

Hewlett-Packard System 45 Desktop Computer

Reference Guide



Reference Guide



HP System 45 Desktop Computer



Hewlett-Packard Calculator Products Division
P.O. Box 301, Loveland, Colorado 80537
(For World-wide Sales and Service Offices see back of manual.)
Copyright by Hewlett-Packard Company 1977

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

Table of Contents

General Information	1
ROMs	2
Special Keys	3
Language Terms	6
Operators	6
BASIC Syntax Guidelines	8
Syntax Listing	13
Mainframe	13
Mass Storage ROM	54
Graphics ROM	55
Error Messages	60
Mass Storage ROM	63
Graphics ROM	64

General Information

The programming information in this guide serves as a handy reference for users of the HP 9800 Series System 45 Desktop Computer. All BASIC language syntax and machine error codes are listed here.

Turn-on

Your desktop computer is ready to use when the power switch on the right hand side is set to the "1" position and the flashing cursor appears near the bottom left hand corner of the CRT.

Memory Size

The standard System 45 has 13498 bytes of available Read/Write Memory. The memory can be expanded in increments of 16384 bytes up to 62650 bytes. The current memory size is indicated on the fourth pull out card under the CRT.

Range

The range of values which can be entered or stored is $-9.9999999999 \times 10^{99}$ through -1×10^{-99} , 0, and 1×10^{-99} through $9.9999999999 \times 10^{99}$. However, the range of calculations is $-9.9999999999 \times 10^{511}$ through -1×10^{-511} , 0, and 1×10^{-511} through $9.9999999999 \times 10^{511}$.

Data Precision

Ten bytes of memory are allocated per simple variable for full (12 digit) precision. All calculations are performed with 12 digit precision. Short precision or integer precision can be used when it is necessary to conserve memory space.

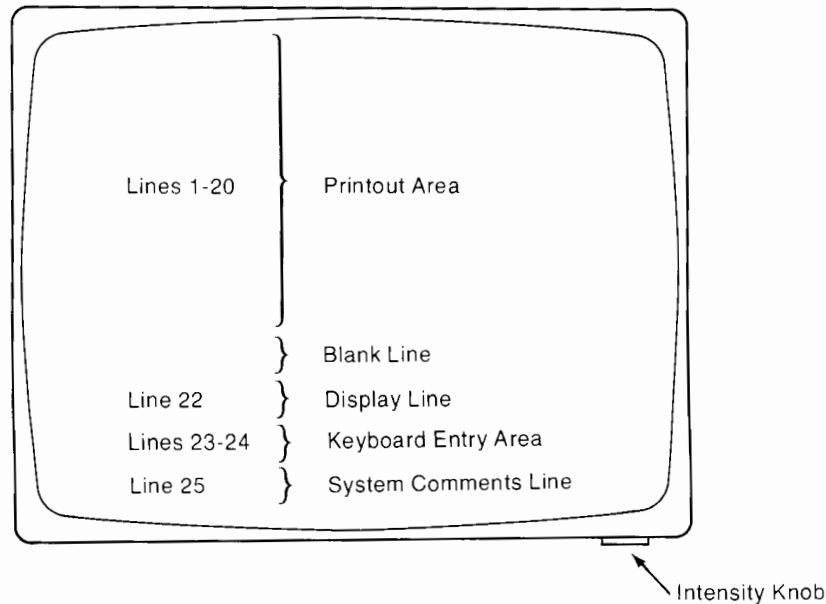
Precision	Bytes Per Simple Variable	Accuracy	Range
Full (Real)	10	12 Digits	$\pm 9.9999999999E \pm 99$
Short	6	6 Digits	$\pm 9.99999E \pm 63$
Integer	4	Integer	-32768 through 32767

A string requires six bytes plus 1 byte per character rounded up to an even integer.

The CRT

The CRT has four areas. Lines 1 through 20 are a printout area. Line 22 is the display line. The keyboard entry area, lines 23 and 24, allows 160 characters to be typed in. The system comments line, line 25, is for error messages, other indicators and results of calculations.

CRT brightness is controlled by a knob underneath the lower right hand side of the CRT.



Error Codes

When an error occurs, the machine beeps and displays an error number or message in the system comments line. The meanings of the error numbers are listed in the back of this guide.

CRT Pull-out Cards

The four cards under the CRT serve as a handy reference for operating the System 45. They are:

- Error Messages
- ROM Error Messages
- Statements
- About the 9845A

ROMs

Plug-in option ROMs (Read Only Memories) expand the language and capabilities of the standard desktop computer.

