
HP-UX Technical BASIC Quick Reference Guide



Edition 1 November 1984

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

Contents

2	Introduction
3	Operators
4	Arithmetic and String Hierarchy
5	Variables
5	General Properties of Statements
6	Mass Storage Parameters
6	File Types
7	Syntax Conventions
8	Keywords
66	Tables
73	I/O Registers
85	Errors

Introduction

The HP-UX Technical BASIC Quick Reference Guide contains:

- A list of the operators used to construct numeric and string expressions.
- A description of the types of variables and files.
- A brief description of the BASIC keywords used to construct functions, statements, and commands.
- Tables containing a variety of information frequently needed by a BASIC programmer.

For additional information about each keyword, refer to the HP-UX Technical BASIC Reference Manual.

Operators

Numeric operators perform operations involving one or two numeric expressions. All numeric operators are dyadic (operating on two expressions) except NOT, which is monadic (operating on one expression). Both + and - may be monadic or dyadic.

String operators perform operations on string expressions.

Numeric Operators

+	Addition
-	Subtraction
*	Multiplication
/	Division
^	Exponentiation
\ or DIV	Integer division
MOD	Modulo; A MOD B evaluates to the remainder of A/B.
=	Equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
<> or #	Not equal to
AND	Boolean and. (See Truth Table)
OR	Boolean or. (See Truth Table)
EXOR	Exclusive or. (See Truth Table)
NOT	Boolean not. (See Truth Table)

Truth Table*

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

* 1 = true 0 = false

String Operators

& Concatenation
 [] Substring

Arithmetic and String Hierarchy

The order in which operations are performed within numeric and string expressions is controlled by a precedence system. Operations with the highest precedence are performed first. When two operations have the same precedence, the operations are performed from left to right.

Arithmetic Hierarchy

Highest	Parentheses; may be used to force any order of operations Functions; user-defined and BASIC Exponentiation: ^ NOT Multiplication and division: *, /, MOD, DIV or \ Addition and subtraction: +, - Relational operators: =, <, >, <=, >=, # or <>
Lowest	AND OR, EXOR

String Hierarchy

Highest	Parentheses
↓	Functions (user-defined and BASIC), substring operations
Lowest	&

Variables

A variable is one of four types:

- Simple numeric.
- Numeric array—explicitly or implicitly dimensioned.
- Simple string.
- String array—explicitly or implicitly dimensioned.

Numeric variables can have one of three precisions: REAL, SHORT, or INTEGER. (Refer to owner's documentation for accuracy and range.) Maximum string length of a simple string variable or array element is 65,530. The maximum upper bound for numeric and string arrays is 65,530.

Variable names can be from 1 to 32 characters in length. The first character must be a letter; the other characters can be letters, numbers, and the underscore character. String variable names must end with \$.

General Properties of Statements

Maximum line length: 159 characters

Line numbers: 1 through 65,535

Line labels: 32 characters maximum; letters, numbers, and underscore permitted. The first character must be a letter. The label is separated from the statement by a colon:

100 SortNames: FOR I=1 TO N*M

Multistatement lines: Statements are delimited by @. Declaratory statements must be the last statement in the line. DATA statements are not permitted in multistatement lines.

Comments: The ! delimiter separates the statement from the program comments. Comments cannot be added to DATA statements. Comments for a multistatement line must be placed at the end of the line.

Mass Storage Parameters

File name: 1 to 14 characters; slash and leading colon are not allowed. If the first character is a period, the file name does not appear in a catalog of its directory.

HP-UX path name: The path name starts at the root directory (absolute path name) or current working directory (relative path name), and traces a path to the directory in which the file is located.

Device name: Used by certain single-user systems with removable file structures. A sequence of characters assigned to each peripheral device by the operating system at power-on. Device names are listed in the /dev directory.

Volume name: Used by certain single-user systems with removable file structures. A name assigned to a disc when it is formatted. When the disc is mounted, the volume name becomes the name of the top-level directory.

File Types

The following files types can be created within BASIC:

Files Created by BASIC

Type	Contents
BASIC/PROG	BASIC programs; created by STORE
BASIC/SUBP	BASIC Subprograms; created in memory by FINDPROG; stored on disc by STORE when the first program line is a SUB statement
BASIC/GRAF	Graphics data; created by GSTORE
BASIC/DATA	numeric and string data; created by CREATE
text/data	ASCII data; not a BASIC-type file; created by SAVE

Syntax Conventions

The following typographic conventions are used in the quick reference guide to describe the BASIC language syntax.

COLOR	Characters in color are keywords and punctuation that must be typed in exactly as shown, except that lowercase letters can be substituted for uppercase letters in keywords.
<i>italics</i>	Italics indicate statement parameters. Parameters and their allowed ranges are described in the HP-UX Technical BASIC Reference Manual.
[]	Square brackets enclose optional items.
<i>stacked items</i>	Items placed on top of one another indicate that one item from the stack must be chosen.
{ }...	Braces followed by ellipsis indicate parameters that can be repeated. Items in the list are separated by commas, or, less frequently, by semicolons. Occasionally, no punctuation is necessary between repeated items. Refer to the examples or to the HP-UX Technical BASIC Reference Manual for further information.

Keywords

A

ABORTIO *interface select code*

With all interfaces, terminates any interrupt transfer in process and resets the interface control lines.

```
10 ABORTIO Isc
```

ABS *(numeric argument)*

Returns the absolute value of the argument.

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

ABSUM *(array name)*

Returns the sum of the absolute values of the elements in an array.

```
30 DISP ABSUM(Array)
```

ACS *(numeric argument)*

Returns the arccosine of the argument in quadrants 1 or 2. The argument must be in the range -1 through $+1$.

```
245 Angle=ACS(A+B)
```

ALPHA *[row] [, column]*

Displays the alpha display. The optional parameters provide for cursor movement:

- no parameters—cursor remains in current position.
- *row* only—cursor moves to specified row in current column; display scrolls if necessary.
- *column* only—cursor moves to specified column in current row.
- *row* and *column*—cursor moves to specified position; display scrolls if necessary.

```
3030 ALPHA 20
```

```
4500 ALPHA R,C
```

AMAX (array name)

Returns the largest element in the array.

```
450 T=MAX(AMAX(array1,AMAX(array2))
```

AMAXCOL

Returns the column number of the largest element in the array specified in the most recent AMAX function.

```
50 Y=AMAXCOL
```

AMAXROW

Returns the row number of the largest element in the array specified in the most recent AMAX function.

```
100 X=AMINROW
```

AMIN (array name)

Returns the smallest element in the array.

```
430 A(X,Y)=AMIN(array)
```

AMINCOL

Returns the column number of the smallest element in the array specified in the most recent AMIN function.

```
900 DISP AMINCOL
```

AMINROW

Returns the row number of the smallest element in the array specified in the most recent AMIN function.

```
910 DISP AMINROW
```

operand **AND** operand

Operator returning 1 or 0 based on the logical AND of the operands.

AREAD *string name* [{*subscript(s)*}]
[*beginning position*[, *ending position*]]

Reads alpha display characters into the specified string variable, starting at the current cursor position. The number of characters read depends on the dimensioned size of the variable or the number of character positions in the specified substring.

```
200 AREAD String$
210 AREAD B$(2)[4]
```

ASN (*numeric argument*)

Returns the arcsine of the argument.

```
250 DISP ASN(ABS(V))
```

ASSERT *interface select code*; *byte*

Sets and clears control lines of the interface. The action taken is interface-dependent.

```
340 ASSERT 7; 12
```

ASSIGN *interface select code* **TO** "*interface designator*"
 "***"

 "*file name*"
ASSIGN *file selector* **TO** "*HP-UX pathname*"
 "***"

Assigns a numeric I/O path (interface select code or file selector) to an interface or file. The interface select code is an integer in the range 3 through 10. The file selector is an integer in the range 11 through 20. An assignment to "***" closes the path.

```
50 ASSIGN 7 TO "hpiB"
60 ASSIGN 18 to "/disc1/myfile"
70 ASSIGN 910 to "*"
```

 "*file name*"
ASSIGN# *buffer number* **TO** "*HP-UX pathname*"

Opens a BASIC/DATA file by assigning a mass storage buffer to it. The file pointer is placed at the beginning of the file. The buffer number must be in the range 1 through 10.

```
600 ASSIGN #1 TO "/user1/myfile"
```

ATN (*numeric argument*)

Returns the arctangent of the argument in quadrants 1 or 4.

```
400 DISP ATN(M*N)
```

ATN2 (*argument Y, argument X*)

Returns the arctangent of Y/X in quadrants 1, 2, 3, or 4.

```
730 IF ATN2(G(2),G(1))<45 THEN 200
```

AUTO [*beginning line number* [, *increment*]]

Non-programmable. Starts automatic line numbering. Default values are AUTO 10,10.

```
AUTO 200, 5  
AUTO 300
```

AWRIT *string expression*

Displays the specified string at the current cursor position. Cursor position is not changed.

```
6090 AWRIT String$&."
```

AXES [*x-tick spacing, y-tick spacing* [, *x-intersection, y-intersection* [, *x-major count, y-major count* [, *major tick size*]]]]

Draws axes across the entire plotting area using the current units and line type. Tick marks are drawn symmetrically from the intersection of the two axes. The x and y major count specify the number of tick intervals between major ticks. The major tick size is interpreted in GUs.

```
100 AXES A,B  
230 AXES 1,2,X(I),Y(I),6,3
```

B

BEEP [*pitch*, *length*]

Produces an audible tone. In some systems, frequency and duration of the tone can be specified as functions of the optional parameters.

```
320 IF S<0 THEN BEEP
```

BINAND (*numeric argument*, *numeric argument*)

Returns the bit-by bit AND of the binary representation of the arguments.

```
40 V(I,J)=BINAND(Y,15)
```

BINCMP (*numeric argument*)

Returns the bit-by-bit binary complement of the argument.

```
70 DISP BINCMP(32+64)
```

BINEOR (*numeric argument*, *numeric argument*)

Returns the bit-by-bit exclusive OR of the binary representation of the arguments.

```
890 PRINT A(5)+BINEOR(R-S,15)
```

BINIOR (*numeric argument*, *numeric argument*)

Returns the bit-by-bit inclusive OR of the binary representation of the arguments.

```
50 PRINT# 1; BINIOR(A(I),B(I))
```

BIT (*numeric argument*, *bit position*)

Returns the value of the specified bit in the argument's binary representation.

```
250 IF(R3,5)=1 THEN 470
```

BPLOT *byte-plot string, bytes per row*

Plots raster dots onto the graphics display according to the binary representation of the characters in the byte-plot string. Plotting starts at the current pen position. At the conclusion of plotting, the pen moves to the next row, directly beneath the first dot plotted on the previous row. Bytes per row must be an even number. When the bytes per row is positive, new dots are EXORed with previous dots. When the bytes per row is negative, new dots overwrite previous dots.

```
200 BPLOT CHR$(8)&"777", 4
```

**BREAD** *string name, bytes per row*

Reads groups of eight dots from the graphics display, interprets the dots as the binary representation of characters, and enters the characters into the specified string variable. Bytes per row must be an even number. The number of characters read depends on the dimensioned size of the variable. Breathing starts at the current pen position.

```
560 BREAD block$,2
```

BTD *{string argument}*

Interprets the string argument as the binary representation of a number, and returns the decimal equivalent.

```
600 IF X=BTB(B$) THEN GOSUB 800
```

C

CALL *"subprogram name"* [*{pass paramters }*]
"HP-UX pathname"

Transfers program execution to the specified subprogram. If the subprogram is not in system memory, it is loaded from mass storage. Optional parameters can be passed by value or address:

Passed by address—variables are shared between the calling (sub)program and the called subprogram. Variable assignments made within the subprogram are returned to the calling (sub)program. Simple numeric and string variables and entire numeric and string arrays can be passed this way.

```
30 CALL "OfTheWild"(A,B$,C(.))
```

Passed by value—variable assignments made within the subprogram are not passed back to the calling (sub)program. Simple numeric and string variables, individual elements of numeric and string arrays, and numeric and string constants can be passed this way. Simple variables passed by value must be enclosed in parentheses.

```
300 CALL "/disc12/scott/readdata" ((A),(B$), C$(3)[4,5],  
"def", 45
```

