rounded to the specified number of significant digits.

E

EDIT ["prompt" $\begin{Bmatrix} , \\ ; \end{Bmatrix}$] string variable

Displays the current value of the string variable (up to 160 characters in length) and waits for the operator to edit it. Pressing ⇧ stores the new string value and continues program execution. EDIT cannot be executed from the keyboard.

EDIT KEY # SFK number

This command sets a mode to define a special function key (SFK) as a series of keystrokes for use as a typing aid.

ELSE

This statement is optionally added to a structured IF ... THEN block. A false expression causes execution to resume at the line following ELSE, rather than after the END IF. A true expression causes execution to skip all lines between ELSE and END IF.

ENABLE

Reactivates any ON KEY# interrupt declaratives that were previously deactivated by DISABLE.

END

Terminates program execution and resets the program line pointer. END is not keyboard executable.

END IF

This statement terminates a structured IF ... THEN block.

END LOOP

This statement terminates a structured LOOP block.

END SELECT

This statement terminates a structured SELECT block.

END WHILE

This statement terminates a structured WHILE block.

ENTER variable $name_1$ [, variable $name_2$...]

Used to input data from the display and continue program execution. ENTER is not keyboard executable.

ERRL

The error line function returns the line number in which the most recent program execution error occurred.

ERRM$

The error message string function returns the most recent error message encountered in the program.

ERRN

The error number function returns the number of the most recent program error.

EXIT IF conditional expression.

This statement is placed within a structured LOOP block. Execution exits the block when the expression is true.

EXP (numeric expression)

The exponential function returns the value of Napierian (e≃2.71828182846) raised to the power of the computed expression.

F

FETCH [line id]

This command displays the specified program line. Omitting the line identifier recalls the current program line.

FIXED number of digits

Sets fixed point mode for output of numeric values and specifies from 0 thru 12 digits to the right of the decimal point.

FLOAT number of digits

Sets floating point mode (scientific notation) for output of numeric values and specifies from 0 thru 11 digits to the right of the decimal point.

FNEND

This statement is an optional last line in a function subprogram.

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

Defines how many times a FOR-NEXT loop is to be executed. The loop counter must be a simple variable. If no increment value is specified, it defaults to 1.

FRACT (numeric expression)

This function returns the fractional part of the evaluated expression and is defined by:

argument − INT (argument).

G

GET file spec [, first line id [, execution line id]]

Brings into memory a program saved with the SAVE statement, or any string data file consisting of valid BASIC statements. When the first line id is specified, the program is renumbered so that it begins with the line number specified. The second line id specifies where execution is to begin.

GOSUB line id

Transfers program control to the subroutine beginning at the specified line.

GOSUB numeric expression OF line id list.

See ON...GOSUB statement.

GOTO line id

Transfers program control to the specified line.

GOTO numeric expression OF line id list

See ON...GOTO statement.

H

HOLE

This function returns the largest number of available contiguous free records on the default mass storage device.

I

IF numeric expression THEN [line id
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. The following statements cannot follow THEN.

COM
DATA
DIM
DEF FN
SHORT
SUB
SUBEND

INTEGER
OPTION BASE
REAL
REM
FN END
END
IMAGE

When no parameters follow THEN, a true expression causes execution to resume with the next line; a false expression causes execution to resume with the line following either an ELSE statement or (if ELSE is omitted) an END IF.

IMAGE image format string

Used with PRINT USING or DISP USING to specify the output format using numeric and string field specifiers, blanks, and carriage control. Field specifiers must be separated by a comma, @, or a slash. Here's a list of symbols which are combined to make up field specifiers —

D Specifies a digit position. The fill character is a blank. nD specifies n digit positions.

Z Specifies a digit position. The fill character is a zero. nZ specifies n digit positions.

* Specifies a digit position. The fill character is an asterisk. n* specifies n digit positions.

X Causes a blank to be printed. nX causes n blanks to be printed.

A Specifies a single string character position. nA specifies n string characters.

. Indicates placement of a decimal point radix indicator. There may be only one radix indicator per numeric specifier.

R Indicates placement of a comma radix indicator. There may be only one radix indicator per numeric specifier.

C Indicates placement of a comma in a numeric specification. It is a conditional character and is output only if there is a digit to its left.

P Indicates placement of a period in a numeric specification. It is a conditional character and is output only if there is a digit to its left.

S Indicates a sign position for a + or −. The sign floats to the left of the leftmost significant digit if S appears before all digit symbols.

M Indicates a sign position; + is replaced by a blank. The sign floats to the left of the leftmost significant digit if M appears before all digit symbols.

E Causes output of an E, a sign and a two-digit exponent for output of numbers in scientific notation.

K Specifies an entire string or numeric field. A numeric specifier is output in standard format, except that no leading or trailing blanks are output. The current value of a string is output.

One of these three control characters can be placed at the beginning of the image string to override normal carriage-return line-feed (CRLF) output with PRINT USING or DISP USING —

+ Suppresses line feed (LF).

- Suppresses carriage return (CR).

# Suppresses CRLF.

@ Outputs a form feed.

/ Causes a CRLF to be output. n / causes n CRLFs to be output.

( ) Parentheses allow specifiers to be replicated.

" " Specifies text.

@ and / can also be used as delimiters to separate field specs.

IMAGE cannot be executed from the keyboard.

INDENT starting column , increment

This command re-positions the starting column of all program lines. Structured constructs (IF ... THEN-ELSE, WHILE, LOOP, SELECT, FOR-NEXT, etc.) are further indented by the incremental value.

INDIRECT

Returns to normal disc-buffered operation from DIRECT mode. The memory buffer is always dumped.

INPUT ["prompt" $\left\{\begin{matrix};\\,\end{matrix}\right\}$] variable name$_1$
[, ["prompt" $\left\{\begin{matrix};\\,\end{matrix}\right\}$] variable name$_2$...]

Suspends program execution, allowing values to be assigned to variables from the keyboard. Program execution is resumed by pressing [illegible] . INPUT cannot be executed from the keyboard. Omitting all parameters simply suspends program execution and returns keyboard control until [illegible] is pressed.

INT (numeric expression)

The integer function returns the greatest integer which is less than or equal to the evaluated expression.

INTEGER numeric variable$_1$ [(subscripts)]
[, numeric variable$_2$ [(subscripts)] , ...]

Dimensions and reserves storage space for integer-precision variables. INTEGER cannot be executed from the keyboard.

L

LDISP [display list]

Displays the list of items on the next unprotected display line. The remainder of the display line is cleared.

LENTER string variable name

Reads the current display line into the string variable and continues program execution. LENTER cannot be executed from the keyboard.

LEN (string expression)

The length function returns the current character length of the string expression.

[LET] { numeric variable$_1$ [= numeric variable$_2$...] = numeric expression
string variable$_1$ [= string variable$_2$...] = string expression }

Assigns a value to one or more variables.

LGT (numeric expression)

The common log function returns the common logarithm (base 10) of a positive numeric expression.

LIN (number of line feeds)

The line function is used with PRINT and DISP, causing a carriage return and the specified number of line feeds to be output. The range of the numeric expression specifying the number of line feeds is from −32768 thru 32767; a negative number suppresses the carriage return.

LINK file spec [, first line id [, execution line id]]

Brings into memory a program saved with SAVE, or any string data file consisting of valid BASIC statements, without erasing the values of variables. If the first line id is specified, the loaded program is renumbered so that it begins with the number of the specified line. The second line id specifies where execution is to continue.

LINPUT ["prompt" {; ,}] string variable

Suspends program execution, allowing any combination of characters to be entered and assigned to one string variable. Pressing ⇧ resumes program execution. LINPUT cannot be executed from the keyboard.

LIST [beginning line id [, ending line id]]

This command outputs a listing of all or part of program lines in memory in order from lowest numbered to highest numbered line. If one line id is specified, the listing begins with that line. If two line ids are specified, that block of lines is listed. The listing is output to the system printer.

LIST KEY# [SFK number list]

This command lists the typing-aid definitions of either the specified special function keys or all definitions when the list is omitted.

LOAD file spec [, execution line id]

Puts back into memory a program stored with STORE, erasing any BASIC lines and binary programs in memory. The execution line id specifies where execution is to begin.

LOAD PATH file spec

Loads the specified binary file into memory without altering any other binary routines already in memory.

LOAD KEY file spec

Loads special function key typing-aid definitions from a file stored with STORE KEY. Program lines in memory are not affected.

LOAD SUB file spec [, line number[, increment]]
[; starting segment [, last segment]]

Loads one or more subprograms from a program file, placing them at the end of the program currently in memory. The starting segment and last segment parameters allow specifying to load only those subprograms into memory.

LOCK# file number [, wait variable]

Used to restrict mass storage data-file access to the issuing console. The optional wait parameter indicates whether the console should wait for access to the specified file if it is already LOCKed. 0 indicates to wait for access; any other value indicates not to wait for access. The drive status is returned to the wait variable. The UNLOCK# statement releases the file for other consoles to use.

LOG (numeric expression)

The natural log function returns the natural logarithm (base e) of a positive numeric expression.

LOOP

This statement begins a structured LOOP block. The block of statements are continually executed until the expression in an EXIT IF statement is true.

LWC$ (string expression)

The lowercase function returns a string with all uppercase letters converted to lowercase.

M

MASS STORAGE IS volume spec

Specifies the default mass storage device. The volume spec is a string specifying either a volume name or a unit spec.

MAT array variable = CON [(redim subscripts)]

The MAT ... CON statement assigns the value 1 to every element in a numeric array. A new working size can be specified.

MAT result vector = CSUM operand matrix

The MAT-CSUM statement finds the sums of the elements of the columns of a numeric matrix and stores them in a vector.

MAT matrix name = IDN [(redim subscripts)]

The MAT ... IDN statement establishes an identity matrix: all elements equal zero except the main diagonal (upper left to lower right) which all equal one. A new working size can be specified; it must have two dimensions.

MAT INPUT array variable $_1$[(redim subscripts $_1$)]
[, array variable $_2$[(redim subscripts $_2$) , ...]

The MAT INPUT statement assigns values from the keyboard to elements of an array during program execution.

MAT result matrix = INV operand matrix

The MAT ... INV statement establishes a square matrix as the inverse of the specified square matrix.

MAT PRINT array,$_1$ [$\left\{\begin{matrix};\\,\end{matrix}\right\}$[array$_2$ [$\left\{\begin{matrix};\\,\end{matrix}\right\}$...]]]

The MAT PRINT statement outputs the specified arrays on the standard printer.

MAT PRINT # file number [, record number] ; array$_1$
[, array$_2$, ...] [, END]

The MAT PRINT # statement records all elements of the specified arrays onto a mass storage medium. END prints an EOF after the data.

MAT READ array$_1$ [(redim subscripts$_1$)]
[, array$_2$ [(redim subscripts$_2$)] , ...]

The MAT READ statement reads values for all the elements in an array or arrays from a DATA statement or statements which specify the values.

MAT READ# file number [, record number] ; array$_1$
[(redim subscripts$_1$)] [, array$_2$[(redim subscripts$_2$)] , ...]

The MAT READ# statement reads values for the elements of the specified arrays from a mass storage medium.

MAT result vector = RSUM operand matrix

The MAT ... RSUM statement finds the sums of the elements of the rows of a numeric matrix and stores the sums in a vector.

MAT result matrix = TRN operand matrix

The MAT ... TRN statement establishes a matrix as the transpose of a specified matrix (rows become columns, columns become rows). A matrix can't be transposed into itself.

MAT array variable = ZER [(redim subscripts)]

The MAT ... ZER statement sets all elements in a numeric array to zero. The array can be redimensioned.

MAT array variable = (numeric expression)

The MAT-initialize statement assigns the value of the expression to every element in a numeric array.

MAT result array = function   operand array

The MAT-function statement evaluates each element in the operand numeric array by the specified system function; the result becomes the value of the corresponding element in the result array. The function can be any single-argument system function.

MAT result array = operand array

The MAT-copy statement copies the value of each element in a numeric array into a second numeric array.

MAT result array = operand array operator operand array

The MAT-operation statement allows an arithmetic or relational operation to be performed on corresponding elements of two numeric arrays; the result becomes the value of the corresponding element in the result array. The following operators are allowed -

+, -, *, /, =, <, >, <=, >=, <> or #

MAT result array = operand array operator (scalar)
MAT result array = (scalar) operator operand array

The MAT-scalar operation statement performs an arithmetic or relational operation on each element of a numeric array using a constant scalar (numeric expression); the result becomes the value of the corresponding element of the result array. The following operators are allowed -

+, -, *, /, =, <, >, <=, >=, <> or #

MAT result matrix = operand matrix$_1$ * operand matrix$_2$

The matrix multiplication statement multiplies two numeric matrices together. The number of columns of the first operand must equal the number of rows of the second.

MAX (list)

This function returns the largest value in the list of numeric expressions.

MERGE file spec [, line id[, execution line id]]

Takes program lines from a data file and positions them in memory, either in front of the program currently there, or between consecutive lines in the program currently there, or behind the program currently there. If the first line id is specified, the program lines in the specified file are renumbered beginning with that line. If two line ids are specified, execution begins with the second line id specified.

MIN (list)

This function returns the smallest value in the list of numeric expressions.

MSI volume spec

This is an abbreviated form of the MASS STORAGE IS statement.

N

NEXT loop counter

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

NUM (string expression)

The numeric function returns the decimal equivalent of the 8-bit binary value of the first character of the string expression.

O

OFF END# file number

Cancels any previous ON END# condition currently active for the specified data file number.

OFF ERROR

Cancels any ON ERROR condition currently active.

OFF HALT

Cancels any ON HALT condition.