Special Keys

In addition to the standard typewriter keys there are blocks of keys with special functions.

Typing Keys




Sets a tab at the location of the cursor.



Clears the tab at the location of the cursor.



Sets the keyboard to typewriter mode. Letters are entered in lower case, upper case when shifted. When pressed with  held down, the keyboard is set to the space dependent mode.




Causes any key held down to repeat rapidly.

Program Control Keys



Runs the program, beginning at the lowest numbered line.



Stops a running program and all I/O at the end of the current line. When pressed with  held down, the computer is reset.



Causes program execution to be suspended at the end of the current line; I/O continues. The next line to be executed is displayed in the system comments line.



Continues execution of a program from where it was suspended.



Stores an individual program line in the keyboard entry area into memory.



Executes the line or expression in the keyboard entry area.

Edit / System Function Keys




Allows various modes and special ASCII control characters to be entered when held down while another key is pressed.



Steps through a program, executing one line at a time.



Recalls previous keyboard entries one at a time in a last-in-first-out basis. When pressed with  held down, more recent entries are recalled.



When latched, the computer is in the print all mode, providing output of all computations, results, stored lines, and error messages to the print all printer.



When latched at power on, the computer executes

```
LOAD "AUTOST:T15",1
```

at power on.



Deletes the program line which is in the keyboard entry area from memory while in the edit line mode.



Allows lines to be inserted between program lines while in the edit line mode.



Deletes the character at the position of the flashing cursor.



Allows characters to be inserted to the left of the flashing cursor.

Display Keys










Clears the entire CRT of everything except any system indicators.



Clears the keyboard entry area from the position of the flashing cursor to the end. It also clears the system comments line of everything except any system indicators.





Clears the keyboard entry area and system comments line of everything except any system indicators.

	Moves the cursor to the home position – the first position in the keyboard entry area.
	Moves the cursor one character position to the left. Pressing down hard causes rapid repetition. If the cursor is in the home position, it moves to the end of the line.
	Moves the cursor one character position to the right. Pressing down hard causes rapid repetition. If the cursor is at the end of the line, it moves to the home position.
	Moves the lines in the printout area of the CRT up one line. Pressing down hard causes rapid repetition.
	Moves the lines in the printout area of the CRT down one line. Pressing down hard causes rapid repetition.
	Moves the lines in the printout area of the CRT up 10 lines, 5 in edit line mode.
	Moves the lines in the printout area of the CRT down 10 lines, 5 in edit line mode.

Numeric Keys

All keys needed to enter numbers and do simple arithmetic are located in this area.

	Enters an E to indicate that an exponent follows.
	A typing aid for the RESult function; RES is entered into the keyboard entry area.

Special Function Keys

The special function keys (SFK's) provide various display modes and typing aids. They can also be defined as typing aids; see `EDIT KEY`.

Language Terms

The System 45 BASIC language consists of **statements**, **functions**, **operators** and **commands**. Operators and functions are used with variables and numbers in creating numeric **expressions**. Expressions and functions can be included in statements and executed from the keyboard. Each statement can also be preceded by a line number and stored as a program line. Some functions and statements can also be separately executed from the keyboard. Commands can only be executed from the keyboard; they are not programmable.

Operators

Arithmetic

+	Add
-	Subtract, unary minus
*	Multiply
/	Divide
^ or **	Exponentiate
MOD	Modulo
DIV	Divide-return integer portion

Relational

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

Logical

AND
OR
EXOR
NOT

Truth Table

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

1 = True 0 = False

String

& String concatenation

Math Hierarchy

Functions

^

NOT, unary -

* / MOD DIV

+ -

Relational Operators (=, <, >, <=, >=, <>, or #)

AND

OR EXOR

Performed First



Performed Last

The order of execution for operators of the same level is from left to right. However, operations within parentheses are performed first.

BASIC Syntax Guidelines

These terms and conventions are used in the syntax listing.

[] – all items enclosed in brackets are optional unless the brackets are in dot matrix.

`dot matrix` – all items in dot matrix must appear as shown.

... – three dots indicate that the previous item can be duplicated.

| – a vertical line between parameters means “or”, only one of the two parameters may be included.

/ – a slash between two parameters means that either or both parameters may be included.

name – a capital letter followed by 0 through 14 lowercase letters, digits or the underscore character. The following parameters can be names.