CALLBIN *"entry point name"* [(*pass parameters*)]

Calls a binary entry point. The rules for passing parameters are the same as for CALL.

```
120 CALLBIN "Kmp"(B$)
```

CAT $\left[\begin{array}{l} \text{"file name"} \\ \text{"HP-UX path name"} \end{array} \right]$

Displays directory information. When the specified file is a directory, CAT displays the name, size, type, permission, and date last modified of each file in the directory. When the specified file is a BASIC file, CAT displays the file name, size, number of records and blocks, type, and date last modified. Executing CAT with no parameters displays the current working directory.

```
CAT "/mydisc/mydirectory"
```

CEIL (*numeric argument*)

Returns the smallest integer greater than or equal to the numeric argument.

```
I=CEIL(X/2)
```

CFLAG *flag number*

Clears the specified flag.

```
700 CFLAG X
```

CHAIN $\begin{array}{l} \text{"file name"} \\ \text{"HP-UX pathname"} \end{array}$

Not keyboard executable. Scratches the current program and all variables except common variables, and then loads and runs the specified program.

```
659 CHAIN "/file1/file2/file3/myprogram"
```

CHR\$ (numeric argument)

Converts a number into its corresponding character according to character set tables.

```
695 DISP "type "&CHR$(34)&"<last name> <first
name>"&CHR$(34)
```

CLEAR [device selector]

When executed without a device selector, clears the alpha display. When the device selector is an interface select code, CLEAR clears the specified interface. When the device selector specifies a device, that device is reset.

```
500 CLEAR lsc
1010 CLEAR 715
```

CLIP [x-min, x-max, y-min, y-max]

Defines the boundaries of the plotting area in the current user units. If the parameters are omitted, program execution halts until the boundaries are entered from the plotting device.

```
90 CLIP X,X+A,0,2*X
```

CNORM (array name)

Returns the column norm of an array—the largest value obtained by summing the absolute values of the elements in each column.

```
30 S=CNORM(A)
```

CNORMCOL

Returns the column number of the column having the largest sum of absolute values, using the most recent CNORM array.

```
45 J=CNORMCOL
```

COM $\left\{ \begin{array}{l} \text{string name } \{\text{upper bound(s)}\} \{\text{length}\} \\ \text{[precision] } \{\text{numeric name } \{\text{upper bound(s)}\}\} \dots \end{array} \right\} \dots$

Dimensions variables and preserves variable assignments during program chaining.

```
200 COM X,Y(16,10),Z$(300),INTEGER W(1000),I,J(12)
```

CONT [*line number*]

Non-programmable. Resumes execution of a program at the specified line after it has been paused. When the line number is omitted, execution resumes at the line where the program was paused.

CONT 4000

CONTROL *interface select code*
I/O buffer name ,

register number; {*control byte*}...

Writes the binary value of one or more control bytes to an interface or I/O buffer control register. If more than one control byte are listed, they are written into consecutive registers, starting with the specified register number.

860 CONTROL 7, 16; A(1), A(2)

CONVERT **IN** *interface select code*
OUT

PAIRS
INDEX ; *string name*]

Enables or disables a specified character conversion table to be used during ENTER and OUTPUT operations. Both IN and OUT conversions can be specified for an interface.

320 CONVERT OUT 7 INDEX ; A\$
7328 CONVERT OUT 7

COPY *"file name"*
"HP-UX path name" **TO** *"file name"*
"HP-UX path name"

Copies one or more files to the location specified after TO. When the copied file is a BASIC file, that file is copied. If the specified file is a directory, all the files in that directory are copied.

40 COPY "ThisFile" TO "/mydisc/folder1/ThisFile"
70 COPY "/disc1/directory4" TO "/disc1/directory2"

COS (*numeric argument*)

Returns the cosine of the argument.

25 PLOT X,COS(X)

COT (*numeric argument*)

Returns the cotangent of the argument.

```
730 DISP COT(Angle)
```

CREATE *"file name"*
 "HP-UX path name" ;

number of records [, record length]

Creates a BASIC/DATA file of the specified size. The default record length is 256 bytes. Minimum file size is 1024 bytes.

```
30 CREATE "thisfolder/newfile",300,30
```

CRT IS *device selector*
 file selector [, line length]

Selects the output device for DISP, LIST, and CAT output.

```
50 CRT IS 701
```

CSC (*numeric argument*)

Returns the cosecant of the argument.

```
1275 PRINT# 5;CSC(y)
```

CSIZE *space height [, aspect ratio [, slant]]*

Specifies the appearance of label characters. The height parameter designates the height of the character space in GU's. The aspect ratio is the ratio of the character space width to the character space height. The slant is the angle of the clockwise slant of the character from vertical.

```
30 CSIZE 12
500 CSIZE 12,.8,30
```

CURSCOL

Returns the current column of the alpha display cursor.

CURSOR *x-coordinate variable, y-coordinate variable*
[, *pen status variable*]

Reads the current cursor location and up/down status of the physical pen from the plotting device. Those values are assigned to numeric variables. Pen status is 0 (up) or 1 (down).

```
450 CURSOR X,Y,Z
500 CURSOR A(i),B(i)
```

CURSROW

Returns the current row location of the alpha display cursor.

D

DATA { *[“]literal[”]* }
 numeric constant } ...

Provides numeric and string constants for READ variables. The data pointer reads data from left to right within each DATA statement, from the lowest to the highest numbered statement.

```
574 DATA 10,20,"ab,cd",abcd
```

DATE

Returns a number representing the current value of the system clock date counter in YYDDD format.

DATE\$

Returns a string representing the current value of the system clock date counter in YY/MM/DD format.

DEFAULT ON **OFF**

Specifies whether warnings halt program execution (DEFAULT OFF) or provide a default value (DEFAULT ON).

DEF FN *numeric function name* [{parameter}...]
[= *numeric expression*]

DEF FN *string function name* [{parameter}...]
[= *string expression*]

DEF FN defines:

- A single-line, user-defined function and its formal parameters.
- The beginning of a multiple-line, user-defined function and its formal parameters.

The parameters can be simple numeric or string variables. Strings longer than 18 characters must be dimensioned within DEF FN (see example below).

Single-line functions must include the assignment expression (= numeric/string expression). Parameters are assigned the values of the function arguments based on the order in which they are listed. The parameters are local to the function definition.

```
7070 DEF FNcode(String$,x)=NUM(String${x,x})  
450 DEF FNsort(A,B,C$,D$[40])
```

DEG

Sets degrees as the unit in which angle parameters and results of trigonometric functions are measured.

DELETE *beginning line number* [, *ending line number*]

Deletes program lines from the current program or subprogram. Specifying only the beginning line number deletes one line.

```
DELETE 30  
DELETE 30,300
```

DET (*matrix name*)

Returns the determinant of the matrix argument.

```
204 H=DET(Array)
```

DETL

Returns the determinant of the last matrix inverted by MAT...INV, or the determinant of the coefficient matrix in the most recent MAT...SYS statement.

```
900 IF DETL(A)=B THEN 400
```

DIGITIZE *x-coordinate, y-coordinate*
 [, *pen status variable*]

Suspends program execution until the physical pen position and optional pen status are entered from the current plotting device. The coordinates are interpreted in the current units. The pen status variable is assigned the value 1 if the pen is down, 0 if the pen is up.

```
680 DIGITIZE X(i),Y(i)
700 DIGITIZE X(i),Y(i),P(i)
```

DIM $\left\{ \begin{array}{l} \text{string name } [\text{upper bound(s)}] \text{ length} \\ \text{numeric array } (\text{upper bound(s)}) \end{array} \right\} \dots$

Allocates memory for REAL numeric arrays, string variables, and string arrays.

```
500 DIM A(15,5),B$(600),C$(30)[5]
```

DIRECTORY

Displays a directory of the main program and subprograms currently in system memory. An arrow indicates the current subprogram.

DISP $\left[\begin{array}{l} \text{IMAGE line number;} \\ \text{USING IMAGE line label;} \\ \text{"format string"} \end{array} \right]$

$\left\{ \begin{array}{l} \text{numeric expression} \\ \text{string expression} \\ \text{TAB(column)} \end{array} \right\} \dots$

Displays items on the CRT IS device. Items can be separated by commas or semicolons. Simple DISP (without USING) displays data in narrow (semicolon) or wide (comma) format. DISP USING uses the format specified in the format string contained in the statement or in the referenced IMAGE statement; commas and semicolons are interchangeable. The TAB function is not allowed with DISP USING.

```
900 DISP A;TAB(20);B$;C;D;E
910 DISP USING "2(2Z.2D,4X,5A)"; A,B$,C,D$
```

dividend **DIV** *divisor*

An arithmetic operator that returns the integer portion of the quotient resulting from the division. The keyword DIV can be replaced by \.

```
230 I=A DIV B*3.5
```

DOT *{vector name, vector name}*

Returns the dot product of two vectors.

```
689 D=DOT(A,B)
```

DRAW *x-coordinate, y-coordinate*

Lowers the pen and draws a line to the specified coordinate position. The pen remains down. The x-coordinate and y-coordinate are interpreted in current units.

```
60 DRAW 40*X(1),Y(1)
```

DTB\$ *(numeric argument)*

Returns a string containing the binary representation of the decimal argument.

```
9700 OUTPUT buffer$; DTB$(I(5))
```

DTH\$ *(numeric argument)*

Returns a string containing the hexadecimal representation of the decimal argument.

```
808 OUTPUT buffer$; DTH$(I(5))
```

DTOS *(numeric argument)*

Returns a string containing the octal representation of the decimal argument.

```
688 OUTPUT buffer$; DTO$(I(5))
```

DTR *(numeric argument)*

Interprets the argument as degrees and returns the equivalent in radians.

```
476 IF Angle=DTR(70) THEN 200
```

DUMP ALPHA

Prints a copy of the alpha display.

DUMP GRAPHICS

Prints a copy of the graphics display.

E

ENABLE INTR *interface select code; enable byte*

Enables or disables the specified interface for generating an interrupt. The binary representation of the enable byte is written into control register 1.

500 ENABLE INTR 7; 8

570 ENABLE INTR ISC; BTD("10001")

ENABLE KBD *mask*

The binary equivalent of the mask is used to activate and deactivate portions of the keyboard. Setting a bit activates the key(s); clearing a bit(0) deactivates the key(s).

ENABLE KBD Mask

Bit	Mode	Keys Affected
7	program execution	RESET *
6	program execution	PAUSE *
5	program execution	special function keys
4	program execution	other keys
3	keyboard input	RESET *
2	keyboard input	PAUSE *
1	keyboard input	special function keys and other keys
0	keyboard input	special function keys and other keys
* Machine-dependent key or control sequence.		

440 ENABLE KBD;32+16

ENTER *I/O buffer*
device selector

$$\left[\begin{array}{l} \text{IMAGE line number} \\ \text{USING IMAGE line label} \\ \text{"format string"} \end{array} \right] ;$$

$$\left\{ \begin{array}{l} \text{numeric variable} \\ \text{string variable} \end{array} \right\} \dots$$

Inputs bytes of data from the specified device. The data is used to build numbers and/or strings, which are assigned to the specified variables. The numeric and string variables can be simple variables, array elements, or substrings. Simple ENTER (without USING) uses a free field format. ENTER USING uses the format specified in the format string contained in the statement or in the referenced IMAGE statement.

```
480 ENTER Buffer$; A,B$,C$(5),D$[1,300]
570 ENTER 712 USING Format; A,B,C
```

EPS

Returns machine epsilon, the smallest positive REAL number.

ERRL

Returns the line number of the program line generating the most recent error or warning.

ERRM

Displays the error message generated by the most recent error.

ERRN

Returns the error number of the most recent error or warning.

ERROM

Returns a number indicating which BASIC module returned the most recent error or warning.

ERRSC

Returns the interface select code of the interface that generated the most recent interface-dependent I/O error.

operand **EXOR** *operand*

A logical operator that returns the exclusive or of the operands.

```
665 IF A EXOR B THEN 400
```

EXP (*numeric argument*)

Returns the natural (base *e*) antilogarithm.

```
450 IF EXP(X)>1 THEN 200
```

F

FINDPROG "*subprogram name*"
 "*HP-UX path name*"

Locates the specified subprogram and makes it available for listing and editing. If the subprogram is not in system memory, it is retrieved from mass storage. If the subprogram does not exist, a new block of system memory is made available for entering a new subprogram.

```
700 FINDPROG "/disc1/folder1/mysubprogram"
```

FLAG (*flag number*)

Returns the status of the specified flag: 1 if set, 0 if clear.

```
200 F1=FLAG(I)
```

FLAG\$

Returns an eight-character string whose binary representation shows the status of the 64 flags.