OFF KEY# [key number] [, keynumber]

Deactivates a corresponding ON KEY# statement; pressing the special function key then has no effect on program control. Omitting the key number deactivates all ON KEY definitions.

$$\texttt{ON END\#}\ \text{file number}\ \left\{\begin{array}{l}\texttt{GOTO}\ \text{line id}\\ \texttt{GOSUB}\ \text{line id}\\ \texttt{CALL}\ \text{subprogram name}\end{array}\right\}$$

Declares a branch to occur when an EOF mark is encountered during a READ# or PRINT# operation to a data file, thus avoiding an end-of-data error message. No parameters can be passed to the subprogram when CALL is used.

$$\texttt{ON ERROR}\ \left\{\begin{array}{l}\texttt{GOTO}\ \text{line id}\\ \texttt{GOSUB}\ \text{line id}\\ \texttt{CALL}\ \text{subprogram name}\end{array}\right\}$$

Used to prevent some recoverable program execution errors from halting execution by causing branching when an error occurs and suppressing the normal error process. No parameters can be passed to the subprogram when CALL is used.

ON numeric expression GOSUB line id list

Accesses any one of the subroutines listed based on the value of the numeric expression. A value of 1 corresponds to the first line id in the list, 2 to the second, etc.

ON numeric expression GOTO line id list

Transfers program control to one of the lines listed, based on the value of the numeric expression. A value of 1 corresponds to the first line id in the list, 2 to the second, etc.

$$\texttt{ON HALT}\ \left\{\begin{array}{l}\texttt{GOTO}\ \text{line id}\\ \texttt{GOSUB}\ \text{line id}\\ \texttt{CALL}\ \text{subprogram name}\end{array}\right\}$$

Activates a branching condition which occurs when HALT is pressed. The branch occurs only during program execution or during the INPUT state.

ON KEY#key number$_1$[, #keynumber$_2$, . . . , #keynumber$_n$]

[, priority] ["label] $\left\{\begin{array}{l}\text{GOSUB line id}\\ \text{GOTO line id}\}\\ \text{CALL subprogram name}\end{array}\right\}$

Allows any special function key (SFK) to be used for program control. When an SFK is pressed during a program and an ON KEY# statement has been declared for it, the specified branching occurs if the specified priority is higher than the current system priority. The range of priority is an integer expression from 1 thru 15. The label is a string expression which appears above a display SFK as its label.

OPTION BASE $\left\{\begin{array}{l}0\\ 1\end{array}\right\}$

Allows the default lower bound of arrays to be specified as 1 rather than 0. OPTION BASE 0 can be declared for documentation purposes since it is the default state. The OPTION BASE statement must be placed before any DIM, COM, REAL, SHORT, and INTEGER statements. OPTION BASE cannot be executed from the keyboard.

P

PAGE

This function is used with PRINT, DISP, or LDISP and causes a form feed to be output. Up to 75 lines are searched for a top-of-form indicator on the internal printer.

PAUSE

Suspends program execution. Use the CONTINUE command to resume execution at the next line. PAUSE cannot be executed from the keyboard.

PI

This function returns the value of PI (π) which is about 3.14159265360.

POS (string expression$_1$, string expression$_2$)

This function determines the position of the second string within the first string. POS returns either the character position of the first character of the second string within the first, or 0 if the second string is not present.

PRINT [print list]

Causes the items specified in the print list to be output on the standard printer. The items can be variables, array identifiers, and expressions (excluding multiple line user-defined functions). The TAB, SPA, LIN, and PAGE functions are allowed. Each item must be separated by a comma or semicolon. Two commas in a row cause a field to be skipped. A CRLF is output if no print list is included. A comma or semicolon at the end of the list suppresses the normal CRLF.

PRINT # file no. { ; [data list][, END]
, record no. ; [data list][, END]
, record no. , word pointer ;
[data list][, END] }

Used to record values onto the specified data file. In serial access mode (first syntax), recording starts either at the beginning of the file or after the last data item accessed. In direct access mode (second syntax), recording starts at the beginning of the record. When the word pointer is specified (third syntax), a serial access is performed, beginning at the specified point. The data list can include variables, constants and literals, separated by commas. END causes an EOF to be printed after the data. Otherwise an EOR is printed after the data list (except when the word pointer is specified). When the data list is omitted in direct access mode, an EOR is printed in that record.

PRINT USING { image format string
line id } [; print-using list]

Allows the exact form of printed output to be determined by the image string. The print-using list can contain variables, array identifiers and expressions. Each item must be separated by a comma or semicolon, and must correspond to an appropriate field specifier in the image string. The line id refers to an IMAGE statement.

PRINT ALL IS {device address
file spec [, SIZE records]}
[, WIDTH line width][, TRANSPARENT]

Defines the destination for operator/system interactions. The device address specifies an alternate output device, while the file spec creates a spool file where all displayed data is copied. Omitting the SIZE creates a file of 100 records. The range of the WIDTH is from 20 thru 264 characters; 80 is default for the display and 132 for external devices; −1 sets an infinite line width. TRANSPARENT specifies no HP250 interpretation of control codes.

PRINTER IS {device address
file spec [, SIZE records]}
[, WIDTH line width][, TRANSPARENT]

Defines the destination for all successive PRINT and PRINT USING outputs. The display is set to be the standard printer at power-up. Other parameters are the same as those used with PRINT ALL IS.

PRINT LABEL string expression [ON volume spec]

Writes a label on the specified storage medium. The string can be from one thru eight characters. The label is written on the standard medium unless the optional ON volume spec is given.

PROTECT file spec, protect code

Protects a file against accidental erasure. The protect code is any valid string expression (except a null string); only the first six characters are recognized.

PROUND (numeric expression, power-of-ten position)

The position round function returns the numeric expression rounded to the specified power-of-ten position.

PURGE file spec [, protect code]

Erases the specified file from the storage medium. The protect code is needed only if the file was previously protected.

R

RANDOMIZE [numeric expression]

Re-initializes the random number seed.

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

Specifies variables for which values are to be assigned from a DATA statement. READ cannot be executed from the keyboard.

READ LABEL { string array name
string variable [ON volume spec] }

Returns the mass storage volume label(s) currently in use. The label of the specified volume is returned to a single string variable, while the labels of all media currently in use are returned to an array variable. An * is returned with the label of the currently-set default device. A ? indicates that the device is not ready.

READ# file no. { ; variable list
, record no. [; variable list]
, record no. , word pointer[; variable list] }

Retrieves values for variables from the specified file. In serial access mode (first syntax), reading starts at the beginning of the file or after the last data item accessed. In direct access mode (second syntax), reading starts at the beginning of the logical record. Including a word pointer (third syntax) begins reading data at the specified word in the logical record. READ# can also be used to reposition the data pointer by omitting the variable list in direct access mode. Each variable in the list must be separated by a comma.

REAL numeric variable$_1$ [(subscripts)]
[, numeric variable$_2$ [(subscripts)] , ...]

Dimensions and reserves storage space for non-subscripted and array variables and declares them as real (full) precision. REAL cannot be executed from the keyboard.

REC (file number)

The record function returns the current position of the record pointer for the specified mass storage file. Specifying a negative file number returns the physical record address for a specified file.

REDIM array variable$_1$ (subscripts)
[, array variable$_2$ (subscripts) , ...]

Defines a new working size for an array. The total number of elements cannot exceed that originally declared. The number of dimensions cannot change.

RELEASE device address

Cancels any REQUEST for exclusive use of a peripheral device by that console. See REQUEST.

REM [any combination of characters]

Allows insertion of non-executable remarks into the listing of a program, to provide documentation and make the program easier to follow.

REN [beginning line number [, increment value]]

This command allows the program in memory to be renumbered. If no parameters are specified, numbering begins with 10 and is incremented by ten.

RENAME old file spec TO new file name [, protect code]

Renames an existing mass storage file. The protect code is needed only if the file was previously protected.

REPEAT

This statement begins a REPEAT block.

REQUEST device address[, wait variable]

Reserves exclusive use of the specified peripheral device for the console. (Discs cannot be reserved.) The value of the wait variable indicates whether the program should wait for access to the device or continue. The peripheral's status is returned to the wait variable. Use RELEASE to cancel REQUEST.

RES

This function returns the result of the last numeric computation executed from the keyboard.

{ RESAVE
RE-SAVE } file spec [, protect code]
[, beginning line id [, ending line id]]

Saves a program into a file previously created by SAVE. The protect code is used only if the file was previously protected. When no line ids are specified, the entire program is saved. When one line id is specified, the program is saved from that line to the end. When both line identifiers are specified, that block of lines is saved.

RE-STORE [KEY
BIN] file spec [, protect code]

Stores a program, SFK typing-aid definitions (RE-STORE KEY), or a binary program (RE-STORE BIN) into a file previously created with STORE, STORE KEY, or STORE BIN (respectively).

RESTORE [line id]

Repositions the DATA pointer either to the beginning of the specified DATA statement, or at the lowest numbered DATA statement in the current program segment if one is not specified, so that the values can be reused. If the line specified is not a DATA statement, the pointer is positioned at the first DATA statement following that line. RESTORE cannot be executed from the keyboard.

RETURN [numeric expression
string expression]

With no expression, this statement is the last line in a subroutine and transfers control back to the line following the GOSUB statement. RETURN is also used with DEF FN to specify the value to be returned to the calling program and transfer control back to the statement which referenced the function subprogram.

REVISION

The revision function returns a value indicating the current operating system revision level.

RND

The random function generates a pseudo random number greater than or equal to 0 and less than 1.

ROW operand array

This function returns the number of rows in the specified array.

RPT$ (string expression , number of repetitions)

The repeat function returns the string expression repeated the specified number of times. The range of repetitions is from 0 thru 32767.

RUN [line id
file spec[, line id]]

This command begins execution of a program at either the specified line or the lowest numbered line in memory (no parameters). The specified line must be in the main program. If a file spec is given, the program is automatically loaded from the file and run, starting at either the first line (no line id) or the specified line.

S

SAVE file spec [, beginning line id [, ending line id]]

Lists and records either all or some of program lines in memory into a data file. If one line id is specified, the program is saved from that line to the end. When both line ids are specified, that block of lines is saved.

```
        ┌ A                  ┐
        │ C                  │
SCRATCH │ KEY# [key number]  │
        │ P                  │
        └ V                  ┘
```

Without a parameter, this statement erases programs and variables from memory. Specifying a parameter:

SCRATCH A erases the entire memory (like HALT).

SCRATCH C erases all variables including those in com.

SCRATCH KEY # [key number] erases all or only the specified SFK typing-aid definitions.

SCRATCH P erases programs and variables.

SCRATCH V erases all variables except those in com.

SCRATCH ▭ erases the SFK typing-aid definition.

SECURE [line id [, line id]]

Prevents either selected lines or an entire program from being viewed; an asterisk appears after the line number replacing the line in the listing. If one line id is specified, only that line is secured. If both line identifiers are specified, that block of lines is secured.

SELECT conditional expression

This statement begins a structured SELECT block. If the expression is within the range of any CASE list within the block, execution resumes after that CASE statement. Otherwise execution resumes after the CASE ELSE or END SELECT statement.

SGN (numeric expression)

The sign function returns a 1 if the expression is positive, 0 if it is zero and −1 if it is negative.

SHORT numeric variable$_1$ [(subscripts)]
[, numeric variable$_2$ [(subscripts)] , ...]

Dimensions and reserves storage space for simple and array variables and declare them as short precision. SHORT cannot be executed from the keyboard.

SIZE (file number)

This function returns the size of the specified file, in logical records. Specifying a negative file number returns the logical record size of a file record, in words.

SLEN (file number)

The string length function returns the number of string characters in the file, beginning at the current word pointer location. A return value of −1 indicates that string data is not found.

SPA (number of spaces)

The space function is used with PRINT and DISP to output a specified number of blank spaces, up to the end of the current line. The number of spaces is a positive numeric expression rounded to an integer.

SQR (numeric expression)

This function returns the square root of a non-negative expression.

STANDARD

Sets standard mode for output of numeric values.

STOP

Terminates program execution and sets the program pointer to the lowest numbered line.

STORE file spec

Stores all program lines and binary routines in memory into a program file on the specified storage device.

STORE BIN file spec

Stores all user binary programs from memory into a file.

STORE KEY file spec

Stores all special function key (SFK) typing-aid definitions into a special key file.

SUB subprogram name [(formal parameter list)]

The first line of a subprogram.

SUB END

The last line in a subprogram, transferring control back to the calling program.

SUB EXIT

Transfers control from a subprogram back to the calling program before SUB END is executed.

SUM (operand array)

This function returns the sum of all elements in the specified array.

```
SYSTEM PRINTER IS {device address
                  {file spec [, SIZE] records]}
                        [, WIDTH width][, TRANSPARENT]
```

Specifies the system printer, which is the destination for all successive LIST, CAT, TRACE, and STEP (step program) outputs. The display (device address 8) is set to be the system printer at power-up. See PRINT ALL IS for details on other parameters.

T

TAB (character position)

This function is used with PRINT, DISP and LDISP and causes the next item to be output beginning at the specified character position. The TAB function is independent of the TAB and ⇄ keys.

TASKID

This function returns the task id number, or number assigned to the workstation currently active (same as USRID).

TIME$

Returns the current system time in the form hh:mm:ss if one has been set.

TRIM$ (string expression)

This function returns the string expression with any leading and trailing blanks deleted.

TYP (file number)

The type function returns a value which indicates what type of data will be accessed next in the specified file. Specifying a positive file number allows the data pointer to advance until it is positioned on something other than an EOR mark. A negative file number suppresses movement of the data pointer.

Return Value	Meaning
0	Unidentified type
1	Real-precision number
2	Complete string
3	End-of-file mark
4	End-of-record mark
5	Integer-precision number
6	Short-precision number
7	(unused)
8	First part of a string
9	Middle part of a string
10	Last part of a string

U

UNLOCK# file number

Unlocks a data file previously LOCKed, for use by other consoles. See the LOCK# statement.

UNTIL conditional expression

This statement terminates a REPEAT block. The block of statements are repeatedly executed until the conditional expression is true.

UPC$(string expression)

The uppercase function returns a string with all lowercase letters converted to uppercase.

USRID

This function returns the user id number, or number assigned to the console currently active.

V

VAL (string expression)

The value function returns the numeric value, including any exponent, of a string of digits so that the value can be used in calculations.