- variables
- labels
- subprogram names

line number – an integer from 1 through 9999. In most cases, when a line number specified is not in memory, the next higher line is accessed.

label – a unique name given to a program line. It follows the line number and is followed by a colon.

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

numeric expression – a logical combination of variables, constants, operators and functions (including user-defined functions) grouped within parentheses as necessary. However, multiple line user-defined functions can't be used in any I/O statement.

The following parameters can be numeric expressions:

- scalar
- initial value
- final value
- increment value (increments dealing with line numbers must be integer constants)
- number of (significant) digits
- number of repetitions
- number of characters per line
- number of line feeds
- number of spaces
- power-of-ten position
- character position
- number of milliseconds
- redim subscripts
- subprogram dimensioning subscripts
- key number (in `ON KEY#` statement only)
- priority
- defined-record number
- file number
- select code
- HP-IB device address
- unit code
- 9885 unit code
- controller address
- heading suppression
- record length
- number of defined records
- interleave factor



string expression – the forms a string expression can take are:

- text within quotes
- string variable
- substring
- string concatenation operation – &
- string function (including user-defined)

The following parameters can be string expressions:

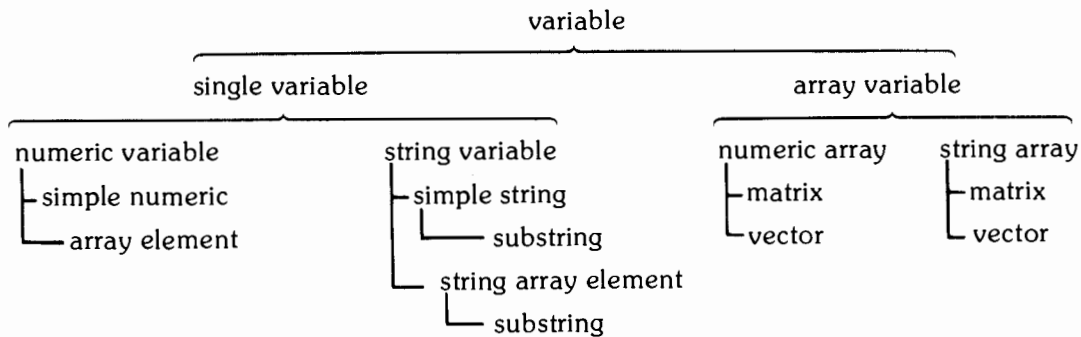
file name
 mass storage unit specifier
 file specifier
 protect code
 selective catalog specifier
 image format string

constant – a number within the System 45's range, such as 2E12, which can't be altered.

character – a letter, number, symbol or ASCII control code (ASCII control codes can be obtained using the control key); any arbitrary 8-bit byte defined by the CHR\$ function.

text – a string of characters, quoted (literal) or unquoted as specified.

variable – a name which is assigned a value and specifies a location in memory; variable types and forms are broken down –



(*) identifies the entire array in an I/O statement

pass parameters – includes variables, array identifiers, expressions and files specified by # file number; any variable can be enclosed in parentheses causing it to be passed by value.

formal parameters – includes non-subscripted variables, array identifiers and files specified by # file number.

redim subscripts – numeric expressions separated by commas and enclosed in parentheses. The number of dimensions can't change; the total number of elements can't increase over the number originally dimensioned.

Mass Storage

file number – the number assigned to a mass storage data file by an `ASSIGN` statement. Its range is one through ten.

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

select code – an expression (rounded to an integer) in the range zero through sixteen. The following select codes are reserved –

- 0 Internal thermal printer
- 13 Graphics Option
- 14 Optional tape cartridge
- 15 Primary tape cartridge
- 16 CRT

mass storage unit specifier – : device type [select code [, controller address | 9885 unit code [, unit code]]]

The letters specifying the various device types are:

- T – Tape Cartridge
- F – 9885 Flexible Disk
- Y – 7905 Removable Platter
- Z – 7905 Fixed Platter
- C – Reserved for future disc
- D – Reserved for future disc
- P – 7920A Disc Pack

Controller address range is zero through seven; 9885 unit code is zero through three; unit code is zero through seven and is ignored if used with 9885 unit code. See the Mass Storage Techniques Manual for further explanation. Mass storage unit specifier is abbreviated msus.

file specifier – a string expression of the form: file name [mass storage unit specifier]

protect code – any valid string expression except one with a length of zero. Only the first six characters are recognized as the protect code, however.