FLIP

Toggles the keyboard between typewriter and BASIC modes.

FLOOR (*numeric argument*)

Returns the greatest integer less than or equal to the numeric argument (same as INT).

```
120 PRINT FLOOR(Y)
```

FN *function name* [(*parameter(s)*)]

FN *function name* = *numeric expression*
 string expression

FN END

The FN statement performs one of three operations:

- The function call invokes a user-defined function. The function type (numeric or string) must match the context of the expression invoking the function.
- FN...= is used within multiple-line functions to assign the value passed back to the invoking expression. The function type must match the expression on the right side of the equation.
- FN END defines the end of a multiple-line function.

The parameter(s) can be simple numeric or string variables, array elements, substrings, and numeric and string constants. Their values are passed to the function's formal variables.

```
120 H1=X*Y/FNtime(A,B$)
230 FNJ$=CHR$(i)&" "
```

FNORM (*array name*)

Returns the square root of the sum of the squares of each element in the array.

```
450 W=FNORM(Array)
```

FOR *loop counter*=*initial value* **TO** *final value* [**STEP** *step size*]

Defines the beginning of a FOR...NEXT loop and the loop parameters. The initial and final values specifying the initial and final values of the loop counter. The step size specifies the amount by which the loop counter is incremented each time the loop is executed.

```
35 FOR I=1 TO 10
612 FOR LoopCounter=700.5 TO 200 STEP -5.5
```

FP (*numeric argument*)

Returns the fractional part of the numeric argument.

```
444 IF FP(X)=0 THEN 200
```

FRAME

Draws a box around the current plotting area using the current line type and pen number.

FXD *x-digits* [, *y-digits*]

Specifies the number of digits (from 1 through 7) to the right of the decimal point in labels plotted by LAXES and LGRID. If only one parameter is used, both sets of labels are formatted according to x-digits.

```
80 FXD 2,0
```

G**GCLEAR** *y-coordinate*

Clears the graphics display to the current background color. If a y-coordinate is specified, only the portion from that position to the bottom of the display is cleared. The y-coordinate is interpreted in the current scaling units. On plotters equipped with automatic paper advance, GCLEAR advances the paper a full page.

```
900 GCLEAR 30
```

GET "*file name*"
"*HP-UX path name*"

Retrieves the specified text file and attempts to enter the contents of each record into memory as a program line. If a string cannot be interpreted as a program line, it is entered as a comment line.

```
450 GET "TextFile"  
333 GET "/X/Y/Zfile"
```

GLOAD "*file name*"
"*HP-UX path name*"

Retrieves the specified BASIC/GRAF file and enters its contents into graphics display memory.

890 GLOAD "GraphicsFile"
770 GLOAD "/X/Y/Graffile"

GOSUB *line number*
line label

Causes unconditional branching to the subroutine located at the specified line. The subroutine RETURN statement causes branching to the statement following the invoking GOSUB.

68 GOSUB 700
5687 GOSUB stitute



GOTO *line number*
line label

Causes unconditional branching to the specified line.

369 GOTO 700
649 GOTO sort

GRAD

Sets grads as the unit in which angle parameters and results of trigonometric functions are measured.

GRAPHICS

Displays the contents of graphics CRT memory.

GRID [*x-tick spacing*, *y-tick spacing* [, *x-intersection*,
y-intersection [, *x-grid spacing*, *y-grid spacing*
[, *minor tick size*]]]]

Draws a grid across the plotting area using the current line type and pen number. The grid lines are drawn symmetrically from the intersection of the two axes. The x- and y-tick spacing, and x- and y-intersection are interpreted in current units. The x- and y-grid spacing specify the number of tick intervals between grid lines. The minor tick size specifies the length of a minor tick in GUs.

868 GRID 5,10
576 GRID t1,t2,x(4),y(4),1,2,2

GSTORE *"file name"*
"HP-UX path name"

Stores the contents of the graphics display memory into a BASIC/GRAF file.

```
248 GSTORE "picture"
460 GSTORE "/picture/postcard"
```

H

HALT *interface select code*

Terminates any currently active interrupt transfer.

```
770 HALT 7
```

HMS (*string argument*)

Interprets the string argument as HH:MM:SS format, and returns the equivalent number of seconds.

```
684 T=HMS("08:12:00")-HMS(A$)
```

HMS\$ (*number of seconds*)

Returns a string in the form HH:MM:SS.

```
249 DISP=HMS$(seconds)
```

HTD (*string argument*)

Interprets the argument as the base 16 representation of an integer, and returns the numeric decimal equivalent.

```
880 DISP HTD("A93F")
```

I

IDRAW *x-displacement, y-displacement*

Draws a line from the current pen position to a position calculated by incrementing the current position by the specified

displacements. The displacements are interpreted in the current units.

```
990 IDRAW Increment(3),20
```

IF *numeric expression* *statement*
relational expression **THEN** *line number*
line label

ELSE *statement*
line number
line label

When the expression following IF is true or non-zero, the program uses the THEN option, branching to the line specified after THEN or executing the statement following THEN. When the expression following IF is false or zero, the program uses the ELSE option, if present.

```
884 IF X=5 THEN programend ELSE A=8E10
880 IF B-A THEN 700
```

IMAGE *format string*

Used with DISP USING, PRINT USING, LABEL USING, OUTPUT USING, and ENTER USING to specify format of input or output. The format string is a string containing one or more numeric or string field specifiers, blank specifiers, and delimiters.

```
658 IMAGE 4A,2(2X,3Z.DD)/20A
```

IMOVE *x-displacement, y-displacement*

Lifts the pen and moves it from the current pen position to a position calculated by incrementing the current position by the displacement values. The displacements are interpreted in the current units.

```
7406 IMOVE D(X1),D(Y1)
```

INF

Returns machine infinity, the largest positive REAL number.

INIT

Initializes the BASIC program in memory. Memory is allocated to all program variables, and they are assigned the values 0 and the null string.

INPUT $\left\{ \begin{array}{l} \text{numeric variable} \\ \text{string variable} \end{array} \right\} \dots$

Assigns values entered from the keyboard to program variables. Program execution halts until the input variables are satisfied.

```
749 INPUT A,B,C$
```

INT (*numeric argument*)

Returns the greatest integer less than or equal to the numeric argument. INT is identical to FLOOR.

```
7687 C=INT(A)-INT(B)
```

INTEGER {*numeric variable*[(*upper bound(s)*)]}...

Declares and allocates memory for integer variables, and dimensions integer arrays.

```
684 INTEGER A,B,C(30)
```

IOBUFFER *simple string variable*

Declares a simple string variable as an I/O buffer. Two control registers and four status registers control buffer activity:

SR0, CR0—buffer empty pointer

SR1, CR1— buffer fill pointer

SR2—active-in select code

SR3—active-out select code

```
8970 IOBUFFER buffer1$
```

IP (*numeric argument*)

Returns the integer part of the numeric argument.

```
4687 IF IP(X)#0 THEN 50
```

IPLOT *x-displacement, y-displacement* [, *pen control*]

Moves the pen from the current position to the position calculated by incrementing the current position by the displacements. The pen control parameter determines the up/down status of the pen. The displacements are interpreted in the current units.

```
3468 IPLOT X(I),Y(I),P
```

K

KEY LABEL

Displays the current key labels for the function keys.

L

LABEL $\left[\begin{array}{l} \text{USING } \textit{IMAGE line number} \\ \textit{IMAGE line label} \\ \textit{"format string"} \end{array} ; \right]$

$\left\{ \begin{array}{l} \textit{numeric expression} \\ \textit{string expression} \\ \textbf{TAB (column)} \end{array} \right\} \dots$

Plots alphanumeric labels. The label is positioned relative to the current pen position according to the LORG statement. Items can be separated by commas or semicolons. Simple LABEL (without USING) outputs label items in narrow (semicolon) or wide (comma) format. LABEL USING uses the format specified in the format string or in the referenced IMAGE statement.

```
5489 LABEL T,"seconds"
990 LABEL USING 100; t
```

LAXES [*x-tick spacing, y-tick spacing* [, *x-intersection, y-intersection* [, *x-major count, y-major count* [, *major tick size*]]]]

Draws a pair of axes and labels them at each major tick mark with the current scale units. The x-tick spacing, y-tick spacing, x-intersection, and y-intersection are interpreted in current units. The x- and y- major count specify the number of tick intervals between major tick marks. The major tick size is interpreted in GU's.

```
8860 LAXES X,Y
7423 LAXES 5,5,30,60,2,4,1
```

LBND (*array name, subscript*)

Returns the lower bound of the array subscripts (the option base).

```
6846 FOR I=LBND(A,1) TO 30
```

LDIR *angle*
run, rise

Specifies the counterclockwise rotation of labels from the horizontal axis. The angle is interpreted in the current trigonometric mode. The run and rise are interpreted in the current scale units.

```
7868 LDIR 30
4545 LDIR 1,1
```

LEN (*string argument*)

Returns the number of characters in the string. The null string has length 0.

```
891 A$=B$[LEN(B$)-1,LEN(B$)]
```

LET (*numeric variable*)... = *numeric expression*

LET (*string variable*)... = *string expression*

Assigns values to variables. The keyword LET is optional when the statement is executed programmatically, but must be included in assignments from the keyboard.

```
770 LET A(5),B=G*20
568 LET X$=A$&"*****"
```

LGRID [*x-tick spacing, y-tick spacing* [, *x-intersection, y-intersection* [, *x-grid spacing, y-grid spacing* [, *minor tick size*]]]]

Draws a grid across the plotting area using the current line type and pen number. Each grid line is labeled with the current scale units. The grid lines are drawn symmetrically from the intersection of the two axes. The x-and y-tick spacing, and x- and y-intersection are interpreted in current units. The x- and y-grid spacing specify the number of tick intervals between grid lines. The minor tick size specifies the length of a minor tick in GU's.

```
996 LGRID 5,10
666 LGRID t1,t2,x(4),y(4),1,2,2
```

LGT (*numeric argument*)

Returns the base 10 logarithm of the argument. The argument must be greater than 0.

```
80 pH=-LGT(H)
```

LIMIT [*x min, x max, y min, y max*]

Defines, in millimeters, the graphics limits of the plotting device and activates the graphics default conditions. Executing LIMIT without parameters provides for digitizing the graphics limits.

```
668 LIMIT 20,100,30,80
```

LINE TYPE *type number* [, *repeat length*]

Selects the line type for drawing lines, axes, frames, and grids. The optional repeat length, measured in GU's, can be used with peripheral plotters.

```
440 LINE TYPE 5
```

Type Number	Pattern
1	_____
2	 endpoints
3	
4	____ _
5	____ _
6	____ _
7	____ _
8	____ _

LINPUT [*"prompt string",*] *string variable*

Accepts a character string entered from the keyboard and assigns the string to the string variable. The maximum length of the string is 159 characters. The string variable can be a simple variable, array element, or substring.

```
550 LINPUT "Enter today's date",Today$
```

LIST [*beginning line number* [, *ending line number*]]

Lists portions of the current program or subprogram on the display. If no parameters are provided, the listing fills one screen.

```
8680 LIST 200  
4608 LIST 500,550
```

LOAD *"file name"*
"HP-UX path name"

Non-programmable. Retrieves the specified BASIC/PROG file and loads the program into system memory. Any previous program and subprograms are scratched.

243 LOAD "/MyFolder/Myprogram"

LOADBIN *"file name"*
"HP-UX path name"

Retrieves the specified binary file and enters it into BASIC memory, making the binary entry points available to CALLBIN.

648 LOADBIN "/disc1/binaries/ThisFile"

LOCAL {*device selector*}...

Places one or more instruments under local control after they have been under remote control. Used with HP-IB and HP-IL only.