VAL$ (numeric expression)

This function returns a string representing the numeric expression in current output mode.

W

WAIT [number of millseconds]

Delays program execution the approximate number of milliseconds before continuing. The range of the numeric expression is from −32768 thru 32767; a negative number defaults to 0. If the number of milliseconds is not specified, program execution will be delayed indefinitely. The wait is cancelled by pressing HALT or a defined softkey.

WHILE conditional expression

This statement allows repeated executed of a block of statements until the conditional expression is false. The block is terminated with an END WHILE statement.

WRD (file number)

The word function returns the position of the word pointer in the specified mass storage file.

X

XPOS

This function returns the current X-axis position of the display cursor.

Y

YPOS

This function returns the current Y-axis position of the cursor, relative to the first line in the display buffer.

DBML and Utility Statements

These parameters are used in describing DBML and utilities.

base$ – A string variable which contains the data base name.

set – A numeric expression evaluating to a data set number.

set$ – A string expression evaluating to a data set name.

mode – A numeric expression evaluating to a valid mode.

status (*) – An integer array containing at least 10 elements in right-most dimension, used to return status codes on most DBML statements.

list$ – A string expression evaluating to either "@", "@;"or "@ ". In all but the first case, any arbitrary character sequence may also follow.

buf$ – A string variable, without any substring specifiers, which is used to transfer information between the BASIC program and the data base.

qual – A numeric expression evaluating to a valid item, set or volume number.

qual$ – A string expression evaluating to a valid item, set or volume name.

maint$ – A string expression evaluating to the maintenance password.

set list$ – A string expression evaluating to a list of set numbers or names separated by commas. An * may be used, depending on the statement.

item list – A list of string or numeric variables (or arrays) and SKPs which correspond to items in the data set specified in an IN DATA SET statement.

line list – A list of line numbers or labels which appears in an IN DATA SET ... USE REMOTE LISTS statement. Each line id must refer to an IN DATA SET LIST statement.

return var – A numeric expression to which the final execution status of the statement is assigned.

backup$ – A string expression evaluating to the name of the backup file.

vol list$ – A string expression evaluating to a list of backup volume names separated by commas.

vol spec$ – A string expression evaluating to a volume label or device specifier.

arg – A numeric expression evaluating to either a record number (mode 4) or a master set numeric search item value (mode 7).

arg$ – A string expression evaluating to a master set string search item value.

item – A numeric expression evaluating to a data item number (corresponding to a detail search item).

item$ – A string expression evaluating to a data item name (corresponding to a detail search name).

value – A numeric expression evaluating to a detail search item value.

value$ – A string expression evaluating to a detail search item value.

pass$ – A string expression containing a left-justified string.

```
DBCLOSE (base$ ; {set } ; mode ; status )
                 {set$}
```

Terminates access to a data base.

——— modes ———

1 – Closes data base.
3 – Rewinds data set.
4 – Dumps data buffer and updates DBCB.

```
DBCREATE base$[ ; maint$][ ; set list$ ][ ; return var ]
                         [ ; vol spec$ ]
```

Creates and initializes all or only selected data sets.

```
DBDELETE (base$ ; {set } ; mode ; status (*) )
                  {set$}
```

Deletes existing entries from a data set. Specify mode 1 to delete the current entry.

```
DBERASE base$[ ; maint$][ ; set list$ ][ ; return var ]
                        [ ; vol spec$ ]
```

Erases existing data entries from all or selected data sets.

```
DBFIND (base$ ; {set } ; mode ; status(*) ; item$ ; {value } )
                {set$}                              {value$}
```

Locates the first and last entries of a data chain in a detail data set in preparation for access to that chain. Specify mode 1 to find head of chain. The item is a detail search item name or number.

```
DBGET (base$ ; {set } ; mode ; status (*) ;
               {set$}
                                   list$ ; buf$ ; {arg } )
                                                  {arg$}
```

Reads the data items of a specified entry in a data set.

——— modes ———

2 – serial read forward, argument parameter ignored.
4 – directed read, argument contains record number.
5 – chain read forward, argument parameter ignored.
7 – calculated read, argument contains word key.

DBINFO (base$, $\left\{\begin{array}{l}\text{qual}\\ \text{qual\$}\end{array}\right\}$, mode, status(*), buf$)

Provides information about the data base, such as the name and field description of data items.

— modes —

101 – identifies data item number for a given data item name.
102 – describes a specific data item for a given data item name or number.
104 – identifies all data items for a given data set name or number.
201 – identifies a data set number for a given data set name.
202 – describes a specific data set for a given data set name or number.
203 – identifies all data sets for a given data base (qualifier ignored).
204 – identifies all data sets containing a given data item name or number.
301 – identifies all data paths for a given data set name or number.
302 – identifies a search item for a given data set name or number.
401 – identifies a volume number for a given data set name or number.
402 – identifies a volume name for a given volume number.
403 – identifies all volumes for a given data base (qualifier ignored).
404 – identifies al data sets for a given volume name or number.

DBLOCK (base$, $\left\{\begin{array}{l}\text{set}\\ \text{set\$}\\ \text{P\$}\end{array}\right\}$, mode, status(*))

Locks a data base to allow the user exclusive write access.

— modes —

1 – with wait, write access to entire data base.
2 – without wait, write access to entire data base.
3 – with wait, write access to specified data set.
4 – without wait, write access to specified data set.
5 – with wait, write access to specified predicate.
6 – without wait, write access to specified predicate.
11 – with wait, read access to entire data base.
12 – without wait, read access to entire data base.
13 – with wait, read access to specified data set.
14 – without wait, read access to specified data set.
15 – with wait, read access to specified predicate.
16 – without wait, read access to specified predicate.

DBOPEN (base$, pass$, mode, status(*))

Initiates access to a data base. Sets up the access mode and user class number for the specified data base.

— modes —

1 – modify shared with data base locking.
3 – modify exclusive.
8 – read shared.

DBPURGE base$; [maint$] $\left[\begin{matrix}\text{, set list\$}\\ \text{, vol spec\$}\end{matrix}\right]$ [, return var]

Purges specific data sets or the entire data base, including the root file and all its data sets.

DBPUT (base$, $\left\{\begin{matrix}\text{set}\\ \text{set\$}\end{matrix}\right\}$, mode , status (*) , list$, buf$)

Adds new entries to a data set. Specify mode 1 to put a new entry in data set.

DBRESTORE backup$ [ON vol spec$]

Re-stores the data base using data in a BKUP file created with DBSTORE. To load this binary statement from the UTILITY disc, execute LOAD BIN "DBSTOR".

DBSTORE base$ [; maint$] [, set list$] TO backup$
[ON vol list$]

Copies all or specified data sets of a given data base to a backup (BKUP) file. To load this binary statement, execute LOAD BIN "DBSTOR".

DBUNLOCK (base$, $\left\{\begin{matrix}\text{set}\\ \text{set\$}\end{matrix}\right\}$, mode , status (*))

Unlocks a data base that was locked with a previous DBLOCK. Specify mode 1 for unconditional unlocking of the data base.

DBUPDATE (base$, $\left\{\begin{matrix}\text{set}\\ \text{set\$}\end{matrix}\right\}$, mode , status (*) , list$, buf$)

Modifies specified item values in an entry. (Search items may not be modified.) Specify mode 1 to update non-search item values in the current entry.

PREDICATE P$ FROM $\text{set\$}_1$ [, item$
[, relop$, value$]] [; $\text{set\$}_2$...[; $\text{set\$}_n$...]

The variable P$ identifies the sets and/or items which are to be locked with the DBLOCK statement. These relops are allowed = or EQ , <= or LE , >= or GE . The alphabetical relops are for European character sets.

EDITOR Commands

RUN"EDITOR [volume spec]"

Load and run EDITOR program.

$\left\{\begin{matrix}\text{A}\\\text{ADD}\end{matrix}\right\}$ [@] [line number] [,HOLD]

Adds lines to the text file.

$\left\{\begin{matrix}\text{C}\\\text{CHANGE}\end{matrix}\right\}$ [@] string$_1$ TO string$_2$ [IN range list]

Changes character strings in the text file.

$\left\{\begin{matrix}\text{D}\\\text{DELETE}\end{matrix}\right\}$ [@] [range list]

Deletes lines in the text file.

$\left\{\begin{matrix}\text{E}\\\text{EXIT}\\\text{END}\end{matrix}\right\}$

Terminates the EDITOR program.

$\left\{\begin{matrix}\text{F}\\\text{FIND}\end{matrix}\right\}$ [@] $\left[\begin{matrix}\text{string [IN range list]}\\\text{line number}\end{matrix}\right]$

Finds specified character strings or current line position.

$\left\{\begin{matrix}\text{G}\\\text{GATHER}\end{matrix}\right\}$ ALL [TO line number] [BY increment value]

Renumbers a text file.

$\left\{\begin{matrix}\text{H}\\\text{HOLD}\end{matrix}\right\}$ [@] [range] [,APPEND]

Saves lines from the text file into the hold file.

$\left\{\begin{matrix}\text{K}\\\text{KEEP}\end{matrix}\right\}$ "file spec" $\left[\begin{matrix}\text{,UNN}\\\text{,UNNUMBERED}\end{matrix}\right]$

Saves the text file as a data file.

$\left\{\begin{matrix}\text{L}\\\text{LIST}\end{matrix}\right\}$ [@] [range] [,OFFLINE]

Lists lines from the text file to the display or printer.

```
{   M    }
{ MODIFY } [range list]
```

Modifies lines in the text file.

```
{  S  } { LENGTH = nnn                    }
{ SET } { PRINTER = n [; WIDTH = nnn]     }
        { LINES = nnn                     }
```

Sets EDITOR parameters.

```
{  T   }            [; UNN        ]
{ TEXT } "file spec " [; UNNUMBERED ]
```

Copies a data file into the text file.

```
{   W   }
{ WHILE }
```

Repeats a group of EDITOR commands.

SCHEMA Commands

```
RUN "SCHEMA [volume spec] "
```

Loads and runs the SCHEMA program.

```
$TITLE[ "character string "]
```

Specifies a character string to be printed at the top of each page of the schema listing.

```
$PAGE[ "character string "]
```

Causes a form feed during the schema listing and prints an optional string in place of the $TITLE string.

```
$CONTROL option list
```

The option list can include:

`LIST`   List each source record from the text file.

NOLIST Turn off the LIST option.

ROOT Build a ROOT file.

NOROOT Turn off the ROOT option.

TABLE List data set information following schema listing.

NOTABLE Turn off TABLE option.

ERROR=nnn Set maximum number of errors to occur before schema operation is halted.

LINES=nnn Set maximum number of lines per page of schema listing.

Default $CONTROL command is:

```
$CONTROL LIST,ROOT,TABLE,
                   ERRORS=100,LINES=66
```

Advanced Data Access

DBASE IS base$

Defines the IMAGE/250 data base to be used prior to the IN DATA SET statement.

```
                                  ( USE ALL        )
IN DATA SET  set$  [IN COM]      {  USE item list   }
                or                ( DIM ALL        )
IN DATA SET  set$   FREE
                or
IN DATA SET  set$  [IN COM]USE REMOTE LISTS
                                           line id list
               with
```

IN DATA SET LIST item list

Automatically packs the buffer parameter during DBPUT and DBUPDATE. Automatically unpacks the buffer after DBGET.

SCHEMA Definition

```
BEGIN DATA BASE base name [:volume name];
PASSWORDS:
       ucn  password;
                .
                .
                .
       ucn  password;
ITEMS:
       item name,[sub-item count]type spec[(control no.)];
              .
              .
              .
SETS:
       list of set definitions
END.
```

base name — 1 thru 6 character data base name, beginning with a letter and containing uppercase letters, digits and dashes.

volume name — 1 thru 8 character string specifying a particular storage media. The name may not contain commas or semicolons.

ucn — A user class number. It is an integer from 1 thru 31.

password — A string of from 1 thru 8 characters not including semicolons. Imbedded blanks are removed.

item name — A 1 thru 15 character item name, beginning with a letter and containing letters, digits and dashes.

sub-item count — An integer from 1 thru 1022 (depending on item type) which specifies the replication count for the item whose type spec it precedes.

type spec — A specifier of the item type. It is either "L", "S", "I" or "X". In the case of "X", it is followed by an even integer from 2 thru 1022 specifying the string length.

control no. — An integer from 0 thru 127. This number may be retrieved by DBINFO. It is used by QUERY to determine the format used to print any data associated with that item.

Manual Master Set Definition

```
{NAME:} set name ; {MANUAL} (read list/write list)
{N:   }            {M     }              [ ; volume name] ;

{ENTRY:}  item name (path count) ;
{E:    }  item name ;
              •
              •
              •
          item name ;

{CAPACITY:}  max entry count ;
{C:       }
```

Automatic Master Set Definition

```
{NAME:} set name ; {AUTOMATIC} (read list/write list)
{N:   }            {A        }             [ ; volume name] ;

{ENTRY:} item name  (path count) ;
{E:    }

{CAPACITY:}  max entry count ;
{C:       }
```

Detail Data Set Definition

```
{NAME:} set name, {DETAIL} (read list/write list)
{N:   }           {D     }             [,volume name];

{ENTRY:}  item name [ (master set name) ],
{E:    }  item name [ (master set name) ],
                    •
                    •
                    •
          item name [ (master set name) ];

{CAPACITY:}  max entry count;
{C:       }
```

set name – A 1 thru 15 character set name, beginning with a letter and consisting of uppercase letters, digits and dashes.

read list – A list of user class numbers(including 0) separated by commas. The list may be null.

write list – A list of user class numbers (including 0) separated by commas. The list may not be null.

path count – An integer from 1 thru 8 corresponding to the number of paths between this master and the associated detail sets.

max entry count – An integer from 1 thru 65534 which specifies the maximum number of entries allowed in the set to which it pertains.

master set name – The name of a previously listed master data set.

PACKFMT pack variable list

Lists the program variable names to be stored or retrieved from a list of variables, specifying the data format for PACK USING and UNPACK operations.