Syntax Listing

Mainframe

A

ABS numeric expression

The **ABS** function returns the absolute value of the numeric expression.

```
10 PRINT ABS(X+4)
20 Y=SQR(ABS(A-B))
```

ACS numeric expression

The **ACS** function returns the principal value of the arccosine of the numeric expression expressed in the current angular unit mode.

```
40 Angle=ACS(X)
50 PRINT ACS(.5)
```

ASN numeric expression

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

```
70 Sine=ASN(X)
80 IF ASN(A)<90 THEN 200
```

ASSIGN file specifier TO# file number [, return variable [, protect code]]

ASSIGN # file number TO file specifier [, return variable [, protect code]]

ASSIGN * TO# file number

ASSIGN # file number TO*

The **ASSIGN** statement is used to open a data file by assigning a number to it. It is also used to close a data file by assigning * to the file number. The return variable can be any simple variable. The protect code is required if the file was protected.

Value of Return Variable	Meaning
0	File available
1	No such file found
2	File is protected or of the wrong type

```
100 ASSIGN #1 TO "DATA"
110 ASSIGN "DATA:F8" TO #2,Return ! Return IS THE RETURN VARIABLE
111 ASSIGN #7 TO "Prog",X,"XYZ" ! PROTECT CODE ALSO SPECIFIED
120 ASSIGN * TO #2 ! CLOSES FILE #2
```

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

```
140 Angle=ATN(Z)
150 PRINT ATN(1)
```

`AUTO` [beginning line number [, increment value]]

The `AUTO` command allows program lines to be numbered automatically as lines are stored. If no parameters are specified, numbering begins with ten and is incremented by ten.

```
AUTO
AUTO 50
AUTO 10,5
```

B

`BEEP`

The `BEEP` statement outputs an audible beep.

`BUFFER#` file number

The `BUFFER` statement attaches a 268 byte, semi-permanent buffer from user Read/Write Memory to the specified file to reduce mass storage device transfers.

```
10 BUFFER #4
30 BUFFER #N
```

C

CALL subprogram name [(pass parameter list)]

The CALL statement transfers control to a subroutine subprogram.

```
210 CALL Mean(X,Y,Z(*),D(1,4),(C),#4,H#)
280 CALL Price(X*Y,(B#[5]),PI,(D(1,3)),44)
```

CAT [selective catalog specifier / msus [, heading suppression]]

CAT # select code [, HP-IB device address] [; selective catalog specifier / msus [, heading suppression]]

The CAT (catalog) statement outputs information about user files: file names, types, and physical specifications. The selective catalog specifier is a string expression of 1 through 6 characters; only those files whose names begin with that combination of characters are cataloged. The heading is suppressed if the value of the numeric expression rounds to one. The second syntax directs the catalog output to the specified device.

```
100 CAT ":T"          ! CATALOGS THE TAPE CARTRIDGE
110 CAT "Pr"         ! CATALOGS ALL FILES BEGINNING WITH "Pr"
120 CAT ":F8",1     ! SUPPRESSES THE HEADING
```

CHECK READ [# file number]

The CHECK READ statement is used to provide verification that the information being stored on a storage medium is identical to that in memory. This verification may be specified for a certain file only. CHECK READ also forces the output of data if it is not buffered.

```
140 CHECK READ      ! ALL FILES WILL BE VERIFIED
150 CHECK READ #6   ! ONLY FILE #6 WILL BE VERIFIED
```

CHECK READ OFF [# file number]

The CHECK READ OFF statement deactivates a previous corresponding CHECK READ statement, either general or for a specified file.

```
170 CHECK READ OFF ! ALL CHECK READ DISABLED
180 CHECK READ OFF # 6 ! CHECK READ FOR FILE #6 DISABLED
```

CHR\$ numeric expression

The CHR\$ (character) function converts a numeric value between -32768 and 32767 into a string character. Any number out of the range 0 through 255 is converted MOD 256 to that range.

```
200 PRINT CHR$(27)           ! PRINTS AN ESCAPE CODE
210 IF CHR$(N)="A" THEN 100 ! BRANCHING WHEN N=65
```

COL array variable

The COL function returns the number of columns (rightmost subscript) in the working size of the specified numeric or string array.

```
230 A=COL(B)
240 REDIM Matrix(5, COL(Array))
```

COM item [, item...]