556 LOCAL 725
231 LOCAL 4

LOCAL LOCKOUT *interface select code*

Sends LLO, which prevents an operator from placing the specified device(s) under local control. LLO is received by all devices on the specified interface. If a device is in the local state, local lockout does not take affect until the device receives a remote message and is addressed to listen.

4545 LOCAL LOCKOUT 5

LOCATE [*xmin, xmax, y min, y max*]

Defines the plotting boundaries in GU's. The LOCATE boundaries replace any previously specified plotting boundaries. Executing LOCATE without parameters provides for digitizing the boundaries.

8763 LOCATE X(1),X(10),Y(15),Y(200)

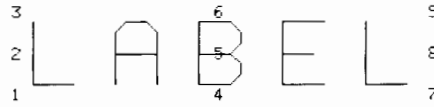
LOG (*numeric argument*)

Returns the natural (base *e*) log of the argument. The argument must be greater than 0.

9264 H=A*LOG(V)

LORG *label position*

Specifies the position of labels relative to the current pen position. Label positions 1 through 9 are available.



```
8798 LORG 5
58 LORG L
```

LWC\$ (*string argument*)

Returns a string in which all uppercase letters in the argument are replaced with lowercase letters.

```
202 DISP LWC$(S$)
```

M**MASS STORAGE IS** *"file name"*
"HP-UX path name"

Selects the current working directory. Files in the current working directory can be accessed by file name alone.

```
MASS STORAGE IS "/disc1/level2b/level3d"
```

MAT *result array* = (*numeric expression*)

Assigns the value of the numeric expression to every array element.

```
120 MAT array=(6*X(1))
```

MAT *array name* = *array name*

Copies all the elements of the operand array to the result array.

```
130 MAT array=array1
```

MAT *array name* = -*array name*

Computes the result array by changing the sign of operand array elements.

```
140 MAT array = - array1
```

MAT *array name* = *array name* *arithmetic operator* *array name*

Performs the specified arithmetic operation (+ - . /) between corresponding elements of the operand arrays, and assigns values to the result array.

```
150 MAT array = array1/array2
```

MAT *array name* = *array name* * *array name*

Performs matrix multiplication between the operand arrays, and assigns the results to the result array.

```
160 MAT array = array1*array2
```

MAT *array name* = (*numeric expression*) *scalar operator* *array name*

Performs an arithmetic operation between a numeric expression (scalar) and each element of the operand array, and assigns those values to the result array.

```
170 MAT array = (A^B) - array1
```

MAT *array name* = (*numeric expression*) * *array* + (*numeric expression*) * *array*

Multiplies each array by the specified numeric expression (scalar), adds the values, and assigns those values to the result array.

```
180 MAT B = (5)*A + (value1)*B
```

$$\mathbf{MAT} \text{ array name} \left[\left(\begin{bmatrix} \text{row} \\ \text{row}:\text{row} \end{bmatrix}, \begin{bmatrix} \text{column} \\ \text{column}:\text{column} \end{bmatrix} \right) \right] =$$

$$\text{array name} \left[\left(\begin{bmatrix} \text{row} \\ \text{row}:\text{row} \end{bmatrix}, \begin{bmatrix} \text{column} \\ \text{column}:\text{column} \end{bmatrix} \right) \right]$$

Copies specified elements of the operand array to the result array. If no row or column numbers are specified for the result array, the result array is redimensioned, if necessary, before values are copied.

```
190 MAT B=A(1:3,2:4)
200 MAT B(3,2)=D(2)
210 MAT B=B(,2)
```

MAT *array name* = **CON** [{*subscript(s)*}]

Assigns the value 1 to all array elements. The optional subscript(s) redimension the array before setting the elements equal to 1.

```
220 MAT C=CON(1,3)
```

MAT *vector name* = **CROSS** (*vector name*, *vector name*)

Computes the cross product of two, 3-element vectors.

```
230 MAT C=CROSS(A,B)
```

MAT *array name* = **CSUM** (*array name*)

Adds the columns of an array and assigns the values to a vector or 1-row matrix.

```
240 MAT vector=CSUM(array1)
```

MAT *matrix name* = **IDN** [{*subscript(s)*}]

Produces an identity matrix by assigning the value 1 to the diagonal elements (elements for which the row subscript equals the column subscript). If the matrix is not a square matrix, it must be redimensioned within the statement.

```
250 MAT B=IDN
260 MAT B=IDN(4,4)
```

MAT *matrix name* = **INV** (*matrix name*)

Calculates the inverse of a square matrix. Inversion and matrix multiplication can be specified in the same statement.

```
270 MAT C=INV(B)
280 MAT C=INV(MAT A)*MAT B
```

MAT *array name* = **RSUM** (*array name*)

Computes the sum of each row of an array and assigns those values to the elements of a vector or one-column array.

```
290 MAT array=RSUM(array1)
```

MAT *result array name* = **SYS** (*coefficient matrix name*, *constant array name*)

Solves the matrix equation $AX=B$ for the unknown array X . The coefficient matrix must be a square matrix. The result array contains the solution to the set of equations.

```
300 MAT X=SYS(A,B)
```

MAT *array name* = **TRN** (*array name*)

Produces the transpose of an array by exchanging the rows and columns of the operand array. Transposition and matrix multiplication can be specified in the same statement.

```
310 MAT array=TRN(array1)
320 MAT array2=array3*TRN(array4)
```

MAT *array name* = **ZER** [(*subscript(s)*)]

Assigns the value 0 to all elements of the array. If subscripts are specified, the array is redimensioned before the values are assigned.

```
3590 MAT C=ZER
351 MAT Q=ZER(3,4)
```

MAT **DISP** $\left\{ \begin{bmatrix} \text{ROW} \\ \text{COL} \end{bmatrix} \text{array name} \begin{bmatrix} ; \\ ; \\ / \end{bmatrix} \right\} \dots$

Displays elements of the specified array(s). The punctuation following the array name delimits the array and specifies the format—wide(,), narrow (;), or one element per line (/). COL outputs the elements column-by-column. ROW (the default) outputs items row-by-row.

```
80 MAT DISP ROW array1; COL array2/
```

MAT DISP USING *IMAGE line number;*
MAT PRINT USING *IMAGE line label ;*
"format string" ;

$\left\{ \begin{bmatrix} \text{ROW} \\ \text{COL} \end{bmatrix} \text{array name} \right\} \dots$



Outputs elements of the specified array(s) to the display or printer. The elements are output using the format specified in the format string contained in the statement or in the referenced IMAGE statement. Items are output row-by-row (ROW, the default) or column-by-column (COL).

```
880 MAT PRINT USING 300; COL array3
558 MAT DISP USING "ZZ.DD,4X"; COL array4, ROW
array5
```

MAT INPUT {array name}...

Not keyboard executable. Accepts values from the keyboard and enters them into elements of the specified array(s). Input continues until all the specified arrays are full.

```
30 MAT INPUT A,B
```

MAT PRINT
MAT PRINT USING

See **MAT DISP** and **MAT DISP USING**

MAT READ {array name}...

Not keyboard executable. Reads values from DATA statements and assigns them to array elements.

```
485 MAT READ Array1,Array2
```

MAX (numeric argument, numeric argument)

Compares the arguments and returns the larger value.

```
785 Abal=MAX (25,R(12))
```

MAXAB (array name)

Computes the absolute value of each element in the array and returns the largest value.

```
982 DISP MAXAB(array3)
```

MAXABCOL

Returns the column number of the element whose absolute value was returned by the most recently executed MAXAB function.

MAXABROW

Returns the row number of the element whose absolute value was returned by the most recently executed MAXAB function.

MDY (string argument)

Returns a number computed by converting a string argument in the form "MM/DD/YYYY" to the equivalent Julian Day number. The argument must be in the range "10/15/1582" through "11/25/4046".

```
684 Days=MDY(A$)-MDY("5/12/81")
```

MDY\$ (Julian Day number)

Returns a string formed by converting the Julian Day number to the form "MM/DD/YYYY".

MERGE *"file name"*
 "HP-UX path name" [*beginning line number*

of merged lines [, *increment of merged lines*]]

Non-programmable. Retrieves a program or subprogram from mass storage and merges it with the current program or subprogram in system memory. The retrieved (sub)program is renumbered according to the specified parameters. If the parameters are omitted, the retrieved lines are appended to the end of the current (sub)program, and the lines are renumbered in increments of 10.

```
MERGE "myfile"
MERGE "/mydisc/myfile" 800,5
```

MIN (*numeric argument, numeric argument*)

Compares the arguments and returns the smaller value.

864 H=MIN(A(1),A(2))

dividend **MOD** *divisor*

An operator returning the remainder of a division operation.

730 Y=A MOD (B*C)

MOVE *x-coordinate, y-coordinate*

Lifts the pen and moves it to the specified coordinate position. The parameters are interpreted in the current units.

396 MOVE xpoint,ypoint

MSCALE *x-offset, y-offset*

Specifies millimeter user units and defines the offset of the origin, in millimeters, from the lower left corner of the plotting area.

468 MSCALE 5,30

N**NEXT** *loop counter*

Defines the end of a FOR...NEXT loop, increments the loop counter, and tests to see if the loop counter has exceeded the final value specified in FOR. The loop counter is a simple numeric variable.

95 NEXT I

NORMAL

Cancels print-all mode and program tracing.

NOT *operand*

An operator returning 1 if the operand evaluates to 0. Otherwise, 0 is returned.

255 IF NOT A THEN 300

NPAR

Returns the number of parameters passed to a subprogram by a CALL statement.

NUM (*string argument*)

Returns the decimal value of the first character in the string argument.

```
250 FOR I=1 TO NUM(X$)
```

O**OFF CURSOR**

Turns off the alpha display cursor.

OFF EOT *interface select code*

Cancels end-of-line branching previously established by ON EOT (end-of-transfer).

```
908 OFF EOT 7
```

OFF ERROR

Cancels event-initiated branching previously established by ON ERROR. Further errors halt program execution.

OFF INTR *interface select code*

Cancels end-of-line branching for interface interrupts previously established by ON INTR.

```
672 ON INTR 6
```

OFF KEY# *key number*

Cancels end-of-line branching previously established by ON KEY#.

```
240 OFF KEY# 5
```

OFF KYBD *string expression*

Cancels end-of-line branching previously established by ON KYBD. The string expression contains characters and escape sequences representing keys for which branching is disabled.

380 OFF KYBD "ABC"&CHR\$(27)&"C"

OFF TIMEOUT *interface select code*

Cancels end-of-line branching for timeouts previously established on the specified interface by ON TIMEOUT.

900 OFF TIMEOUT 5

OFF TIMER# *timer number*

Cancels end-of-line branching previously established for the specified timer by ON TIMER#.

570 OFF TIMER# 2

ON *pointer* **GOSUB** $\left\{ \begin{array}{l} \text{line number} \\ \text{line label} \end{array} \right\} \dots$

ON *pointer* **GOTO** $\left\{ \begin{array}{l} \text{line number} \\ \text{line label} \end{array} \right\} \dots$

Not keyboard executable. Causes conditional branching to one of the specified lines based on the value of the pointer. A value of 1 causes branching to the first line listed, a value of 2 causes branching to the second, etc.

649 ON X GOSUB 100,200,300

ON CURSOR

Redisplay the alpha display cursor after it has been turned off by OFF CURSOR.