PACK USING line id # destination string

Packs the items specified by a packing list (via line id) into the destination string. The string is then used by DBPUT, DBUPDATE or non-IMAGE/250 operations.

UNPACK USING line id # source string

Works opposite of PACK USING, unpacking the source string for DBINFO, DBGET or non-IMAGE/250 operations into variables specified by PACK FMT.

FIND $\left\{ \begin{array}{l} \text{ALL} \\ \text{condition} \end{array} \right\}$

Select a subset of records from the data base or the current workfile. FIND ALL is equivalent to FIND 1=1, and gets all records in unsorted order.

SORT BY variable$_1$[DES] [, ... , variable$_{10}$[DES]]

Specify the order in which data is to be sorted.

WFLEN (file number)

This function returns the length of the specified workfile in logical records.

WORKFILE IS # file number

[; THREAD IS [set id $\left\{ \begin{array}{l} \text{LINK link} \\ \text{, path id} \end{array} \right\}$, ...] set id]

Specifies the hierarchical structure (thread) of the data sets to be sorted, the work space for sorting, and the workfile name. Up to 10 thread sets can be listed. The path id can be from 1 thru 8.

Description Statements

BREAK level WHEN control CHANGES [BY increment]

Establishes the criteria for determining the level break condition.

END REPORT DESCRIPTION

Ends the description section.

GRAND TOTALS ON exp_1[; exp_2...]

Provides automatic totaling for the entire report.

HEADER level [WITH number LINES]

[USING $\left\{\begin{matrix}\text{line id}\\ \text{image string}\end{matrix}\right\}$ [; list]]

Defines what is to be done as a heading when the specified level break occurs. The USING parameters are the same as in a PRINT USING statement.

LEFT MARGIN column

Sets the column in which each line of the report will begin.

PRINT DETAIL IF condition expression

Causes exceptional detail lines only to be printed without affecting the totaling functions.

[label :] REPORT HEADER [WITH number LINES]

[USING $\left\{\begin{matrix}\text{line id}\\ \text{image string}\end{matrix}\right\}$[; list]]

Begins the description section. It specifies what is to be done at the beginning of the report. The USING parameters are the same as in a PRINT USING statement.

PAGE HEADER [WITH number LINES]

[USING $\left\{\begin{matrix}\text{line id}\\ \text{image string}\end{matrix}\right\}$ [; list]]

Defines what is to be done at the top of every page. The USING parameters are the same as in a PRINT USING statement.

PAGE LENGTH lines per page [; blank top ; blank bottom]

Specifies the number of lines there are on the page. The number of blank lines to be printed at the top and bottom of the page may also be specified.

PAGE TRAILER [WITH number LINES]

[USING $\left\{\begin{matrix}\text{line id}\\ \text{image string}\end{matrix}\right\}$ [; list]]

Defines what is to be done at the bottom of each page. If more than one line is used, that number should be specified. The USING parameter are the same as in a PRINT USING statement.

PAUSE AFTER number PAGES

Causes a pause to occur after the specified number of pages has been output. Press ⇧ to resume output.

REPORT EXIT (exec flag) [WITH number LINES]

[USING $\left\{\begin{matrix}\text{line id}\\ \text{image string}\end{matrix}\right\}$ [; list]] [block statements]

The REPORT EXIT statement defines the action to be taken when the report is prematurely stopped.

REPORT TRAILER [WITH number LINES]

[USING $\left\{\begin{matrix}\text{line id}\\ \text{image string}\end{matrix}\right\}$ [; list]]

Defines what is to be done at the end of the report. If more than one line is required, that number should be specified. The USING parameters are the same as in a PRINT USING statement.

SUPPRESS PRINT AT level

Specifies the level of headers and trailers that will be printed. Those with equal or higher levels will not be printed.

SUPPRESS PRINT FOR number PAGES

Causes the first specified number of pages not to be printed.

TOTALS ON exp_1[; exp_2...]

Provides automatic totaling for a break level. It immediately follows a header statement.

TRAILER level [WITH number LINES]
[USING $\left\{\begin{array}{l}\text{line id}\\ \text{image string}\end{array}\right\}$ [; list]]

Defines what is to be done as a trailer for the specified break level. If more than one line is required, that number should be specified. The USING parameters are the same as in a PRINT USING statement.

Execution Statements

BEGIN REPORT line id

Initiates execution of a report, the description section of which is referenced by the line id.

DETAIL LINE detail [WITH number LINES]
[USING $\left\{\begin{array}{l}\text{line id}\\ \text{image string}\end{array}\right\}$ [; list]]

Causes all break conditions to be tested, totals to be incremented, and data to be printed. If more than one line is required for data output, that number should be specified. The USING parameters are the same as in a PRINT USING statement.

END REPORT

Causes final trailers to be executed and terminates the Report Writer.

NUMPAGE= expression

Causes the page counter to take the specified value.

STOP REPORT

Immediately terminates an active report. No trailing statements are printed.

TRIGGER BREAK level

Forces a break condition at the specified level.

TRIGGER PAGE BREAK

Forces a page to break.

Functions

AVG (level , sequence)

Returns the average for the specified expression in a TOTALS ON statement. The level of the TOTALS ON statement and the sequential position of the expression are specified.

LAST BREAK

Returns the value of the last break condition level number detected.

NUMBREAK level

Returns the number of times the specified level break condition has occurred.

NUMDETAIL level

Returns the number of DETAIL LINE statements that have been executed since the specified level header was last executed.

NUMLINE

Returns the current line number to which output will go.

NUMPAGE

Returns the current page number to which output will go.

OLDCV [$] (level)

Returns the value of the control variable as it was in the last level break condition. If the control variable is a string, the $ is appended to OLDCV.

RWINFO (integer)

This function returns Report Writer information.

Integer	Information Returned
1	Page size*
2	Effective page size = Page size − (blank lines top + blank lines bottom + lines in PAGE HEADER + lines in PAGE TRAILER)
3	Number of lines used in current page (same as NUMLINE)
4	Number of lines left in current page*
5	Number of lines left in effective page*
6	Page break cause flag: 0 = Not caused by DETAIL LINE 1 = Caused by DETAIL LINE
7	Page count (same as NUMPAGE)
8	Number of pages left to suppress
9	Number of logical pages produced
10	Same as LAST BREAK
11	Current LEFT MARGIN
12	Current HEADER/TRAILER level if in a break condition.

*will return a zero if pagination is turned off (PAGE LENGTH = 0).

TOTAL (level, sequence)

Returns the running total for the specified expression in a TOTALS ON statement. The level of the TOTALS ON and the sequential position of the expression are specified.

RUN"CFORM [volume spec]"

Load and run the Create Form program.

RUN"MFORM [volume spec]"

Load and run the Modify Form program.

CURSOR item list

These additional cursor control items are provided with FORMS/250:

IF# numeric expression	Set input field number.
OF# numeric expression	Set output field number.
CF# numeric expression	Move cursor to specified field.

See CURSOR in the "BASIC Syntax" section for more details.

CLEAR FORM

Erases input and output fields and resets field pointers.

DELETE FORM

Erases the form from the display and breaks the program-form link.

EXIT FORM

Breaks the link between the form and the program.

GET FORM "form name [volume spec]"

Displays a new form on the screen.

TFNUM

Returns the tab position of the cursor.

RUN "QUERY[volume spec]"

Begins QUERY operation.

ADD item list[FROM "form name"]

Adds entries to the data items or data set listed. A form can be used to input the values.

[BREAK ON item] [TOTAL item list]

Sets report breaks and their associated totals for the LIST or LINEAR LIST commands. TOTAL alone causes a grand total to be printed. A maximum of ten items can be totaled.

DATA BASE base name [volume spec]

Causes future commands to operate on the specified data base.

DELETE item list

Deletes data item values or entries from the data set. The data items must have been found by a previous FIND command.

DO "file name[volume spec]"

Transfers control to a file containing QUERY commands. At the end of the file, control transfers back to the operator.

EXIT

Terminates QUERY.

FIND item list FOR search expression

Finds entries which satisfy the search expression and places the data items listed in the item list or the entire data entry (if the set name is in the item list) into the workfile. The search expression is a numeric expression using data items as variables. The Data Variables chapter of the BASIC Programming Manual describes the process of rounding which is done with numerical expressions.

INFO

Prints a modified schema listing of the data base on the current output device.

LINEAR LIST[" string "] [item list]

Lists data items from the workfile in a linear format (one item per line) on the current output device using all BREAKS and TOTALS specified.

LIST["string "] [item list]

Lists data items from the workfile in columnar format (one data entry per line) on the current output device using all BREAKS and TOTALS specified.

OUTPUT TO device address [, width[, length]]

Changes the output device for future LIST, LINEAR LIST and INFO commands. Device address can be replaced with either "PRINTER" or "DISPLAY" to indicate output device.

PASSWORD password

Defines the data base password to be used for subsequent commands.

REPLACE item list

Replaces values for the data items specified which are in the workfile.

RUN "report name[volume spec]"

Causes a report subprogram to be run.

SORT BY item list

Sorts the entries in the workfile by the data items listed. Data items are sorted in ascending order unless a D follows the data item name.

THREAD set$_1$: set$_2$[; set$_2$: set$_3$...]

Defines the order in which data sets are accessed during a FIND command. The first data set is searched sequentially and all others are searched via the data chains and data paths. Up to 10 sets can be threaded.

WORKFILE "file name[volume spec]"

Specifies the workfile to be used for subsequent commands.

ACS (numeric expression)

This function returns the principal value of the arccosine of the numeric expression expressed in the current angular units.

ASN (numeric expression)

The ASN function returns the principal value of the arcsine of the numeric expression expressed in the current angular units.

ATN (numeric expression)

The ATN function returns the principal value of the arctangent of the numeric expression expressed in the current angular units.

COS (numeric expression)

This function returns the cosine of the angle represented by the numeric expression.

DEG

Sets degree mode for results and arguments of trigonometric functions. A degree is 1/360th of a circle.

GRAD

Sets grad mode for all results and arguments of trigonometric functions. A grad is 1/400th of a circle.

RAD

Sets radian mode for trigonometric functions. There are 2π radians in a circle.

SIN (numeric expression)

This function returns the sine of the angle which is represented by the numeric expression.

TAN (numeric expression)

This function returns the tangent of the angle which is represented by the expression.

TRACE [beginning line id [; ending line id]]

Used to trace program logic flow, in all or part of a program. Any branching causes a trace output to the system printer, which designates the branching origin and destination. When one line id is specified, tracing begins after that line is executed. An ending line id causes tracing to stop after that line is executed.

TRACE ALL

Traces all program logic flow and variable asssignments. It is like executing both TRACE and TRACE ALL VARIABLES.

TRACE ALL VARIABLES [beginning line id [; ending line id]]

Monitors value changes of all variables either in a specified program segment, or throughout the entire program. When one line id is specified, tracing begins after that line is executed. An ending line id causes tracing to stop after that line is executed.

TRACE PAUSE line id [; numeric expression]

Used as a breakpoint, causing execution to halt before a specified line is executed a certain number of times. If just the line id is specified, execution stops at that line before it is executed. The numeric expression is rounded to an integer n. Execution stops at the line before it is executed the nth time.

TRACE VARIABLES variable list

Monitors value changes of selected variables; the trace output indicates the new value of the variable and in what line the assignment occurred. The variable list can contain from one thru five variables and array identifiers separated by commas.

TRACE WAIT number of milliseconds

Used with any selective TRACE statement, or TRACE ALL, and causes the computer to wait the specified amount of time after each line which causes a trace printout. The range of the numeric expression is from −32768 thru 32767; a negative number defaults to 0.

NORMAL

Cancels all tracing operations.

AREAD$ (device address)

This function returns input data from the specified device address.

BLOCK MODE OFF# device address

Turns off block mode data transfer to a computer.

BLOCK MODE ON# device address

Turns on block mode data transfer.

ECHO OFF# device address

Turns off character echoing to the specified terminal.

ECHO ON# device address

Turns on character echoing (default mode).

OFF BREAK# device address

Cancels any corresponding ON BREAK condition.

OFF CONNECT# device address

Cancels any corresponding ON CONNECT condition.

OFF DISCONNECT# device address

Cancels any corresponding ON DISCONNECT condition.

OFF INPUT# device address

Cancels any corresponding ON INPUT condition.

```
ON BREAK# device address   ┌ GOTO line id          ┐
              [ #priority] ┤ GOSUB line id         ├
                           └ CALL subprogram name  ┘
```

Causes an interrupt when the terminal's BREAK key is pressed.

```
ON CONNECT#device address  ( GOTO line id           )
                [ ; priority] { GOSUB line id          }
                           ( CALL subprogram name  )
```

Causes an interrupt when a device is connected to the specified port.

```
ON DISCONNECT#               ( GOTO line id           )
     device address[ ; priority] { GOSUB line id          }
                              ( CALL subprogram name  )
```

Causes an interrupt when the device is disconnected from the port.

```
ON INPUT#device address      [ GOTO line id           ]
                 [ ; priority] [ GOSUB line id          ]
                              [ CALL subprogram name  ]
```

Causes an interrupt when receiving either a carriage return from a terminal, or a carriage return or ASCII DC1 from a computer. Omitting the branching statement causes the branch statement in a previous corresponding ON INPUT to be used.

```
ON OUTPUT# device address    [ GOTO line id           ]
                 [ ; priority] [ GOSUB line id          ]
                              [ CALL subprogram name  ]
```

Causes an interrupt when the sepcified device's buffer is empty. Omitting the branching statement causes the branch in a previous corresponding ON OUTPUT to be used.

```
ON TRIGGER#                  ( GOTO line id           )
     device address[ ; priority] { GOSUB line id          }
                              ( CALL subprogram name  )
```

Causes an interrupt when a computer sends an ASCII DC1 as a data transmission terminator when the control buffer is empty. This indicates that the computer is ready to accept data from the HP 250.

SEND# device address ; character code

Sends a single ASCII character to a computer. The character code can be a numeric expression.