ON EOT *interface select code* **GOSUB** $\begin{array}{l} \text{line number} \\ \text{line label} \end{array}$

ON EOT *interface select code* **GOTO** $\begin{array}{l} \text{line number} \\ \text{line label} \end{array}$

Not keyboard executable. Defines an end-of-line branch to be taken when the last byte of data has been transferred by a TRANSFER statement.

```
450 ON EOT 4 GOTO 300
460 ON EOT 7 GOSUB subroutine
```

ON ERROR GOTO *line number*
line label

ON ERROR GOSUB *line number*
line label

Not keyboard executable. Defines an event-initiated branch to be taken when an error occurs during program execution.

```
ON ERROR GOSUB 400
ON ERROR GOTO readdata
```

ON INTR *interface select code* **GOTO** *line number*
line label

ON INTR *interface select code* **GOSUB** *line number*
line label

Defines an end-of-line branch to be taken when an interface generates an interrupt. The interrupt causes are specified using ENABLE INTR or CONTROL.

```
572 ON INTR 7 GOSUB 300
740 ON INTR lsc GOTO checkio
```

ON KEY# *key number* [, "*key label*"] **GOTO** *line number*
line label

ON KEY# *key number* [, "*key label*"] **GOSUB** *line number*
line label

Not keyboard executable. Establishes end-of-line branching for the user-defined keys. Maximum length of the key label is 8 characters.

```
ON KEY# 3 GOTO 200
ON KEY# 5,"Print" GOSUB printqueue
```

ON KEY# *key number*, "*key label*", "*typing aid*"

Assigns a user-defined key as a typing aid for a sequence of characters. This assignment replaces any default assignment.

```
560 ON KEY# 5, "Trace", "TRACE VAR"
```

ON KYBD *numeric variable*

[, *string expression*] **GOSUB** *line number*
line label

ON KYBD *numeric variable*

[, *string expression*] **GOTO** *line number*
line label

Not keyboard executable. Establishes an end-of-line branch to be taken when the specified keys are pressed. The keys are specified by including their characters or escape sequences in the string expression parameter. When a specified key is pressed, its numeric value is assigned to the numeric variable. When the string expression is omitted, branching remains in effect for all previously specified keys.

```
ON KYBD Keypressed "1qAz"&CHR$(27)&"w" GOSUB 200
ON KYBD Keypressed GOTO NewPage
```

ON TIMEOUT *interface select code* **GOSUB** *line number*
line label

ON TIMEOUT *interface select code* **GOTO** *line number*
line label

Not keyboard executable. Established end-of-line branching for timeouts on the specified interface. The time limit for which the system waits for completion of a handshake is set by SET TIMEOUT.

```
444 ON TIMEOUT 3 GOSUB 400
```

ON TIMER# *timer number,*

milliseconds **GOSUB** *line number*
line label

ON TIMER# *timer number, milliseconds* **GOTO** *line number*
line label

Establishes an end-of-line branch to be taken when the specified number of milliseconds has elapsed. When the statement is executed, the timer is set to 0.

```
338 ON TIMER# 3,5000 GOSUB BeginCount
```

OPTION BASE *lower bound*

Not keyboard executable. Specifies the lower bound of all arrays in a program. The lower bound must be 0 or 1. The default is option base 0.

```
200 OPTION BASE 1
```

operand OR operand

The OR operator returns a 1 or 0 based on an inclusive-or operation between the two operands.

```
640 LET Y=A OR B
```

OTD (*string argument*)

Interprets the string argument as the octal representation of a number, and returns the decimal equivalent.

```
740 DISP OTD("7"&O$)
```

OUTPUT *I/O buffer name* *device selector* **USING** *IMAGE line number* *IMAGE line label* *"format string"*

$$\left[; \left\{ \begin{array}{l} \text{numeric expression} \\ \text{string expression} \end{array} \right\} \dots \right]$$

Outputs data from the computer to the specified device or I/O buffer. Output items can be separated by commas or semicolons. Simple OUTPUT outputs items in free field (commas) or narrow (semicolons) format. OUTPUT USING uses the format specified in the format string contained in the statement or in the referenced IMAGE statement.

```
533 OUTPUT 712; A,B$,C*D
649 OUTPUT Buffer$ USING 300; A,B$
```

P**PASS CONTROL** *device selector*

Passes active controller responsibility to the specified device.

```
860 PASS CONTROL 720
```

PAUSE

Pauses program execution at the current line.

PDIR *angle*
run, rise

Specifies the rotation of coordinates, counterclockwise from horizontal, used in incremental and relative plotting (IPLOT, IMOVE, IDRAW, RPLOT). Axes and labels are not affected. The angle is interpreted according to the current trigonometric mode. The run and rise are interpreted in current units.

```
503 PDIR PI/4
820 PDIR 20,40
```

PEN *pen number*

Selects a pen on the current plotting device. When plotting to the display, the pen number is interpreted as follows:

Pen Number	Graphics Display Pen Operation
≥ 1	Plots white dots.
0	Pen is deactivated and does not plot.
-1 or < -2	Plots black dots.
-2	Performs an exclusive or, plotting white dots over black dots and black dots over white dots.

```
750 PEN -1
538 PEN PenNumber
```

PENUP

Lifts the pen on the current plotting device.

PI

Returns the value of π with full machine precision.

PLIST [*beginning line number* [, *ending line number*]]

Lists the current program or subprogram in system memory on the PRINTER IS device. If the parameters are omitted, the entire program is listed. If only the beginning line number is specified, all lines from that line to the end of the program are listed.

```
PLIST 50
PLIST 700,900
```

PLOT *x-coordinate*, *y-coordinate* [, *pen control*]

Moves the pen from the current position to the specified x-, y-coordinate position. The optional pen control parameter specifies the up/down status of the pen:

Pen Control Parameter	Pen Action
positive, even	pen moved and then lifted
positive, odd (default)	pen moved and then lowered
negative, even	pen lifted and then moved
negative, odd	pen lowered and then moved

```
800 PLOT x,y,z
810 PLOT 4,4*y
```

PLOTTER IS *device selector*

Selects the device to receive graphics output, reads the device's graphics limits, and activates the default graphics conditions. The default device is the graphics display.

```
20 PLOTTER IS Dselect
```

POS (*string searched*, *substring searched for*)

Returns the position of the first character of a substring in another string.

```
96 IF POS(A$,"tyf")<5 THEN B$=ROTATE$(A$,1)
```

PPOLL *interface select code*

Returns the response byte of a parallel poll operation (HP-IB, HP-IL only).

```
80 IF PPOLL(7)=10 THEN GOTO 400
```

PRINT $\left[\begin{array}{l} \text{IMAGE line number;} \\ \text{USING IMAGE line label;} \\ \text{"format string"}; \end{array} \right]$

$\left\{ \begin{array}{l} \text{numeric expression} \\ \text{string expression} \\ \text{TAB(column)} \end{array} \right\} \dots$

Outputs items to the PRINTER IS device. Items can be separated by commas or semicolons. Simple PRINT (without USING) displays data in narrow (semicolon) or wide (comma) format. PRINT USING uses the format specified in the format string contained in the statement or in the referenced IMAGE statement; commas and semicolons are interchangeable. The TAB function is not allowed with PRINT USING.

```
486 PRINT "Balance due:";TAB(30);B$
420 PRINT USING "3D.DD,2(10X,7A);A,A$,B,B$"
```

PRINT# *buffer number*; $\left\{ \begin{array}{l} \text{numeric expression} \\ \text{string expression} \\ \text{array name ()} \end{array} \right\} \dots$

Prints the PRINT# items to a data file serially. The file must be opened (by ASSIGN#) before data can be printed to it. The buffer is assigned to the file when it is opened. At the conclusion of the PRINT# operation, the file pointer remains positioned after the last printed item.

```
500 PRINT# 1; A,B$
```

PRINT# *buffer number, record number*;

$\left[\left\{ \begin{array}{l} \text{numeric expression} \\ \text{string expression} \\ \text{array name ()} \end{array} \right\} \dots \right]$

Prints the PRINT# items to a data file using random access. The file must be opened (by ASSIGN#) before data can be printed to it. The buffer is assigned to the file when it is opened. The PRINT# items are printed to the specified record. When the PRINT# items are omitted, the file pointer moves to the beginning of the specified record.

```
490 PRINT# 3,12;A,B$,C*3,"aaaaa"
```

PRINT ALL

Directs a copy of all displayed alphanumeric output to the PRINTER IS device.

PRINTER IS $\begin{matrix} \text{file selector} \\ \text{device selector} \end{matrix} \left[, \text{line length} \right]$

Selects the device or file to receives output from PRINT (USING) and PLIST. The line length specifies the number of characters sent between end-of-line sequences.

```
570 PRINTER IS 701
573 PRINTER IS Pdevice, 60
937 PRINTER IS 15
```

PURGE $\begin{matrix} \text{"file name"} \\ \text{"HP-UX path name"} \end{matrix}$

Deletes the entry for the specified BASIC file from its directory. The space occupied by the file becomes available.

```
779 PURGE "thisfile"
340 PURGE "/user1/data1/thisfile"
```

R

RAD

Establishes radians as the units in which all angle parameters and results of trigonometric functions are measured.

RANDOMIZE [seed]

Specifies a new seed for the RND function.

```
2860 RANDOMIZE NewSeed
```

RATIO

Returns the ratio of the dimensions of the graphics limits—horizontal dimension divided by vertical dimension.

READ $\left\{ \begin{matrix} \text{numeric variable} \\ \text{string variable} \end{matrix} \right\} \dots$

Not keyboard executable. Reads numeric and/or string constants from one or more DATA statements and assigns those values to the specified variables.

250 READ Number,Characters\$

READ# *buffer number*; $\left\{ \begin{array}{l} \text{simple variable} \\ \text{array variable ()} \end{array} \right\} \dots$

Retrieves items serially from a data file and assigns those values to the specified variables. The file must be opened (by ASSIGN#) before data can be read from it. The buffer is assigned to the file when it is opened. At the conclusion of the READ# operation, the file pointer remains positioned after the last printed item.

4721 READ# 1; A,B\$,C()

READ# *buffer number, record number*

$\left[; \left\{ \begin{array}{l} \text{simple variable} \\ \text{array variable ()} \end{array} \right\} \dots \right]$

Retrieves items from the specified record of a data file. The file must be opened (by ASSIGN#) before data can be read from it. The buffer is assigned to the file when it is opened. When the READ# items are omitted, the file pointer moves to the beginning of the specified record.

4729 READ# 3,12;A,B\$,C()

READTIM *timer number*

Returns the number of seconds elapsed on the specified timer (1, 2, or 3). The timers are reset to zero by ON TIMER#.

6740 READTIM 2

REAL {*numeric name* [{*upper bound(s)*}] }...

Declares and reserves memory for full precision numeric variables and arrays.

50 REAL A, B(2,5)

REDIM {array name (upper bound(s))}...

Changes the upper bound(s) of a previously dimensioned array, and reassigns the array elements to new positions. Array elements are stored in order from left to right along each row, from top row to bottom row.

```
2180 REDIM A(25,5), B(100)
```

REM *literal*

Characters following REM are regarded as program comments. The comment delimiter, !, can also be used.

```
200 REM Load the required binaries
400 CALL "/user2/curveplot" ! Load the curveplot
subprogram
```

REMOTE {device selector}...

With HP-IB or HP-IL, places the specified device(s) into remote control. With BCD, sends a partial field separator. If the device selector is an HP-IB interface select code, devices do not go into remote state until they are addressed to listen.

```
4732 REMOTE 7
305 REMOTE 705,710
```

REN [new initial line number [, new increment value [, original initial line number [, original ending line number]]]]

Non-programmable. Renumbers all or part of a program. The portion to be renumbered is delimited by the original initial and ending line numbers (default is the entire program).

```
532 REN 1,1
840 REN 1000,5,200,400
```

RENAME "file name" "HP-UX path name" TO "file name"

Renames the specified file by changing the name of the file in its directory. Subprogram files cannot be renamed this way.

```
70 RENAME "/writers/anna/baref" TO "baqrg"
```

REPLACEVAR *simple variable name* **BY** *simple variable name*

REPLACEVAR *array variable name* **BY** *array variable name*

Non-programmable. Replaces all occurrences of the specified variable name with another name. The comma within parentheses must be included for two-dimensional arrays.

REPLACEVAR A by B
REPLACEVAR array(,) by a(,)

REQUEST *interface select code; response byte*

Used by the non-active controller to send a response byte to the active controller (HP-IB, HP-IL, and serial only).

60 REQUEST 7; 64

RESET *interface select code*

Performs a hardware reset of the interface.

RESET lsc

RESTORE $\left[\begin{array}{l} \text{line number} \\ \text{line label} \end{array} \right]$

Not keyboard executable. Moves the data pointer to the beginning of the specified DATA statement (default is the first DATA statement in the program).

80 RESTORE 600
400 RESTORE data1

RESUME *interface select code*

Re-enables I/O operations after they have been disabled by HALT or SEND (HP-IB, HP-IL, serial).

580 RESUME 7

RETURN

Causes branching from the current subroutine to the statement following the invoking GOSUB.