SEND BREAK# device address

Breaks the data transfer link to a computer.

ASSIGN { file spec TO # file number
# file number TO file spec } [, return variable

[, protect code][; class list]]

Opens a file. The class list parameter can contain the keywords EBCDIC or EBCDIK when data needs to be translated from ASCII to EDCDIC or for Katakana translations, EBCDIK.

CREATE file spec , record count , record size ,
start address ; CHAR

Creates a CHAR file on an IBM media.

CREATE file spec , record count ; CHAR

Creates a CHAR file on an HP250 or interchange format media.

DELETE # file number , record number

Deletes a record from a CHAR file.

IBMDUMP record number [ON device specifier]

{ , numeric array [, return variable]
, string variable [, return variable [; conversion specifier]]
; display specifier }

Dumps a sector of an IBM disc into a string variable, into a numeric array or onto the CRT.

IBMWREC record number [ON device specifier] ,

{ string variable
numeric array } [, numeric variable [; conversion specifier]]

Dumps a sector of data to an IBM disc.

LINPUT #file number [, record number [, column position]]

; string variable

Reads data from a CHAR file into the string variable.

PRINT # file number [, record number [, word pointer]] ;

$$\left\{\begin{array}{l}\texttt{END}\\ \text{data list [, }\texttt{END}\text{]}\\ \text{print list [, }\texttt{END}\text{]}\end{array}\right\}$$

Prints data to a CHAR file. Spacing functions (TAB, SPA, LIN and PAGE) can be included.

```
RUN "RJE"
```

Checks memory size, loads and runs the RJE program.

```
AUTO RECEIVE

   ["list file"  ] [; "punch file"  ] [; WAIT n]
   [list device  ] [; punch device  ]

   [; D   ]
   [; DISP]
```

Request to receive data as soon as the communication line is open. Defaults back to receive state.

```
                               [; D   ]
AUTO RECEIVE *[; WAIT n]       [; DISP]
```

Causes Auto Receive to wait for data. The amount of time is specified in n. DISP causes all data received to be displayed on the CRT as well as being sent to the file or device specified.

```
AUTO RECEIVE OFF
```

Halts the Auto Receive state. This command should be preceded by pressing the BREAK key.

```
CLOSE
```

Closes the communication line.

```
COMMAND
```

Allows execution of any HP250 command.

```
CONTINUE
```

Continues execution of a command file.

```
                      [; *                    ]
COPY "source file"    [; "destination file"   ]
                      [; destination printer  ]

    [; PURGE][; START n][END n][; DISP]
```

Prints a file onto a list device or sends it to another file. If the PURGE parameter is specified, the source file is erased after the data is copied. The START and END parameters are used to specify portions of a file to be copied.

```
END
```

Terminates the RJE program and closes the communication line, if open.

```
INFO
```

RJE asks for the list device, then prints a table of reference information which can be used to debug line problems.

```
LOAD TAPE
```

Used to load a tape different from the default carriage control tape, CC TAPE.

```
 OPEN["config file"][; #n]

[ ( AD          )        ] [   ( u          ) ]
[;( AUTO-DIAL   ) number ] [ ; (            ) ]
[ ( MD          )        ] [   ( unattended ) ]
[ ( MANUAL-DIAL )        ] [   (            ) ]
```

Opens the line so that communications can begin. This syntax is displayed when the open softkey is pressed.

```
PROCEDURE
```

Begins execution of a command file.

```
RECEIVE ["list file" ] [; "punch file" ]
        [ list device] [; punch device ]

    [; WAIT n][; D   ]
              [; DISP]
```

Sets up the files or devices to receive data.

```
RECEIVE * [; WAIT n][; D   ]
                    [; DISP]
```

Operator specifies list and punch devices to receive data.

```
SEND [*          ] [; {NX         }]
     ["file spec"] [  {NO XPARENT }]
                   [  {X          }]
                   [  {XPARENT    }]

    [; {NC         }] [; {NT         }]
    [  {NO COMPRESS}] [  {NO TRUNCATE}]
    [  {C          }] [  {T          }]
    [  {COMPRESS   }] [  {TRUNCATE   }]

    [; D   ]
    [; DISP] [;]
```

Takes data from the specified source, one record at a time, blocks it and sends it to the host computer. The data source can be either the keyboard or a file. When the end of the data is reached, an end of transmission (EOT) is sent. If the SEND command ends in a semicolon (;), no EOT is sent. XPARENT turns on the transparent mode. COMPRESS compresses blocks of two or more space out of each record. TRUNCATE strips trailing spaces off of each record.

```
[SEND/RECEIVE] sendrecord
[SR          ]
```

The SEND/RECEIVE sends one record, e.g. request for status, then waits to receive data. Data received is directed to the CRT. The SR or SEND/RECEIVE parameter can be left out if the first character of the send record is non-alphanumeric.

```
TRACE
```

Stores all data and controller commands being sent or received in a TRACE file.

ATTACH# userid [; result]

Switches the console currently attached to the executing task to the task whose USERID is specified. The executing task must be the home task.

DETACH

Switches the console currently attached to the executing background task to its home task.

RELEASE# userid

Terminates ownership of the addressed task.

REQUEST# userid [; result]

Requests that a logical link be established between the partition and the task (userid).

TIMER/250

CLOCK

Returns the number of milliseconds that have elapsed since the HP 250 system was last loaded.

```
ON DELAY delay spec [ ; priority] { GOTO lineid              }
                                  { GOSUB lineid             }
                                  { CALL subprogram name     }
```

Schedules a software interrupt after the specified period of time. Delay spec is in milliseconds.

OFF DELAY

Cancels any software interrupt established with an ON DELAY statement.

```
FNGpl_errm$(Error)
```

Displays error messages

```
Gpl_clear(Gpl$,Crt,Hardcopy)
```

Clears the plotter display. Sets up the device for paper change if the change is not programmable. Moves the pen stable out of the way and/or puts the pen away as a convenience. Crt should always be set to zero.

```
Gpl_config (Config$,Protect$,No_
            devices,Model$(*),
            Dev_add$(*),Mess_
            add$(*), Error)
```

Interrogates the graphics device configuration file (GPL%CF is the default) and returns the device addresses and model numbers.

```
Gpl_cset (Gpl$,Char_set,Character
        _file$)
```

Selects a software character set.

```
Gpl_defldc (Gpl$, X_min, X_max,
            Y_min, Y_max)
```

Defines the logical device coordinate system.

```
Gpl_devident (Dev_add,Model$,
              Advance,Buffer,Pens,
              Error)
```

Interrogates the plotter in order to determine important parameters at the specified plotter address.

```
Gpl_devorigin(Gpl$,X_lower_left_
                   pamm,Y_lower_left_
                   pamm)
```

Places the physical plotting area (as specified by Gpl_physarea) on the plotter.

```
Gpl_draw(Gpl$,X,Y)
```

Draws a straight line from the current pen (CP) to the specified X and Y.

```
Gpl_fileis(Gpl$,Picture_file$,
           Protect$,Option,Error)
```

Used to create and open a picture file in which plotter data is stored.

```
Gpl_frame(Gpl$)
```

Draws a frame around the current window rectangle (or viewport rectangle).

```
Gpl_idraw (Gpl$,Dx,Dy)
```

Incrementally draws a straight line from the CP. The incremental units are specified by Dx and Dy.

```
Gpl_imove (Gpl$,Dx,Dy)
```

Moves the CP incrementally from one CP location to another. The pen movement occurs only when this call is followed by a "draw" or "text" call.

```
Gpl_it extwidth (Gpl$,Text$,Width)
```

Returns the width of the Text$ String.

```
Gpl_linestyle (Gpl$,linestyle)
```

Selects a line style.

```
Gpl_lorg (Gpl$,Label_origin)
```

Selects the current text formatting option.

```
Gpl_message(Gpl$,Message $,Wait )
```

Sends an operator message to the message address.

```
Gpl_move (Gpl$,X,Y)
```

Moves the CP to the location specified by X and Y.

```
Gpl_pen(Gpl$,Pen_number)
```

Selects a pen from the pen number stall.

```
Gpl_penspeed (Gpl$,Velocity)
```

Changes the pen velocity to the value specified (in cms/sec).

```
Gpl_physarea Gpl$,width_pamm,
                    height_pamm)
```

Specifies the width and height of the plot in mm.

```
Gpl_physrotate(Gpl$,Rotation)
```

Controls the rotation of the physical plotting area.

```
Gpl_physviewp (Gpl$,Xmin_pamm,Xmax_
                     pamm,Ymin,Ymax)
```

Specifies the placement of the LDC limits in the physical area coordinate system.

```
Gpl_plotteris (Gpl$, Model$, Dev_
                     Add, Mess_Add, Error)
```

Initializes the devices and parameters in the Gpl$ string actually used to plot.

```
Gpl_terminate (Gpl$,Error)
```

Causes orderly termination of plotting program.

```
Gpl_text (Gpl$,Text$)
```

Draws a line of text on the graphics display.

```
Gpl_textrotate(Gpl$,Rotation)
```

Rotates text in 90 degree increments.

```
Gpl_transmit(Gpl$,Option)
```

Clears the graphics buffer of data and sends the data to the device.

```
Gpl_tsize (Gpl$,Cell_height,Cell
               _width)
```

Varies text character height and width. Cell height is specified in LDC units. The cell width is specified as a percentage deviation from default, which is 0.5 times the cell height.

```
Gpl_useldc(Gpl$)
```

Temporarily makes the window and viewport equal to the logical device coordinates. The viewing transformation is restored when Gpl_usewc, Gpl_window, Gpl_viewport, or Gpl_defldc are called.

```
Gpl_usewc(Gpl$)
```

Restores the viewing transformation to the value before a call to Gpl_useldc.

```
Gpl_viewport (Gpl$, Xmin_ldc, Xmax
                   _ldc, Ymin_ldc, Ymax
                   _ldc)
```

Defines the viewport rectangle. Four rectangle parameters are specified in the following sequence:

```
Gpl_where (Gpl$,X,Y)
```

Stores the CP coordinates in X and Y.

```
Gpl_window (Gpl$, Xmin_wc, Xmax_wc,
                Ymin_wc, Ymax_wc)
```

Defines the window rectangle. The rectangle is defined by the four parameters in sequence. The default is the logical device coordinate space.

The following utilities are available on the HP250 Utilities Disc.

RUN "TAPFIX [volume spec]"

For use in special cases only. Used to remedy tape cartridge problems.

RUN "FVBACK [volume spec]"

Backs up the entire contents of a disc to a tape cartridge. Also restores to disc the backup file which resides on tape cartridge.

RUN "CONFIG [volume spec]"

Allows a programmer to review and change system software configuration, read/write memory assignment, default peripheral addresses and autostart. Requires the MFIG, AFIG and RFIG files, which may be run separately.

RUN "DBLOAD [volume spec]"

Loads data entries into a data base from a backup (BKUP) file created by DBUNLD. Requires the files DBLOAD, DBLOD, DBLD, LDERRC, DBFM3C, DBFM4C and DBFM5C.

RUN "DBUNLD [volume spec]"

Copies data set entries to a backup file. Requires the files DBUNLD, DBULD, UNERRC, DBFM1C and DBFM2C.

RUN "DBMODS [volume spec]"

Allows making certain changes in the data base structure without the need to unload and load data stored in the data base. Can be used to modify data base passwords, user class accesses, item names, item format numbers, set names, set capacities and data base volume names. Requires the files DBMODS, DMerrs, DMsub1 thru DMsub7 and DMfm01 thru DMfm10.

RUN "DUPL [volume spec]"

Allows you to copy the entire contents of one disc to another compatible medium.

RUN "EDITOR [volume spec]"

EDITOR is used to create and maintain data files containing lines of text. The primary purpose is to build and modify data base definitions (schemas). The EDITOR commands are described in the IMAGE/250 section of this manual. EDITOR requires the files EDITOR and EDERRS.

RUN "INIT [volume spec]"

Tests the media for defective tracks, establishes physical records and creates both main and spare file directories. INIT can also be used to purge all files on the disc.

RUN "ROUTIL [volume spec]"

Allows copying or purging a complete set of files that are a part of a single run-only program set. ROUTIL can also be used to make programs run-only, and copy or purge SYSTEM and DROM files. Requires the files ROUTIL, ROUTD, ROUTL and ROUTK.

RUN "SCHEMA [volume spec]"

Loads and runs the SCHEMA program. SCHEMA commands are listed beginning on page 51. SCHEMA requires the files SCHEMA, SCHOV2, SCHOV3 and SCHERR.

RUN "XREF [volume spec]"

Examines an HP250 type PROG file and lists where constants, line numbers, line labels, variables, functions and subprograms appear. XREF requires the PACK, IMAGE, SORT, FORMS, and REPORT WRITER DROMs to be loaded. The files XRSB1, XRFM1, REF1, and XRFILE are used in XREF.

RUN "BACKUP [volume spec]"

Stores the contents of NON-IMAGE files into one BKUP file. Purpose of this utility is to transfer data from one disc type to another.

RUN "RECOVER [volume spec]"

Recovers the contents of BACKUP files. The BACKUP file and recovered files can be on different disc types.

RUN "REPACK [volume spec]"

Repacks the files on the disc toward the front of the disc.

LOAD BIN "CATBIN[volume spec]"

Loads two statements, CATFILE and CATLINE; each returns one line of catalog information from a storage medium.

CATFILE filespec [;dset string], string variable

Returns a line of catalog information on a specified file. The 50-character line is stored in the string variable. The optional dset string is a two-character string expression evaluating to either two digits indicating a DSET file or the null string for ROOT files.

CATLINE index [ON volume spec], string variable

Returns a 50-character line of catalog information from the specified location in the file directory. The index is a positive integer numeric expression specifying the file entry location in the file directory.

LOAD BIN "DBPASS[volume spec]"

Loads the DBMAINT, DBPASS, READ DBPASSWORD and WRITE DBPASSWORD statements.

DBMAINT root file spec, old word TO new word

Allows changing the maintenance password for a specified data base.

DBPASS root file spec, user-class number,
old password TO new password

Allows changing the password for a stated user-class number.

READ DBPASSWORD root file spec ;
maintenance word ; string array variable

Reads all user passwords from the specified data base into a string array.

WRITE DBPASSWORD root file spec ;
maintenance word ; string array variable

Re-assigns all passwords in the specifid root file with those in a speciﬁed string array.

LOAD BIN "DUP [volume spec] "

Loads the DUPLICATE and DUPTEST statements.

DUPLICATE volume spec TO volume spec

Copies the contents of one disc to another under program control.

DUPTEST volume spec TO volume spec

Checks for media compatibility befcre executing DUPLICATE. An 800-series error is returned if a problem is found.

LOAD BIN "REVCHK [volume spec] "

Immediately compares the current operating system revision level with its own revision level. If both revision levels do not match, ERROR 999 occurs.

LOAD BIN "R-ONLY [volume spec] "

Loads the RUN-ONLY statement.

RUN-ONLY file spec

Converts individual programs to run-only.

LOAD BIN "XCOPY [volume spec]"

Loads the XCOPY statement.

XCOPY source file spec , file type , protect code
TO dest file spec [, REPLACE]

Creates the specified file on the specified volume and copies the file. If the destination file already exists, specify , REPLACE to copy the source file to the existing file. Data sets and root files can be copied with XCOPY.

LOAD BIN "BIT [volume spec]"

Loads the BINAND, BINCMP, BINEOR, BINIOR, BIT, ROTATE, and SHIFT statements.

BINAND (numeric expression , numeric expression)

Performs a logical AND comparison of bits in two numeric expressions which have been rounded to integers.

BINCMP (numeric expression)

Returns the binary complement of the argument.

BINEOR (numeric expression , numeric expression)

Performs an exclusive OR comparison of each bit in two integer values.

BINIOR (numeric expression , numeric expression)

Performs an inclusive OR comparison of each bit in two integer values.

BIT (numeric expression , numeric expression)

Returns the value (either 0 or 1) of the bit in the specified bit position. The first numeric expression is the integer to be evaluated. The second expression is the bit position you wish to evaluate (0–15).

ROTATE (numeric expression , numeric expression)

Returns a value obtained by rotating the first expression the number of positions specified by the second expression MOD 16. If the second expression is positive, the rotation is toward the least significant bit; if negative, the rotation is toward the most significant bit.

SHIFT (numeric expression , numeric expression)

Shifts the bits of the first numeric expression by the number of bits specified in the second expression MOD 16. If the second expression is positive, the shift is toward the least significant bit; if positive, the shift is toward the most significant bit. Bits shifted out are lost and replaced by 0's at the opposite end of the argument. Shift does not change the value of its first argument.

LOAD BIN "DATE [volume spec] "

Loads the SET DATE TO and SET TIME TO statements.

SET DATE TO string expression

Sets the system date for use by the TIMER DROM.
To set the date in US format, use the / separator (e.g. 01/23/81); for the European format, use the . separator (e.g. 23.01.81).

SET TIME TO string expression

Sets the system time in hours (h), minutes (s), and seconds (s) on a 24-hour clock. To set the time, use the hh:mm:ss format.

LOAD BIN "ACCEPT [volume spec] "

Loads the ACCEPT statement.

ACCEPT string variable

Provides a command which prevents input from being displayed.

LOAD BIN "SCAN [volume spec] "

Loads the SCAN statement.

SCAN (string expression1 , string expression2)

Searches the first string expression for the first occurrence of any single character existing in the second string expression.

System Reset Conditions

(R indicates resetting to a default condition)

	Default Setting	SCRATCH				SCRATCH	RUN	END STOP	HALT	CONT
		A	P	C	V					
Variables	none	R	R	R	3	R	R	.		.
BASIC Programs	none	R	R		.	R	.	.	.	.
Binary Programs	none	R	R			.		.	.	.
Program Execution	halted	R	R	4	4	R	.	R	R	.
Standard Printer	display									
System Printer	(device	R				.		.	.	.
Printall Printer	address 8)									
Standard Mass Storage Device	2	R		.				.		.
SFK Definitions (typing aids)	none	R						.	.	.
Subroutine Return Pointers	none	R	R	.	.	R	R	.	.	.
Angular Units	RAD	R	R		.	R	R	.	.	.
Numeric Output Format	Standard	R	R		.	R	R	.	.	.
Random Number Seed	π 180	R	R			R	R	.	.	.
Files Table	files closed	R	5	R	5	5	5	5		.
DATA Pointers	none	R	R	.		R	R			.
ERRL ERRN	0,0	R	R	R	R	R	R	.		.
ON Declaratives	none	R	R	R	R	R	R	R	.	.
DOOR LOCK	unlocked	R	R	R	R	R	R	R	R	.
TRACE Operations	none	R	R	R	R	R		.	.	.
Single Step Mode	halted		.		.	.	R	.	.	R
Device Requests	none	R	R	R	R	R	.	R	.	.

1 Setting a value at power up or after pressing CTRL HALT

2 Device used to load operating system at power up

3 Resets all variables except those declared in COM

4 Halts program only if executed while in a subprogram

5 Also caused by LOAD and GET

ASCII Character Codes

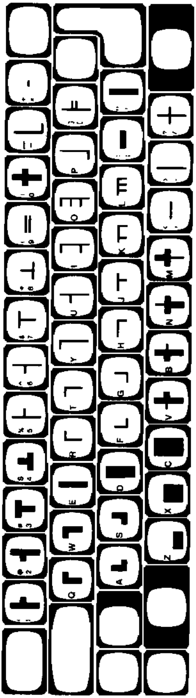
ASCII Char	EQUIVALENT FORMS	
	Binary	Dec
NULL	00000000	0
SOH	00000001	1
STX	00000010	2
ETX	00000011	3
EOT	00000100	4
ENQ	00000101	5
ACK	00000110	6
BELL	00000111	7
BS	00001000	8
HT	00001001	9
LF	00001010	10
VT	00001011	11
FF	00001100	12
CR	00001101	13
SO	00001110	14
SI	00001111	15
DLE	00010000	16
DC$_1$	00010001	17
DC$_2$	00010010	18
DC$_3$	00010011	19
DC$_4$	00010100	20
NAK	00010101	21
SYNC	00010110	22
ETB	00010111	23
CAN	00011000	24
EM	00011001	25
SUB	00011010	26
ESC	00011011	27
FS	00011100	28
GS	00011101	29
RS	00011110	30
US	00011111	31

ASCII Char.	EQUIVALENT FORMS	
	Binary	Dec
space	00100000	32
!	00100001	33
"	00100010	34
#	00100011	35
$	00100100	36
%	00100101	37
&	00100110	38
'	00100111	39
(	00101000	40
)	00101001	41
*	00101010	42
+	00101011	43
,	00101100	44
	00101101	45
	00101110	46
/	00101111	47
0	00110000	48
1	00110001	49
2	00110010	50
3	00110011	51
4	00110100	52
5	00110101	53
6	00110110	54
7	00110111	55
8	00111000	56
9	00111001	57
	00111010	58
;	00111011	59
<	00111100	60
	00111101	61
>	00111110	62
?	00111111	63

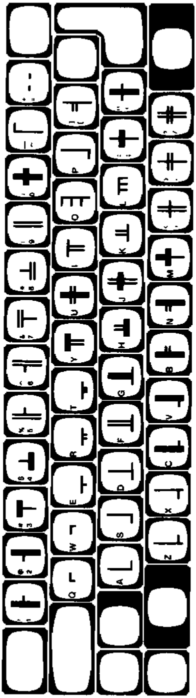
ASCII Char.	EQUIVALENT FORMS	
	Binary	Dec
@	01000000	64
A	01000001	65
B	01000010	66
C	01000011	67
D	01000100	68
E	01000101	69
F	01000110	70
G	01000111	71
H	01001000	72
I	01001001	73
J	01001010	74
K	01001011	75
L	01001100	76
M	01001101	77
N	01001110	78
O	01001111	79
P	01010000	80
Q	01010001	81
R	01010010	82
S	01010011	83
T	01010100	84
U	01010101	85
V	01010110	86
W	01010111	87
X	01011000	88
Y	01011001	89
Z	01011010	90
[	01011011	91
	01011100	92
]	01011101	93
^	01011110	94
_	01011111	95

ASCII Char.	EQUIVALENT FORMS	
	Binary	Dec
`	01100000	96
a	01100001	97
b	01100010	98
c	01100011	99
d	01100100	100
e	01100101	101
f	01100110	102
g	01100111	103
h	01101000	104
i	01101001	105
j	01101010	106
k	01101011	107
l	01101100	108
m	01101101	109
n	01101110	110
o	01101111	111
p	01110000	112
q	01110001	113
r	01110010	114
s	01110011	115
t	01110100	116
u	01110101	117
v	01110110	118
w	01110111	119
x	01111000	120
y	01111001	121
z	01111010	122
{	01111011	123
¦	01111100	124
}	01111101	125
~	01111110	126
DEL	01111111	127

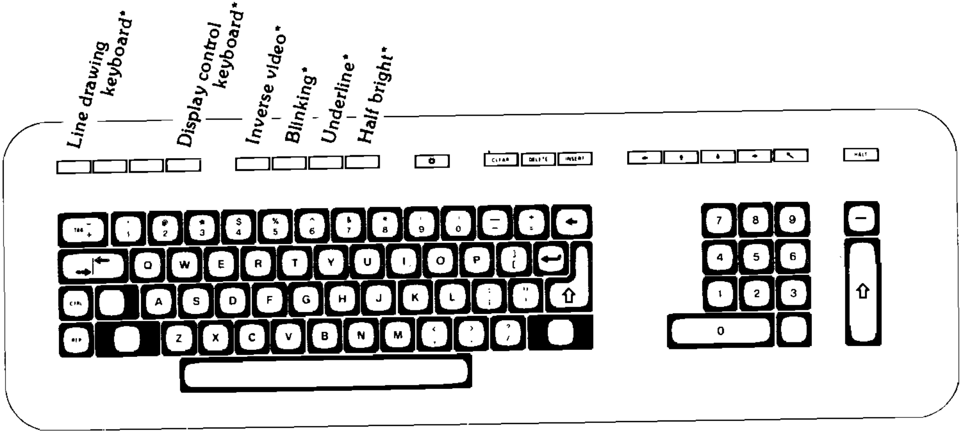
Line Drawing Keyboards



Primary Line Drawing Keyboard

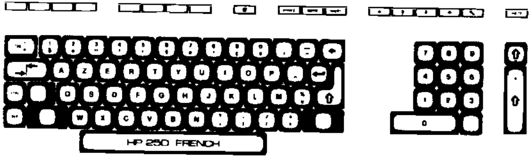


Shifted Line Drawing Keyboard

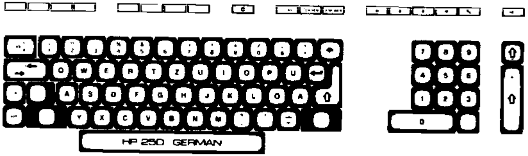


US Language Keyboard

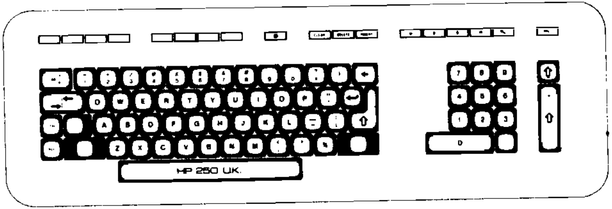
* Press CTRL ▭ to set each mode. Press SHIFT CTRL ▭ to clear each mode.



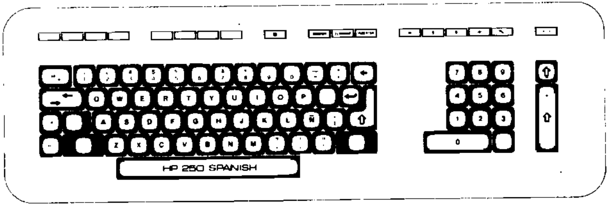
French Language Keyboard



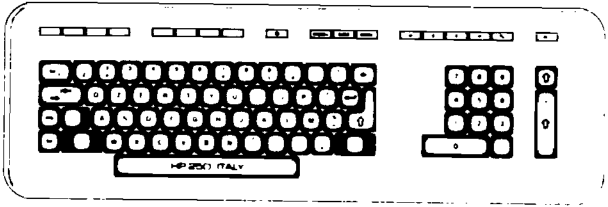
German Language Keyboard



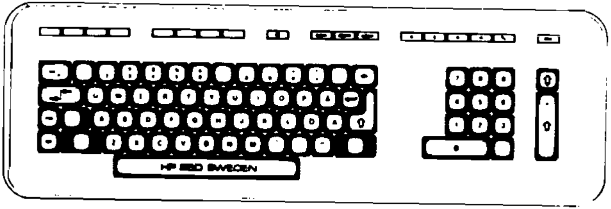
United Kingdom Language Keyboard



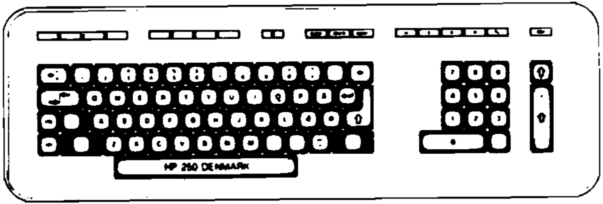
Spanish Language Keyboard



Italian Language Keyboard



Swedish Language Keyboard



Danish Language Keyboard

Error Codes

1 Software (DROM) configuration error.

2 Memory overflow.

3 Line not found or not in current program segment.

4 Improper RETURN.

5 Abnormal program termination.

6 Improperly matched.

7 Undefined function or subprogram.

8 Improper parameter matching.

9 Improper number of parameters.

10 String value required.

11 Numeric value required.

12 Attempt to re-declare a variable.

13 Array dimensions not specified.

14 Incorrect OPTION BASE statement usage.

15 Invalid bounds on array dimension, or string length in memory allocation statements.

16 Dimensions are improper or inconsistent.

17 Subscript out of range.

18 Substring out of range or substring too long.

19 Improper value.

20 Integer-precision overflow.

21 Short-precision overflow.

22 Real-precision overflow.

23 Intermediate-result overflow.

24 TAN (N*PI/2), when N is odd.

25 Argument of ASN or ACS is >1 in absolute value.

26 0 to a negative power.