REV\$ (string argument)

Returns a string formed by reversing the sequence of characters in the string argument.

```
740 B$=REV$(A$&"asdfghjkl")
```

RMD (dividend,divisor)

Returns the remainder of the division. (MOD and RMD return the same result when the dividend and divisor have the same sign.)

```
470 IF RMD(d,10)<5 THEN 200
```

RND

Returns a pseudorandom number in the range $0 \leq N < 1$.

RNORM (array name)

Returns the row norm of an array, computed by summing the absolute values of the elements of each row and selecting the largest value.

```
246 Array(X,X)=RNORM (B)
```

RNORMROW

Returns the row number from which the most recent FNORM value was returned.

ROTATES\$ (string argument, shift factor)

Returns a string in which the characters in the argument have been shifted right (positive shift factor) or left (negative shift factor) by the specified number of positions.

```
842 Dtt$=ROTATES$(String$,x)
```

RPLOT x-coordinate, y-coordinate [, pen control]

Moves the pen from the current pen position to the specified x-, y-coordinate position, using a local coordinate origin. The local origin is changed by any statement, except RPLOT, that alters the pen position. Refer to PLOT for interpretation of the pen control parameter.

```
123 RPLOT COS(angle),SIN(angle)
```

RPTS (*string argument, repeat factor*)

Returns a string consisting of the string argument repeated the specified number of times.

```
592 row1$=RPTS("1234567890",8)
```

RTD (*numeric argument*)

Interprets the argument as radians and returns the equivalent in degrees.

```
890 PLOT angle,SIN(RTD(angle))
```

RUN [*line number*]

Non-programmable. Performs prerun initialization of the main BASIC program and starts program execution. If no line number is specified, execution starts at the lowest-numbered line.

```
RUN 500
```

S

SAVE *"file name"*
"HP-UX path name" [, *beginning line number*
 [, *ending line number*]]

Copies the current program as ASCII character strings to the specified text file. If necessary, the file is created. The beginning and ending line numbers specify the portion of the program to be saved. The default values are the first and last program lines.

```
423 SAVE "/MAY/vouchers"  

9875 SAVE "/users/JohnB/MAY/vouchers", 400, 1000
```

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

Scales the current plotting area defined by the graphics limits (GU's mode) or plotting boundaries (UU's mode).

```
484 SCALE 0,100,40,50  

299 SCALE Xlower,Xlower*2,Ylower,Ylower+.005
```

REVS (string argument)

Returns a string formed by reversing the sequence of characters in the string argument.

```
740 RA = REVS A
```

"literal"

SCAN *simple variable* [, *line number*]
array variable ([,])

Non-programmable. Searches the current program or subprogram for the specified variable or literal, displaying all lines on which a match is found.

```
SCAN "DISP USING 300"
SCAN Array()
```

SCRATCH

Erases the current BASIC program, all subprograms, and variables. Also cancels all mass storage and I/O buffer assignments.

SCRATCHBIN *"file name"*

Erases the specified binary program from BASIC memory.

```
3607 SCRATCHBIN "CplotD"
```

SCRATCHSUB *"subprogram name"* [TO END]

Erases the specified subprogram(s) from BASIC memory. If TO END is included, all subprograms from the specified subprogram to the end of the directory are scratched.

```
472 SCRATCHSUB "Kbase"
5478 SCRATCHSUB "Jst" TO END
```

SEC (numeric argument)

Returns the secant of the argument.

```
25 X(4)=SEC(angle(4))
```

"file name"

SECURE *"HP-UX path name"*

, *"security code"* , *security type*

Secures BASIC files against being listed, copied, or overwritten, according to the security type:

File Security

Security Type	File Type(s)	Protection Provided
0	PROG, SUBP	Prevents LIST, PLIST, and editing.
1	PROG, SUBP	Prevents LIST, PLIST, editing, and file-to-file COPY. The file is ignored during directory-to-directory COPY.
2	PROG, SUBP, DATA, GRAF	Prevents the file from being overwritten by STORE, GSTORE, or PRINT#.

The first two characters of the security code becomes associated with the file for types 0 and 1 security only.

97 SECURE "/Agroup/Ausers/plan1","Gg",1

SEND interface select code;

CMD {numeric/string expression}...
DATA {numeric/string expression}...[EOL]
TALK primary address
LISTEN {primary address}...
SCG {secondary address}...
UNL
UNT
MLA
MTA



Sends the specified commands or data to one or more devices:

- HP-IB, HP-IL: All secondary keywords send the specified command. All commands except DATA require that the computer be active controller.
- Serial: Only the DATA command can be sent.
- BCD: CMD, LISTEN, and TALK—primary addresses 0 through 6 set a partial field specifier. DATA sends the lower bits of data bytes.
- GPIO: CMD, LISTEN, TALK—primary addresses 0 through 15 send the port configuration. DATA sends bytes of data.

65 SEND 7; CMD A\$,32 SCG 12 DATA A1,A2
350 SEND 5; UNL

REV\$ (*string argument*)

Returns a string formed by reversing the sequence of characters in the string argument.

740 B\$ = RFV\$/ΔC9 " " " " " "

SETGU

Sets the system to graphics units (GU's) mode. Plotting is in graphics units, and the plotting boundaries equal the graphics limits.

SET I/O interface select code, register number, data byte

Writes a byte of data to the specified interface register.

95 SET I/O 7,16,1

SET TIMEOUT *interface select code; milliseconds*

Sets the maximum amount of time the system will wait for the specified interface to complete a handshake during an I/O operation.

384 SET TIMEOUT lsc; 5000

SETUU

Sets the system to user units (UU's) mode. Plotting is in user units, and the plotting boundaries set by LOCATE or CLIP, previously deactivated by SETGU, are reactivated.

SFLAG *flag number*
 string expression

Sets one flag (specified by the flag number) at a time, or sets and clears all 64 flags at once. The 64 set/clear status of the flags are represented by 8, 8-bit characters, with the first character representing flags 1 through 8.

537 SFLAG 4

359 SFLAG "~0~0~0~0~0~0~0~0"

SGN (*numeric argument*)

Returns 1 if the argument is positive, -1 if the argument is negative, 0 if the argument is 0.

```
427 Tvt=SGN(Pd)*Wrk
```

SHORT {*numeric name* [(upper bound(s))]}...

Declares and reserves memory for short precision numeric variables and arrays.

759 SHORT S1,S2,Dividends(500)

SHOW *x-min, x-max, y-min, y-max*

Specifies plotting user units such that one unit of x equals one unit of y.

264 SHOW -10,10,-8,8

SIN (*numeric argument*)

Returns the sine of the argument.

768 PLOT Angle,SIN(Angle)

SINGLESTEP

Non-programmable. Executes the current program line and then halts execution.

SPOLL *device selector*

Returns an integer representing the status byte of the specified device. The computer must be active controller in order to perform a serial poll.

942 DeviceStatus(address)=SPOLL(700+address)

SQR (*numeric argument*)

Returns the positive square root of the argument.

IF SQR(velocity)<3 THEN GOSUB 600

STATUS *interface select code, register number; {numeric name}...*

Reads the contents of one or more interface or I/O buffer status registers. If more than one numeric name is specified, registers are read in consecutive order starting with the specified register number.

312 STATUS 7,16; R16, R17

STORE *"file name"*
"HP-UX path name"

Non-programmable. Stores the current BASIC program or subprogram.

STORE "Richeys"

SUB "subprogram name" [{variable name [(,)]}]...

Defines the beginning of a subprogram and lists the formal parameters passed into the subprogram. Array variables are designated by a pair of parentheses following the array name.

```
400 SUB "marine"  
600 SUB "fileread" (Records,Precision$,D())
```

SUBEND

Returns program execution to the calling program or subprogram.

SUBEXIT

Same as **SUBEND**.

SUM (array name)

Returns the sum of all the elements in an array.

```
612 ArraySum=SUM(Array)
```

T**TAB** (column)

Used with simple PRINT and DISP (without USING) to position output in the specified column. TAB should be used only with narrow format (items separated by semicolons).

```
4798 DISP X;TAB(10);Y
```

TAN (numeric argument)

Returns the tangent of the argument.

```
437 PLOT angle,TAN(angle)/angle
```

TIME

Returns the current value of the system clock seconds counter.

TIMES

Returns the system clock reading in HH:MM:SS notation.

TRACE [ALL]

TRACE VAR $\left\{ \begin{array}{l} \text{simple variable} \\ \text{array variable } ([,]) \end{array} \right\} \dots$

Traces program variable assignments and/or program branching. Simple TRACE traces branching. TRACE VAR traces changes in the values assigned to the specified simple variables and arrays. The program must be allocated before TRACE VAR is executed. TRACE ALL traces all program execution and changes in values assigned to all program variables. To trace within a subprogram, the trace statement must be executed within the subprogram.

```
30 TRACE VAR A,B$,C()
```

TRANSFER *device selector* **TO** *I/O buffer*

INTR $\left[; \left\{ \begin{array}{l} \text{DELIM byte} \\ \text{COUNT \# of bytes} \\ \text{EOI} \end{array} \right\} \dots \right]$

TRANSFER *device selector* **TO** *I/O buffer*

FHS $\left[; \left\{ \begin{array}{l} \text{COUNT \# of bytes} \\ \text{EOI} \end{array} \right\} \dots \right]$

Transfers bytes of data from the specified device to an I/O buffer. The transfer ends when the buffer is full or when any of the terminating conditions is met.

```
3528 TRANSFER 712 TO Buffer1$ INTR
80 TRANSFER 9 TO H$ FHS; COUNT 1200 EOI
```

TRANSFER *I/O buffer* **TO** *device selector* **INTR**
FHS

Transfers bytes of data from an I/O buffer to the specified device.

```
2349 TRANSFER buffer$ TO 720 INTR
```

SUB "subprogram name" [{variable name [(,)]}...]

Defines the beginning of a subprogram and lists the formal parameters passed into the subprogram. Array variables are designated by a pair of parentheses following the variable name.

TRIGGER {device selector}...

Sends the Group Execute Trigger message to the specified device(s).

472 TRIGGER 706,708

TRIM\$ (string argument)

Returns a string stripped of all leading and trailing spaces.

80 DISP TRIM\$(Name\$)

TYP (buffer number)

Returns the data type of the next item in a BASIC/DATA file.

TYP Function Values

TYP Value	Data Type
1	Numeric
2	Complete String
3	End-of-file
5 through 7	Not used
8	Beginning of string
9	Middle of string
10	End of string

645 IF TYP(5)=2 THEN READ# 5; D\$

U

UBND (array name, subscript)

Returns the upper bound of the first or second subscript of the specified array.

47 IF I=UBND(array) THEN 600

UNCLIP

Cancels plotting boundaries set by CLIP or LOCATE, setting the plotting boundaries equal to the graphics limits. Current scaling units are retained.

UNSECURE "file name"
"HP-UX path name"

, "security code", security type

Non-programmable. Cancels file security previously specified by SECURE.

UNSECURE "/writers/max/pisces/c1","hk",2

UPCS (string argument)

Returns a string in which all the lowercase letters in the argument are converted to uppercase.

612 CV\$=UPC\$(string\$)&"ZZZ"

V



VAL (string argument)

Returns a numeric value based on the digits contained within a string. The number begins with the first digit, decimal point, or sign.

983 PRINT# 3; VAL(A\$)

VAL\$ (numeric argument)

Returns the string representation of the argument in standard number format.

614 S\$=S\$&VAL\$(Acnt1)

VOLUME "old volume name"
":device name" **IS** new "volume name"

Implementation is machine dependent. Changes the name of a disc and remounts the disc under the new volume name. Volume names can be a maximum of six characters long.

4 VOLUME "/A-LIST" IS "B-LIST"

W

WAIT (*milliseconds*)

Delays program execution for the specified number of milliseconds.

```
400 WAIT T
```

WHERE *x-coordinate, y-coordinate* [, *pen status variable*]

Assigns the current location and up/down status of the plotting device's logical pen to numeric variables. Pen status is 0 (up) or 1 (down).

```
410 WHERE x1,y1
420 WHERE X(I),Y(I),P(I)
```

X

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

Draws a horizontal axis at the specified y-intercept with optional tick marks. The axis extends across the specified interval, or across the entire plotting area.

```
281 XAXIS 30, 2
135 XAXIS Yint,2,0,X(30)
```

XREF L

Displays a table containing a cross-reference of program line numbers, line labels, and user-defined functions.

XREF V

Displays a table of all variables and user-defined functions, including the line numbers of lines referencing those variables and functions.

Y

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

Draws a vertical axis at the specified x-intercept with optional tick marks. The axis extends across the specified interval, or across the entire plotting area.

```
592 YAXIS 30, 2
```

```
840 YAXIS Xint,2,0,X(30)
```

Tables

US ASCII Character Set

ASCII Char.	Equivalent Forms		ASCII Char.	Equivalent Forms	
	Dec	Binary		Dec	Binary
NUL	0	00000000	space	32	00100000
SOH	1	00000001	!	33	00100001
STX	2	00000010	"	34	00100010
ETX	3	00000011	#	35	00100011
EOT	4	00000100	\$	36	00100100
ENQ	5	00000101	%	37	00100101
ACK	6	00000110	&	38	00100110
BEL	7	00000111	'	39	00100111
BS	8	00001000	<	40	00101000
HT	9	00001001	>	41	00101001
LF	10	00001010	*	42	00101010
VT	11	00001011	+	43	00101011
FF	12	00001100	,	44	00101100
CR	13	00001101	-	45	00101101
SO	14	00001110	.	46	00101110
SI	15	00001111	/	47	00101111
DLE	16	00010000	0	48	00110000
DC1	17	00010001	1	49	00110001
DC2	18	00010010	2	50	00110010
DC3	19	00010011	3	51	00110011
DC4	20	00010100	4	52	00110100
NAK	21	00010101	5	53	00110101
SYNC	22	00010110	6	54	00110110
ETB	23	00010111	7	55	00110111
CAN	24	00011000	8	56	00111000
EM	25	00011001	9	57	00111001
SUB	26	00011010	:	58	00111010
ESC	27	00011011	;	59	00111011
FS	28	00011100	<	60	00111100
GS	29	00011101	=	61	00111101
RS	30	00011110	>	62	00111110
US	31	00011111	?	63	00111111

US ASCII Character Set (continued)

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

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

Roman Extension Character Set

ASCII Char.	Equivalent Forms		ASCII Char.	Equivalent Forms	
	Dec	Binary		Dec	Binary
	128	10000000	ı	167	10100111
	129	10000001	˘	168	10101000
	130	10000010	˙	169	10101001
	131	10000011	˚	170	10101010
	132	10000100	˛	171	10101011
	133	10000101	˜	172	10101100
	134	10000110	ˆ	173	10101101
	135	10000111	˚	174	10101110
	136	10001000	˚	175	10101111
	137	10001001	˚	176	10110000
	138	10001010		177	10110001
	139	10001011		178	10110010
	140	10001100	°	179	10110011
	141	10001101	˚	180	10110100
	142	10001110	˚	181	10110101
	143	10001111	˚	182	10110110
	144	10010000	˚	183	10110111
	145	10010001	!	184	10111000
	146	10010010	˚	185	10111001
	147	10010011	˚	186	10111010
	148	10010100	˚	187	10111011
	149	10010101	˚	188	10111100
	150	10010110	˚	189	10111101
	151	10010111	˚	190	10111110
	152	10110000	˚	191	10111111
	153	10011001	˚	192	11000000
	154	10011010	˚	193	11000001
	155	10011011	˚	194	11000010
	156	10011100	˚	195	11000011
	157	10011101	˚	196	11000100
	158	10011110	˚	197	11000101
	159	10011111	˚	198	11000110
space	160	10100000	˚	199	11000111
À	161	10100001	˚	200	11001000
Á	162	10100010	˚	201	11001001
Â	163	10100011	˚	202	11001010
Ã	164	10100100	˚	203	11001011
Ä	165	10100101	˚	204	11001100
Å	166	10100110	˚	205	11001101

Roman Extension Character Set (continued)

ASCII Char.	Equivalent Forms		ASCII Char.	Equivalent Forms	
	Dec	Binary		Dec	Binary
ö	206	11001110	ó	231	11100111
û	207	11001111	ô	232	11101000
ä	208	11010000	õ	233	11101001
í	209	11010001	ö	234	11101010
ð	210	11010010	÷	235	11101011
ñ	211	11010011		236	11101100
â	212	11010100	ù	237	11101101
ï	213	11010101	ÿ	238	11101110
ø	214	11010110	ü	239	11101111
æ	215	11010111	ÿ	240	11110000
å	216	11011000		241	11110001
ì	217	11011001		242	11110010
ò	218	11011010		243	11110011
ó	219	11011011		244	11110100
é	220	11011100		245	11110101
í	221	11011101	-	246	11110110
ê	222	11011110	+	247	11110111
ë	223	11011111	±	248	11111000
ä	224	11100000	±	249	11111001
å	225	11100001	±	250	11111010
â	226	11100010	±	251	11111011
ð	227	11100011	■	252	11111100
í	228	11100100	±	253	11111101
ì	229	11100101	±	254	11111110
ï	230	11100110		255	11111111

Boundaries and Scaling

Condition or Statement	Parameter Units	Effect on Mode GU's vs. UU's	Effect on Scaling Units	Effect on Graphics Limits	Effect on Plotting Boundaries
Power-on or Reset	—	In UU's	UU's = GU's Shortest dimension = 100 GU's	Set to default graphics limits of the graphics display	Set to default graphics limits of the graphics display
PLOTTER IS	—	Set to UU's mode	UU's = GU's	Read from device	Set to graphics limits
LIMIT	mm	Set to UU's mode	UU's = GU's	Set according to LIMIT parameters	Set to graphics limits
LOCATE	GU's	No effect	No effect	No effect	Set according to LOCATE parameters
CLIP	Current units	No effect	No effect	No effect	Set according to CLIP parameters
UNCLIP	—	No effect	No effect	No effect	Set to current graphics limits

Boundaries and Scaling (Continued)

Condition or Statement	Parameter Units	Effect on Mode GU's vs. UU's	Effect on Scaling Units	Effect on Graphics Limits	Effect on Plotting Boundaries
SCALE	UU's	Set to UU's mode	Set according to SCALE parameters	No effect	No effect
SHOW	UU's	Set to UU's mode	Set in equal x,y units according to SHOW parameters	No effect	No effect
MSCALE	mm	Set to UU's mode	Set to mm units according to MSCALE parameters	No effect	No effect
SETGU	—	Set to GU's mode	GU's ,	No effect	Temporarily set to graphics limits
SETUU	—	Set to UU's mode	UU's	No effect	Restores plotting boundaries

Pen Control With PLOT, IPLOT, and RPLLOT

Pen Control Parameter	Pen Action
Positive, even	Pen moved and then lifted
Positive, odd	Pen moved and then lowered
Negative, even	Pen lifted and then moved
Negative, odd	Pen lowered and then moved

Graphics Display Pen Numbers

Pen Number	Graphics Display Operation
≥ 1	Plots white dots.
0	Pen is deactivated and does not plot.
-1 or < -2	Plots black dots.
-2	Performs an exclusive or, plotting white dots over black dots and black dots over white dots.

Reflecting Plots With LIMIT, LOCATE, SCALE, and SHOW

Order of Statement Parameters	Effect
x-max, x-min, y-min, y-max	Reflects output across y-axis
x-min, x-max, y-max, y-min	Reflects output across x-axis
x-max, x-min, y-max, y-min	Reflects output across origin

I/O Registers

Buffer Registers

Register	Default Value	Function
SR0	1	Buffer empty pointer
SR1	0	Buffer fill pointer
SR2	0	Active-in select code
SR3	0	Active-out select code
CR0	1	Buffer empty pointer
CR1	0	Buffer fill pointer

HP-IB Registers

Register: Function	Bit							
	7	6	5	4	3	2	1	0
SR1: ID	0	0	0	0	0	0	0	1
CR1: Parity	—	—	—	—	Odd	Even	Always 1	Always 0
SR1: Interrupt Cause	IFC	LA	CA	TA	SRQ	DCL SDC	GET	SCG
CR1: Interrupt Mask	IFC	LA	CA	TA	SRQ	DCL SDC	GET	SCG
SR2: Control Lines	—	REN	SRQ	ATN	EOI	DAV	NDAC	NRFD
CR2: Control Lines	—	REN	SRQ	ATN	EOI	DAV	NDAC	NRFD
SR3: Data Lines	DIO8	DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1

CR3: Data Lines	DIO8	DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1
SR4: Address/System Controller	—	—	System Controller	HP-IB Address				
SR5: State	SC	LA	CA	TA	SPE	Parity Error	REN	LLO
SR6: Secondary Command	—	—	—	Secondary Command				
CR16: EOL Sequence Enable	EOL	—	—	—	—	Number of Characters in EOL Sequence		
CR17—CR23: EOL Characters	Decimal Value of EOL Character							

Serial Registers

Register: Function	Bit							
	7	6	5	4	3	2	1	0
SR0: ID	0	0	0	0	0	0	1	0
S/C R1: Interrupt Mask	Break Received	Framing Error	Parity Error	Received Data Available	DCD* RTS†‡	Auto Disconnect	DSR* RTS†	CTS* DTR†‡
S/C R2: Modem Control Signals	—	—	—	—	—	DRS* No Connection	RTS* DCD†‡	DTR* CTS/DSR† CTS†
SR3: Modem Status	—	—	—	—	DCD* RTS†‡	—	DSR* DTR† —†	CTS* DTR†‡
S/C R4: Line Characteristics	—	Set Break	Force Parity	Even Parity	Enable Parity	Stop Bits	Character Length	
S/C R5: Modem Features	—	—	Received Handshake	Transmit Handshake	DCD* RTS†‡	—	DSR* DTR† —†	CTS* DTR†‡

S/C R6: Baud Rate Divisor	(115,200/ baud rate) DIV 256									
Status/Control R7: Baud Rate Divisor	(115,200/ baud rate) MOD 256									
S/C R8: Replacement Character	Parity and Framing Error Replacement Character									
S/C R9: Transmitter/Receiver Control	Enable Transmitter	Strip Received Rubouts	Strip Received Nulls	Change Character if Error	Set Bit 7 of Character if Error	Reset Receive Queue	Auto-Echo Enable	Enable Receiver		
Status R10: Line Status	—	—	Transmit Register Empty	Break Received	Framing Error	Parity Error	—	Received Data Available		
SR11: I/O Termination Cause	End of Output Data List	End of Input Data List	COUNT Satisfied	CR15 Character Received	CR14 Character Received	CR13 Character Received	CR12 Character Received	DELIM Character Received		

Serial Registers (Continued)

Register: Function	Bit							
	7	6	5	4	3	2	1	0
CR11: Input Data Control	Enable Transmit Flag	Disable Transmit Flag	—	Terminate if CR15	Terminate if CR14	Terminate if CR13	Terminate if CR12	—
CR12—CR15: Termination Characters	Input Termination Characters							
CR16: EOL Sequence	Auto RTS Enable	EOL Transmit Disable	Number of Characters in EOL Sequence					
CR17—23 EOL Characters	Decimal Value of EOL Character							
* Modem Cable F/M. † Instrumentation Cable F/F. ‡ Printer Cable F/F.								