27 Negative number to non-integral power.

28 Argument of LOG or LGT is negative.

29 Argument of LOG or LGT is 0.

30 Argument of SQR is negative.

31 Division by 0, or modulo 0.

32 String does not represent valid number, or string response when numeric data required.

33 Argument of NUM, CHR$, or RPT$ is improper.

34 Referenced line is not an IMAGE statement.

35 Improper image.

36 Out of data.

37 Edit string too long.

38 Syntax error in LENTER or ENTER. Also attempting to input from a protected display line..

39 Function subprogram not allowed.

40 Improper REPLACE or DELETE.

41 First line number > second line number.

42 Attempt to replace or delete a busy line or subprogram.

43 Matrix not square.

44 Illegal operand in matrix transposition or matrix multiplication.

45 Nested keyboard-entry statements.

46 No binary in (RE-)STORE BIN or no program in (RE-)SAVE or (RE-)STORE or no key in (RE-)STORE KEY.

47 Subprogram COM declaration is not consistent with main program.

48 Recursion in single-line function.

49 Line specified in ON declaration not found.

50 File number out of range from 1 thru 10.

51 File not currently assigned.

52 Improper volume label or mass storage unit specifier.

53 Improper file name.

54 Duplicate file name.

55 Directory overflow.

56 File name is undefined.

57 Attempt to use device of unknown type for mass storage.

58 Improper file type.

59 End of file found.

60 Physical or logical end of record found in direct access mode.

61 Defined record size too small for data item.

62 File is protected, or wrong protect code specified.

63 Number of records,bytes per record, or physical sectors exceeds 65534.

64 Medium overflow.

65 Incorrect data type. hue 1.00

66 Unused.

67 Parameter is ≤ 0.

68 Invalid line number encountered during MERGE, GET, or LINK.

69 -76 Unused.

77 Specified label not found.

78 Disc operation completed on device with possible volume label conflict.

79 Requested subprogram segment not present or binaries are not allowed in LOAD SUB.

80 Mass storage device door open or medium has been removed.

81 Mass storage device failure.

82 Mass storage device not present.

83 Mass storage device is write-protected.

84 Record not found.

85 Mass storage medium is not initialized.

86 Access not allowed to specified device.

87 Record address error.

88 Read data error.

89 Checkread error.

90 Mass storage system error.

91 Attempt to access a busy file.

92 Cannot get exclusive access to a specified file.

93 File opened in conflicting mode.

94 Specified file cannot currently be locked.

95 String not intact on file.

96 Program is run-only.

97 Door opened - data files closed.

98 Door opened - data lost.

99 Locked door opened.

100 Image specification expects a numeric item.

101 Image specification expects a string item.

102 Numeric field specification is larger than internal buffer size.

103 Item in PRINT USING list has no corresponding image specification.

104 Functions TAB, LIN, SPA, and PAGE are allowed in print lists for CHAR files only.

105–119 Unused.

120 Output field overflow.

121 Improper value in CURSOR statement.

122 -129 Unused.

130 Parameter for REQUEST OR RELEASE out of range.

131 Specified device not available.

132 Referenced device missing or wrong type.

133 Printer is down.

134 Printer is offline.

135 -139 Unused.

140 Spool file record length must be 256 bytes.

141 Incorrect data type found in spool file.

142 Door open - spool operation aborted.

143 Expansion of spool file would cause medium overflow.

144 Spool file size too small.

145 -149 Unused.

150 Type of expression in CASE does not match type of expression in SELECT.

151 Parameter out of range on INDENT.

152 Improper matching of structured construct.

153 No structured construct active.

155 Invalid statement specified in COMMAND.

156 More than one level of recursion not allowed in COMMAND.

160 Tape Operation Pending: the referenced tape was removed from the drive before the proper updating could take place. Insert the tape into the drive it was removed from and allow it to update properly before removal.

161 Disc Buffer Pending: the buffer required for this operation holds data for a tape that was prematurely removed. Locate the proper tape, insert it into the drive, and let the normal procedure complete before its removal.

162 Buffer Disc Not Ready: The disc holding the buffer for this tape is not ready for use.

163 Tape Door Locked.

164 Writing to tape not allowed until tape is initialized.

165 Self-test Failure on Disc.

603 Configuration file version is incompatible with currently loaded operating system.

PACK/250 Errors

200 Referenced line not a PACKFMT.

201 Unused.

202 Insufficient dimension length in PACK statement, or insufficient curren length in an UNPACK.

203 List item >32K in PACK or UNPACK.

204 Conversion error.

205 UNPACK requires a source string of greater current length.

IMAGE/250 Errors

210 Bad status array.

211 No DBASE IS statement active; improper data base specified or data base is not open.

212 Specified data set not found.

213 Too many variables in list.

214 IN DATA SET already active for data set.

215 Number of elements does not match.

216 Variable type does not match with associated field in set.

217 String length in list insufficient, or length of list array >255 bytes.

218 Variable not in common.

219 Line referenced is not an IN DATA SET LIST statement.

220 Improper or illegal use of maintenance word.

221 Data set not created.

222 Needed volume lost during dismount.

223 Improper backup file.

224 Incomplete backup file.

225 Improper utility version number in root file.

226 Corrupt data base – must recreate it.

227 Corrupt data base – must erase it in its entirety.

228 Data sets cannot be restored without a root file.

229 No volume name on data base or backup volume.

320 Set or item specifier is out of range or is an invalid set or item name.

321 Relational operator is invalid.

322 The predicate specifier is not a valid form.

SORT/250 Errors

230 Improper nesting of SORT statements, including DATA BASE IS and IN DATA SET.

231 Cannot reactivate workfile.

232 Data base mode improper for sort.

233 Required data set or root file not mounted.

234 Missing or improper set linkage.

235 No WORKFILE IS # statement active.

236 Improper data item or data item not found.

237 Sum of sort field lengths plus overhead exceeds 256 bytes in SORT BY.

238 Improper synthetic linkage.

239 Insufficient space in workfile.

240 Program lost due to disc failure.

241 Improper operation attempted on workfile.

242 Improper READ# or PRINT# on workfile.

243 Workfile contains invalid information.

244 Data Base Corrupt.

REPORT WRITER/250 Errors

250 BEGIN REPORT does not reference a REPORT HEADER statement.

251 Report Writer is already active.

252 An END REPORT DESCRIPTION statement is missing as terminator to the Report Description section.

253 Duplicate Report Writer Description section.

254 Blank lines in PAGE LENGTH statement is greater than page size, or is negative.

255 Expression in a Report Writer statement evaluates to an unacceptable value.

256 A TOTALS ON or GRAND TOTALS ON statement is improperly positioned in the Report Description section.

257 A Report Writer operation was requested while outside the program scope of an active Report Writer, or an END REPORT was not executed for an active Report Writer before subprogram termination.

258 Effective page size is less than three lines.

259 Illegal execution of a Report Description section statement.

260 Insufficient space for printed output within the current page.

261 Left margin specified is less than 1 or greater than current printer width.

262 Control variable in BREAK WHEN statement has a length greater than was initially allocated.

263 A DETAIL LINE statement may not appear within the Report Description section.

264 Level parameter is out of range of from 0 thru 9.

265 (GRAND) TOTALS ON statement is not active for the level requested.

266 Sequence parameter is out of range for (GRAND) TOTALS ON statement at the level requested.

267 WITH number LINES parameter in a header, trailer, or detail line is greater than the effective page size or is negative.

268 OLDCV($) function references a level which does not have a break defined.

269 OLDCV($) function does not match the data type for the control variable in the BREAK WHEN statement at the level requested.

270 PRINTER IS statement may not be executed while Report Writer is active.

271 A Report Writer statement may not be used recursively.

280 Language cannot be changed during SORT BY.

FORMS/250 Errors

290 Not allowed when form is active.

291 Not allowed within form image.

292 Attempt to input after last field of form.

293 Attempt to output after last field of form.

294 Not allowed unless form is active.

TIMER/250 Errors

300 Date not in acceptable format or incorrect.

301 Time not in acceptable format or incorrect.

302 Date or time has already been set. It may be set only once per system boot-up.

303 ON DELAY value is incorrect.

TIO/250 Errors

310 Port ordinal out of range of from 11 thru 15.

311 Priority value out of range from 1 thru 15.

312 Invalid address in ON...interrupt statement.

314 Ownership error: must do REQUEST before ON INPUT.

315 No input available: cannot do AREAD$ from specified port.

316 Invalid SEND or SEND BREAK statement: specified device is nct a computer.

320 -322 IMAGE Errors.

MEDIA/250 Errors

General MEDIA Errors

340 Operation only allowed on IBM media.

341 Improper operation on CHAR file.

342 Operation not allowed on this media.

343 Invalid IBM data set record length.

344 File on IBM media must be type CHAR.

345 Invalid IBM file start address in CREATE command.

346 Cartridge tape in HP interchange format cannot be accessed while in INDIRECT mode.

IBMDUMP and IBMWREC Errors

370 Record number out of range for IBM media.

371 Device does not contain IBM format media.

372 Invalid display or conversion parameter.

373 Deleted record read.

TASK/250 Errors

The error codes have different meanings for the REQUEST and ATTACH commands. The error numbers in the table are execution errors caused by unsuccessful commands with no optional result parameter. The result in the table is the returned status indicating the outcome of the command.

REQUEST# Command

Error Number	Result	Description
none	0	Ownership granted
401	1	Specified TASKID not a task
402	2	Specified TASKID not a secondary task or already owned by another user
403	3	Executing task not the home user of a workstation

ATTACH Command

Error Number	Result	Description
none	0	Attach initiated
401	1	Specified TASKID not a task
402	2	Specified TASKID not owned by executing task
403	3	Executing task not the home user of a workstation or executing task currently not attached to a workstation

RJE/250 Errors

1 Invalid file specifier.

2 Invalid printer number.

3 More parameters than expected in this command.

4 Invalid command.

5 Invalid delemeter; comma expected.

6 Same parameter assigned more than once on this command.

7 Invalid parameter.

8 Invalid channel number.

9 Invalid phone number.

10 Illegal number of records specified on repeat.

11 Command not allowed when transmitting.

12 Configuration file not for IBM 2780/3780.

13 Command not allowed when channel not opened.

14 One channel already opened; another open not allowed.

15 Space compression not allowed on 2780.

16 Unable to open file as specified.

17 Unable to create file as specified.

18 Unable to write record to specified file.

19 Unable to read record from specified file.

20 Unable to read command from command file.

21 Specified printer not available.

22 Specified printer off-line.

23 Control codes found in nontransparent record.

24 Unable to read data record from command file.

25 Received data block for wrong emulator type.

26 Record overflow due to horizontal tabbing.

CS/250 Errors

Irrecoverable Error Codes

0 Request completed successfully.

10 Invalid INP channel number.

12 COPEN called while line already open.

13 INP channel has already been reserved and opened by another user.

14 Invalid ID sequence length.

15 Invalid system buffer size.

17 Invalid phone number length.

18 Illegal character in phone number.

20 Invalid information type in COPEN information.

21 Invalid information value in COPEN information.

24 CTRACE DROM is not present.

25 No background task available for trace process or cannot find trace process.

26 Trace buffer not configured.

27 Insufficient common block space for TRACE process to run.

31 Insufficient user memory available for INP control tables and system buffers.

41 Invalid request—INP RAM Control Program is not executing or has not been initialized for this operation.

42 Invalid request—INP ROM Control Program is not executing.

44 No CS/250 System Buffer is available to complete this COPEN request.

51 INP has not been reserved and opened by a call to COPEN.

52 Undefined CS/250 function code.

57 No answer to dial attempt.

59 Auto-dial hardware failure.

61 Invalid CCONTROL code.

63 No I/O in progress for abort.

64 Abort ignored.

73 Invalid CINFO information type code.

76 All system buffers are in use; at least one concurrent I/O operation now in progress must be completed and CCOMPLETE must be called before any additional I/O operations can be initiated.

77 All Input/Output Control Blocks (IOCB's) are in use; at least one concurrent I/O operation now in progress must be completed and CCOMPLETE must be called before any additional I/O operations can be initiated.

79 No I/O in progress.

80 Invalid byte count parameter. One or more of the following conditions may have been detected:

(a) the byte count is negative,

(b) the byte count is not a positive, even-valued number for calls to CLOAD or CDUMP, or

(c) the byte count specifies more data than can be contained in a system buffer or the byte count exceeds 8190.

81 The RAM address that was specified for an INP down-line load or up-line load is invalid. One or more of the following conditions may have been detected:

(a) the RAM address is negative,

(b) for a call to CDUMP, the block to be dumped has a FWA less than zero, a LWA greater than 16383, or both, or

(c) for a call to CLOAD, the block of memory to be loaded has a FWA less than 384, a LWA greater than 16383, or both.

82 INP intemal ROM self-test failed.

83 INP intemal RAM self-test failed.

84 INP internal connector panel self-test failed.

85 INP internal timer self-test failed.

86 INP intemal output flip-flop self-test failed.

87 INP intemal SIO self-test failed.

88 INP intemal DMA self-test failed.

89 INP internal interrupt self-test failed.

90 INP intemal microprocessor self-test failed.

91 INP intemal self-test general failure.

92 CS/250 Physical Driver—Word transfer timed-out on input operation.

93 CS/250 Physical Driver—Word transfer timed-out on output operation.

94 CS/250 Physical Driver—INP ROM/RAM Control Program returned an undefined state code.

95 CS/250 Physical Driver—INP ROM/RAM Control Program performed an illegal state transition.

96 CS/250 Physical Driver—INP ROM/RAM Control Program indicated a "crash" state, was unable to return a crash code, or returned an invalid crash code.