BCD Registers

Register: Function	Bit									
	7	6	5	4	3	2	1	0		
SR0: ID	0	0	0	0	0	0	1	1		0
SR1: Interrupt Cause	Generated from Function B (Most Significant Digit)				Generated from Function A (Most Significant Digit)					
CR1: Interrupt Mask	Function B (Most Significant Digit)				Function A (Most Significant Digit)					
SR2: Control Line Messages	I/OA	I/OB	CTLAB	CTLB	FLGA	FLGB	--	--		--
CR2: Handshake Lines & Port 10	I/OA	I/OB	CTLA	CTLB	Port 10 Output (when available)					
S/C R3: Mantissa Digits	Number of Digits for Channel B Mantissa (0-11)				Number of Digits for Channel A Mantissa (0-11)					
S/R R4: Exponent Digits	Number of Digits for Channel B Exponent (0-3)				Number of Digits for Channel A Exponent (0-3)					

BCD Registers (Continued)

Register: Function	Bit									
	7	6	5	4	3	2	1	0		
S/R R5: Function Digits	Number of Digits for Channel B Function (0–11)									
S/R 6: Decimal Point Placement	Number of Mantissa Digits to the Right of the Decimal Point (Channel B)									
S/C R7: Control Sense & Hand- shake Mode	Logic Sense I/OA	Logic Sense I/OB	Logic Sense CTLA	Logic Sense CTLB	Logic Sense FLGA	Logic Sense FLGB	Handshake Mode A	Handshake Mode B		
S/C R8: Mantissa & Exponent Digit Sense	Logic Sense for Channel B Mantissa and Exponent Digits									
S/C R9: Function Digit Sense	Logic Sense for Channel B Function Digits									
S/R R10: Hand- shake Lines & Port 10	Exponent B Logic Sense	Mantissa B Logic Sense	Exponent A Logic Sense	Mantissa A Logic Sense	Logic Sense for Port 10 Data					

GPIO Registers

Register: Function	Bit							
	7	6	5	4	3	2	1	0
SR0: ID	0	0	0	0	0	1	0	0
CR0: Parity Control	—	—	—	—	Odd	Even	Always 1	Always 0
SR1: Interrupt Cause	FLGD	FLGC	FLGB	FLGA	—	—	Parity Error	—
CR1: Interrupt Mask	FLGD	FLGC	FLGB	FLGA	—	—	Parity Error	—
SR2: Line Status	FLGD	FLGC	FLGB	FLGA	CTLD	CTLB	CTLC	CTLA
CR2: Assertion Control	RESB	RESA	—	—	CTLD	CTLB	CTLC	CTLA
S/R R3: Handshake Line Normalization	Invert FLGD	Invert FLGC	Invert FLGB	Invert FLGA	Invert CTLD	Invert CTLB	Invert CTLC	Invert CTLA

GPIO Registers (Continued)

Register: Function	Bit							
	7	6	5	4	3	2	1	0
S/C R4: Data Normalization & Handshake	Handshake Mode		Flag Transition	Always 1	Invert Port D	Invert Port C	Invert Port B	Invert Port A
S/C R5: Primary Address & Trigger Action	R7 > DATA	R7 = DATA	R7 < DATA	Enable Auto Response	Primary Address			
S/C R6: Strobe Control	Resolution Increment	Strobe Pulse Duration						
S/C R7: Trigger Character	Decimal Value of Trigger Character							
S/C R8: Output Enable	—	—	—	—	—	—	Output Enable B	Output Enable A
S/C R9: Output Inhibit	—	—	—	—	—	—	—	Enable Output Inhibit
CR16: EOL	—	—	—	—	—	Number of Characters in EOL Sequence		
R17—23 EOL Character	Decimal Value of EOL Character							

HP-IL Registers

Register: Function	Bit							
	7	6	5	4	3	2	1	0
SR4: Loop Address	AAD	—	—	A4	A3	A2	A1	A0
SR0: ID	0	0	0	0	0	1	0	1
SR1: Interrupt Cause	IFC	LA	CA	TA	SRQ	CLR	GET	DDC
CR1: Interrupt Mask	IFC	LA	CA	TA	SRQ	CLR	GET	DDC
SR2: Last Control Bits	—	—	—	—	—	C2	C1	C0
CR2: Next Control Bits	—	—	—	—	—	C2	C1	C0
SR3: Last Loop Byte	D7	D6	D5	D4	D3	D2	D1	D0
CR3: Next Data Byte	D7	D6	D5	D4	D3	D2	D1	D0

HP-IL Registers (Continued)

Register: Function	Bit							
	7	6	5	4	3	2	1	0
CR4: Loop Address	AAD	—	—	A4	A3	A2	A1	A0
SR5: Interface Status	Tx	LA	CA	TA	SRQ	EAR	REN	LLO
CR5: Interface Status	—	—	—	—	—	—	—	EAR
SR6: Device-Dependent Commands	—	—	T/L	C4	C3	C2	C1	C0
SR7: Responding Devices	—	—	—	A4	A3	A2	A1	A0
CR16: EOL Sequence	End Frame Enable	—	—	—	—	EOL2	EOL1	EOL0
CR17—23: EOL Characters	Decimal Value of EOL Character							

Errors

The following tables list prerun and runtime errors and warnings. With DEFAULT OFF, warnings 1 through 8 return an error message and halt program execution.

Warnings

Warning Number	Message	Causes
1		Not used.
2	OVERFLOW	Overflow; returns maximum value for specified precision. ■ REAL, SHORT, or INTEGER value is out of range. ■ Division by 0.
3	COT/CSC=INF	COT or CSC of $n \times 180^\circ$; returns INF
4	TAN/SEC=INF	TAN or SEC of $n \times 90^\circ$; returns INF
5	0~NEG	Zero raised to a negative power; returns INF
6	0~0	Zero raised to the zero power; returns 1
7	NULL DATA	Executing AXAMCOL or AMINCOL for a vector; returns 0.
8		Not used.

Errors

Warning Number	Message	Causes
9	NEG*NON-INT	Negative value raised to a non-integer power.
10	SQR(-)	Square root of a negative number.
11	ARG OUT OF RANGE	Argument or parameter out of range.
12	LOG(0)	Logarithm of zero.
13	LOG(-)	Logarithm of a negative number.
14		Not used.
15	SYSTEM	System error.
16	CONTINUE BEFORE RUN	Program not allocated.
17 and 18		Not used.
19	MEM OVF	Memory overflow.
20 and 21		Not used.
22	SECURED	Attempting to violate system file security.
23 and 24		Not used.
25	BAD BIN LOAD	LOADBIN operation has failed: ■ The specified file does not exist. ■ The specified file is not formatted properly.

Errors (Continued)

Warning Number	Message	Causes
26 through 29		
30	OPTION BASE	Not used.
31		OPTION BASE parameter is not 0 or 1, or OPTION BASE statement is at inappropriate place in the program.
32	COM MISMATCH	Not used. Common variable mismatch.
33	DATA TYPE	Data type mismatch for READ or READ#.
34	NO DATA	Insufficient data for DATA statement.
35	DIM EXIST VRBL	Attempting to dimension a variable that has previously been explicitly or implicitly dimensioned.
36		Not used.
37	DUP FN	Duplicate user-defined function name.
38	NO FN END	A second DEF FN statement has been executed before the first function was ended with FN END.
39	FN MISSING	Referencing a non-existent user-defined function, or branching to the middle of a function.
40	FN PARAM	Illegal function parameter, function parameter mismatch.
41		Not used.
42	RECURSIVE FN CALL	Recursive user-defined function.

43	NUMERIC INPUT	Numeric input is required.
44	TOO FEW INPUTS	Too few inputs for INPUT or MAT INPUT.
45	TOO MANY INPUTS	More items were given than were requested by INPUT.
46	NEXT MISSING	FOR with no matching NEXT.
47	NO MATCHING FOR	NEXT with no matching FOR.
48 and 49		Not used.
50	BIN PROG MISSING	Binary program could not be found in memory.
51	RETURN W/O GOSUB	Attempt to execute RETURN before GOSUB.
52	IMAGE	Illegal IMAGE format string.
53	PRINT USING TAB	Illegal PRINT USING image specifier.
54		Illegal TAB argument.
55	SUBSCRIPT	Array subscript out of dimensioned range.
56	STRING OVF	String overflow; a string is too large for the length of a string variable.
57	MISSING LINE	Referencing a nonexistent line.

Errors (Continued)

Warning Number	Message	Causes
58 and 59		Not used.
60	WRITE PROTECT	The medium is write-protected, or the file is secured against overwriting.
61 and 62		Not used.
63	DUP NAME	Duplicate path name for RENAME, CREATE, or COPY.
64 and 65		Not used.
66	FILE CLOSED	Attempting to access (by READ# or PRINT#) or close a closed file.
67	FILE NAME	Incorrect file name or path name.
68	FILE TYPE	Attempting a mass storage operation with a wrong file type.
69	RANDOM OVF	Overflow during random file access.
70	READ	Failure by MERGE or FINDPROG to access the mass storage medium.
71	EOF	Attempting to PRINT#/READ# beyond the end of the file.
72	RECORD	Attempting to READ#/PRINT# to a nonexistent record.
73 through 87		Not used.
88	BAD STATEMENT	SUB statement must be first line of subprogram.
89	INVALID PARAM	Invalid parameter; parameter out of range.

90			Not used.
91	MISSING PARAM		Missing parameter.
92 through 100			Not used.

Error numbers over 100 often have more than one message associated with them. In cases where no error message is displayed (for example, in an ON ERROR routine), the errors can be differentiated by the ERROR module number.

Error Number	Message	ERROR Number	Causes
101		192	Warning; program paused with an active TRANSFER.
102 through 108	XFR		Not used.
109	DIGITIZE	1	Digitizing for DIGITIZE, LOCATE, LIMIT, or CLIP has been interrupted from the keyboard and aborted.
109	# DIMS	176	Incorrect number of dimensions in an array.
109	PRGM TYPE	232	Attempting to CALL a non-subprogram file.
110	NOT A 3- VECTOR	176	The specified vector does not have 3 elements.
111	DIM	176	Incorrect number of array elements.
	MISMATCH		
111	I/O OPER	192	The I/O operation is invalid for the specified interface.
111	RECURSIVE	232	A subprogram attempts to CALL or SCRATCHSUB itself.

Errors (Continued)

Error Number	Message	ERROR Number	Causes
112	DETERMINE IS 0	176	Determinate of a matrix is 0.
113	DIM SIZE	176	Illegal dimension size.
113	INTERFACE- DEPENDENT	192	Interface dependent error: ■ HP-IB: interface must be system controller. ■ Serial: UART receiver overrun. ■ BCD: port 10 is not currently available. ■ GPIO: odd number of bytes was transferred in the 16-bit word configuration. ■ HP-IL: take control message was ignored by the device.
113	PARAM MISMATCH	232	Mismatch between CALL and SUB parameters.
114	NOT SQUARE	176	Array is not square.
114	INTERFACE- DEPENDENT	192	Interface-dependent error: ■ HP-IB: HP-IL: interface must be active controller. ■ Serial: receiver buffer overrun. ■ BCD: port 10 not currently available. ■ GPIO: FHS transfer was aborted.
115	NON-VECTOR	176	Array is not a vector.
115	INTERFACE- DEPENDENT	192	Interface-dependent error: ■ HP-IB: HP-IL: interface must be addressed to talk. ■ Serial: automatic disconnect forced. ■ BCD: FHS transfer aborted. ■ GPIO: configuration does not allow output enable or output operation on port A or port B.

115	SUB STMT MSG	232	SUB statement is missing in called subprogram.
116	INTERFACE- DEPENDENT	192	Interface-dependent error: ■ HP-IB, HP-IL: interface must be addressed to listen. ■ BCD: data direction mismatch has occurred. ■ GPIO: CTL line is not in the proper state.
117	INTERFACE- DEPENDENT	192	Interface-dependent error: ■ HP-IB, HP-IL: interface must be non-controller. ■ BCD: command was directed to nonexistent field.
118	INTERFACE- DEPENDENT	192	Interface-dependent error: ■ BCD: CTL is not in the proper state to start the operation. ■ HP-IL: a transmission error or protocol violation occurred.
119	INTERFACE- DEPENDENT	192	Interface-dependent error: ■ BCD: data format does not match mode of the interface. ■ HP-IL: addressed talker ignored the start of transmission message.
120	NO M.S. DEVICE	208	No mass storage device is currently active.
121 through 123			Not used.
124	ISC	192	Illegal interface select code.
125	ADDR	192	Improper primary address.
125	VOLUME	208	The specified volume name (top-level directory) was not found.

Errors (Continued)

Error Number	Message	ERROR Number	Causes
126	BUFFER	192	I/O buffer problem: ■ Attempting to OUTPUT or TRANSFER data to a full buffer. ■ Attempting to ENTER data from an empty buffer. ■ The specified string variable is not a declared I/O buffer.
126	PLOTTER IS	1	The designated plotter does not respond.
126	MSUS	208	The specified device name was not found.
127	NUMBER	192	Incoming character sequence is not a valid number.
127	READ VFY	208	A read verify error has occurred.
128	EARLY TERM	192	Buffer was emptied before all enter fields were satisfied.
128	FULL	208	The directory or mass storage medium is full.
129	VAR TYPE	192	An ENTER variable does not match the image specified for that variable.
130	NO TERM	192	Required terminator was not received during ENTER.
130	DISC	208	Disc error.
131	TIMEOUT	192	An I/O timeout has occurred.