97 CS/250 Physical Driver—No buffer was specified for an input operation.

98 CS/250 Physical Driver—No buffer was specified for an output operation.

101 Non-responding device.

102 Transfer error.

103 Data-set not ready.

104 Carrier loss.

105 Data overrun.

106 Designated INP channel does not exist.

107 INP self-test failed.

110 CS/250 Logical Driver—Invalid message type code received from RAM CP.

111 CS/250 Logical Driver—Invalid request identifier received from RAM CP.

112 CS/250 Logical Driver—Invalid request state code.

113 CS/250 Logical Driver—Invalid request state transition.

114 CS/250 Logical Driver—Invalid event for this request state.

116 CS/250 operation timed-out.

117 INP ⟷ HP250 interface state register self-test failed.

118 INP ⟷ HP250 interface interrupt self-test failed.

119 INP ⟷ HP250 interface data register self-test failed.

120 INP RAM CP—Not enough memory for a system table.

121 INP RAM CP—Routine called with zero CQE parameter.

122 INP RAM CP—Routine called with invalid parameter.

123 INP RAM CP—Request to queue a CQE that has already been queued.

124 INP RAM CP—Routine called with invalid PIN.

125 INP RAM CP—Miscellaneous system problem.

126 INP RAM CP—No free CQE.

127 INP RAM CP—Invalid request to release memory.

128 INP RAM CP—Secondary buffer problem.

129 INP RAM CP—Interrupt from an unknown source.

130 INP RAM CP—No-source interrupt.

131 INP RAM CP—Buffer address out of bounds.

132 INP RAM CP—Invalid time-out parameter.

133 INP RAM CP—Checksum of down-line load file failed.

134 INP RAM CP—Invalid address detected by hardware.

135 INP Interconnect—Illegal interrupt source error.

136 INP Interconnect—I/O completion interrupt.

137 INP Interconnect—Illegal interrupt source.

138 INP Interconnect—Illegal new-request CQE type.

139 INP Interconnect—Illegal buffer-available activation.

140 INP Interconnect—Erroneous DMA completion.

141 INP Interconnect—Illegal mainframe request type.

142 INP Interconnect—Miscellaneous error.

143 INP Trace—Illegal activation reason.

144 INP Trace—Illegal CQE type.

145 INP Protocol—Bad user request code.

146 INP Protocol—Bad system information request code.

147 INP Protocol—Bad external LCM event code.

148 INP Protocol—Undefined state transition event.

149 INP Protocol—Invalid external physical driver event.

150 INP Protocol—Unexpected SIO interrupt.

Driver-Dependent Irrecoverable Errors Resulting in Disconnection

151 INP Protocol—Invalid initialization request.

152 INP Protocol—Memory allocation failure.

151 Connect time-out.

153 Remote rejected the connection.

154 Power failure occurred.

155 Local time-out.

156 An internal error was detected by the driver.

157 Remote protocol error.

158 Remote sent shutdown sequence and disconnected.

159 Remote sent shutdown sequence and disconnected before the I/O request was issued.

Driver-Dependent Irrecoverable Errors not Resulting in Disconnection

201 Operation aborted.

202 Invalid user request.

203 Remote is not ready to accept line bid.

204 Remote rejected the line bid.

205 Remote primary station bid for the line while local user was also bidding.

206 Remote has requested to send (an RVI sequence was received).

207 Retry count exhausted.

208 Unexpected text was received.

209 Receive time-out.

210 Remote sent end-of-transmission.

211 Remote sent end-of-transmission sequence and disconnected before the I/O request was issued.

212 During the execution of a conversational CWRITE with output buffer also specified to be the input buffer, the remote requested a resend of the output buffer but its contents had been modified while receiving from the remote.

213 Remote sent an ACK sequence in response to a local CREAD acknowledgment.

214 Remote sent a NAK sequence in response to a local CREAD acknowledgment.

215 Remote send and RVI sequence in response to a local CREAD acknowledgment.

216 Remote requested a download sequence be initiated.

217 No line bid was received from the remote; local timed-out.

218 Remote sent a delay sequence instead of the expected text/response.

219 The entries in the poll list were polled the required number of times, and no station responded.

220 An EOT was received from the remote before the last block of a multiblock transmission was sent.

221 After an RVI was sent to the remote, the remote responded with text instead of the expected EOT.

222 Poll entry down or poll list down.

223 Too much data was transmitted by the remote; part of the data was lost.

Recoverable Error Codes

0 No recoverable error occurred.

1 Invalid ID sequence received.

2 Received unintelligible sequence.

3 Block check character failed check sequence error.

4 Response time-out.

5 Received incorrect acknowledgment.

6 Remote attempted to bid for the line.

7 Remote did not respond to a local's line bid.

8 Received unintelligible sequence after sending text.

9 Received enquiry character after sending text.

10 Remote requested a resend of local's last response.

11 Remote requested a resend of last text block.

12 Received end-of-transmission character while in control state.

13 Received text overflow.

14 Data overrun occurred on SIO multiplexor.

15 Transfer error occurred on SIO multiplexor.

Binary Program Errors

800 Source and destination must not be the same device.

801 Devices not compatible.

802 Destination device is too small.

803 Cannot duplicate media.

810 Protect code parameter must be 2 characters long.

850 Bad file-type specifier.

851 Files not similar, or destination file space is too small for file to be REPLACEd.

860 Old password does not match.

861 **Improper number of array elements.**

999 Binary program not compatible with current operating system revision.

System Errors

1000 System files table full.

1001 Too many accesses to specified file.

1002 Request would result in deadlock.

1003 Cannot get exclusive access to device.

1004 Keyword not recognized by this operating system revision.

1005 Memory overflow in common block.

1010 Memory parity error.

Some system malfunctions are denoted by an error-like message on the display. These messages will appear as the words "SYSTEM ERROR" followed by a letter. In addition, a table of numbers is listed. If a condition of this type occurs, you should record the message and table shown on the display. These conditions are remedied only by powering off the system and reloading. Call HP for assistance if the condition persists.

Loader Errors

LOADER ERROR messages indicate that the operating system cannot be loaded successfully –

A Checksum error.
B Disc read error.
C Checkread error.
D Insufficient memory.
E Interface error.
F Disc or system error.

Loader errors A thru C may indicate that the operating system disc is worn or damaged. Try loading the system with the backup (spare) copy of the operating system disc. If any loader error persists after repeated tries, record the error message and call HP for service.

IMAGE Status Errors

The following list describes the condition word values for IMAGE programming statements.

Condition Word	Error Description
0	Successful execution – no error.
–1	No such data base. Data base is currently opened in an incompatible mode. Bad root file reference. Data base opened exclusively.
–7	Data base lock request was already made in current environment.
–10	User may not open additional data bases, five are already opened.
–11	Bad data base name or preceding blanks missing.
–12	DBPUT, DBDELETE or DBUPDATE called with data base not locked.

−14 DBPUT, DBDELETE and DBUPDATE not allowed in access mode 8.

−21 Bad password − grants access to nothing.
Data item nonexistent or inaccessible.
Data set nonexistent or inaccessible.
Data set volume nonexistent.

−23 User lacks write access to data set.

−24 DBPUT, DBDELETE, DBUPDATE not allowed on automatic master.

−31 Bad mode.
DBGET mode 7 − illegal for detail data set.
DBGET mode 5 − specified data set lacks chains.

−52 Item specified is not an accessible search item in the specified set.
Bad LIST variable − must be "@;" or "@".

−91 Root file not compatible with current IMAGE/250 statements.

−92 Data base requires creation.

−94 Data or structure information lost. Data base must be erased or re-created.

-95 No automatic master set entry for current detail. DBDELETE only.

-96 Corrupt pointer value detected in current data set.

−120 Not enough memory to perform DBLOCK.

−122 Descriptor list bad. Not within string limits.

−123 Illegal relational operator.

−124 Descriptor too short; must be greater than or equal to 9 words.

−125 Bad set name/number.

−127 Attempt to lock using a compound item.

−128 Bad descriptor length for numeric item.

−134 Two descriptors conflict.

−135 Second lock is not allowed in modes 1, 3, 5, 11, 13 and 15.

−136 Descriptor list exceeds 2047 words.

−137 Qualifier parameter is of wrong type.

11 End-of-file.

12 Directed beginning of file.

13 Directed end of file.

15 End of chain.

16 The data set is full.

17 There is no chain for the search item value.
There is no entry with the specified key value.
No current record or the current record is empty.
The selected record is empty.

18 Broken chain.

20 Data base locked or contains locks.
Status word 3: 0 – data base locked.
1 – data set or entries locked.

22 Data set locked by another process.

23 Entries locked within set.

24 Item conflicts with current locks.

25 Entry or entries already locked.

27 Relational operator type conflict.

41 DBUPDATE will not alter a search item.

43 Duplicate key value in Master.

44 Can't delete a Master entry with non-empty Detail chains.

50 User's buffer is too small for requested data.

53 ARGUMENT field type incompatible with search field type (DBGET, mode 7, or DBFIND).
ARGUMENT's current string length is less than the string length of the search field.

80 Data set volume is not on-line.

90 Root file volume is not on-line.

94 Corrupt data base opened successfully in mode 8.

1xx There is no chain head for path xx.

3xx The automatic master for path xx is full.

4xx The master data set for path xx is not currently mounted (applies to DBPUT and DBDELETE for detail data sets).

DBLOAD/DBUNLD Errors

Error Number	Error Message
1	INCORRECT PASSWORD
2	IMPROPER SET COUNT
3	IMPROPER ITEM COUNT
4	SEARCH ITEM SUBCOUNT > 1
5	UNKNOWN SEARCH ENTRY TYPE
6	IMPROPER SEGMENT ENTRY COUNT
7	PROGRAM COMPLETION REQUIRES ROOT FILE[1]
8	NO ROOM ON CURRENT BACKUP VOLUME
9	DATA SET NAME NOT FOUND
10	DATA BASE STATUS
11	DATA BASE NOT AVAILABLE
12	BACKUP FILE VOLUMES OUT OF ORDER
13	DUPLICATE BACKUP FILE NAME[1]
14	PURGE NOT CONFIRMED; OLD FILE KEPT
15	FATAL ERROR
16	ROOT FILE NOT FOUND
17	ATTEMPT TO UNLOAD OR LOAD AUTOMATIC MASTER
18	ITEM POSITION VALUE EXCEEDS ITEM COUNT
19	IMPROPER VOLUME COUNT
20	ITEM TYPES DO NOT MATCH
21	ATTEMPT TO LOAD CORRUPT DATA BASE
22	REQUESTED DATA SET NUMBER NOT FOUND
23	ZERO LENGTH BACKUP FILE
24	IMPROPER DATA SET NUMBER
25	FORM IS NOT COMPLETE

[1] These messages are informational and warnings.

26	FILE NAME NOT FOUND
27	IMPROPER PATH NUMBER
28	IMPROPER INPUT VALUE
29	INCORRECT FILE TYPE
30	BACKUP FILE NOT CREATED BY DBUNLD UTILITY
31	ERASE REQUIRES ALL VOLUMES BE MOUNTED[1]
32	FEWER ENTRIES UNLOADED THAN EXPECTED[1]
33	FEWER ENTRIES LOADED THAN EXPECTED
34	DATA BASE IS MARKED CORRUPT[1]
35	PROGRAM FILE VERSION DISAGREEMENT
36	BACKUP SET NUMBER NOT IN DATA BASE
37	READ FAILURE IN DATA SET RECORD POSITION[1]
38	SEARCH ITEM ERROR
39	DATA ENTRY OMITTED FOR SEARCH VALUE[1]
40	VOLUME NAME TOO LONG: TRUNCATED VALUE[1]
41	FILE PROTECT CODE DOES NOT MATCH
42	MISSING DATA SET
43	DATA ITEM LENGTH OR PRECISION LOST[1]
44	ITEM CONVERSION ERROR
45	CORRUPT DATA BASE REQUIRING SERIAL MODE
46	DATA SET REQUIRES ITEM RESTRUCTURING[1]

EDITOR Errors

Error Code	Error Message
1	CLEAR NOT CONFIRMED, HOLD FILE UNCHANGED
2	CLEAR NOT CONFIRMED, WORK FILE UNCHANGED
3	FILE NOT FOUND
4	FILE NOT NUMBERED, WORK FILE IS EMPTY
5	FILE NOT NUMBERED, WORK FILE UNCHANGED

6 HOLD FILE FULL

7 ILLEGAL COMMAND

8 ILLEGAL FILE NAME

9 ILLEGAL FILE NUMBER

10 ILLEGAL SET PARAMETER

11 ILLEGAL SET PARAMETER VALUE

12 ILLEGAL VOLUME OR MASS MEMORY SPECIFIER

13 IMPROPER FILE TYPE

14 LINE ALREADY PRESENT

15 LINE NOT FOUND

16 LINE NUMBER OUT OF RANGE

17 NESTED WHILE COMMAND IS ILLEGAL

18 NO TEXT IN HOLD FILE

19 NO TEXT IN WORK FILE

20 NULL RANGE OR FIRST > SECOND

21 PURGE NOT CONFIRMED, TEXT NOT KEPT

22 SCRATCH FILE ERROR (FATAL)

23 STRING NOT FOUND WITHIN RANGE

24 SYNTAX ERROR

25 WORK FILE FULL...KEEP (NUMBERED) AND THEN TEXT

26 UNABLE TO OPEN OR READ FILE

27 UNDELIMITED FILE SPECIFIER

28 UNDELIMITED STRING

29 UNEXPECTED SYSTEM ERROR (FATAL)

30 VOLUME NOT FOUND

31 WARNING, COMMANDS FOLLOWING WHILE ARE LOST

32 WARNING, LINE TRUNCATED

HEWLETT
PACKARD

Manual Part No. 45260-90063
E0183

Printed in U.S.A.
January 1983