The COM statement dimensions and reserves memory space for simple and array variables—numeric and string—in a “common” memory area, allowing values to be passed to subprograms or to other programs.

The item can be –

simple numeric

numeric array (subscripts)

simple string [[number of characters]]

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

REAL, SHORT or INTEGER can precede one or more numeric variable items to specify precision.

```
260 COM H(3,2),J(1,2,3),K#[56]
270 COM B(3,2),C(1,2,3),D$(2,3)[56],INTEGER E(4,4)
300 COM H(*),REAL I,J(5,4) ! (*) CAN BE USED IN SUBPROGRAM COM
```

CONT [line identifier]

The CONT (continue) command resumes execution of a program at the specified line, or where it was halted, without altering program conditions and modes.

```
CONT
CONT 270
CONT Sub
```

`COPY` source file specifier TO destination file specifier [, protect code]

The `COPY` statement selectively copies a file from one location to a previously undefined file. The protect code is used only if the source file is protected.

```
340 COPY "DATA1:F8" TO "DATA1:T15"    ! DISK TO TAPE
350 COPY "RES-A" TO "RES-B"          ! SAME MEDIUM
```

`COS` numeric expression

The `COS` function returns the cosine of the angle which is represented by the numeric expression.

```
370 IF COS(N)=1 THEN 450
380 PRINT COS(Angle)
```

`CREATE` file specifier, number of defined records [, record length]

The `CREATE` statement establishes a data file of the specified size and places an EOF mark in the first word of every record.

```
400 CREATE "DATA",7                ! SEVEN 256-BYTE RECORDS
410 CREATE "ACCTS:F8",10,400       ! TEN 400-BYTE RECORDS
```

D

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

The `DATA` statement provides constants and quoted or unquoted text from which `READ` and `MAT READ` obtain values for numeric and string variables. It can't be executed from the keyboard.

```
20 DATA 99,74,53.9,86,92,81,75,100
30 DATA JONES,"SMITH",BROWN ! TEXT CAN BE QUOTED OR UNQUOTEI
40 DATA "J.C.",5,"F.B",7,"M.C.",1
```

`DEFAULT OFF`

The `DEFAULT OFF` statement cancels any `DEFAULT ON` statement previously executed.

DEFAULT ON

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

The default values are:

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

```
DEF FN subprogram name [ (formal parameter list) ] = numeric expression
```

```
DEF FN subprogram name $ [ (formal parameter list) ] = string expression
```

```
DEF FN subprogram name [ (formal parameter list) ]
```

```
DEF FN subprogram name $ [ (formal parameter list) ]
```

The `DEF FN` statement defines a single-line function (first two syntax), or, with `RETURN` and `FN END`, a multiple-line function subprogram (second two syntax).

```
60 DEF FNC$(A$,B$)=A$&B$&C$           ! SINGLE-LINE STRING
70 DEF FNChange=M^3-FNSlope(M,N,0)    ! SINGLE-LINE NUMERIC
80 DEF FNShipments(A,B,C)             ! MULTIPLE-LINE NUMERIC
90 DEF FNResults$(X$,Y$,Z$)           ! MULTIPLE-LINE STRING
```

DEG

The `DEG` statement is used to set degree mode for results and arguments of trigonometric functions. A degree is 1/360th of a circle.

`DEL first line identifier [, second line identifier]`

The `DEL` command is used to delete a line or section of a program. If only one line identifier is specified, just that line is deleted. If two line identifiers are specified, the block of lines is deleted.

```
DEL50
DEL Sub,Subend
```

`DET`

`DET` matrix variable

The `DET` function returns the determinant of the specified numeric matrix or of the last numeric matrix which was inverted if no matrix is specified.

```
140 IF DET=0 THEN 200 ! CHECKS DETERMINANT OF LAST INVERTED MATRIX
160 PRINT DET(Array)
```

`DIM item [, item...]`

The `DIM` statement is used to declare the number of dimensions and the maximum number of elements in each dimension for real precision array variables and initializes all elements to zero. The `DIM` statement is also used to define the maximum length of simple and array string variables, declare the number of dimensions and maximum number of elements in each dimension and initialize all strings to the null string.

The item can be –

numeric array (subscripts)

simple string [number of characters]

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

```
170 DIM L(2,-2:2,3),N#[100]           ! ARRAY AND SIMPLE STRING
180 DIM A(-2:2),B(5,2,2),C(2,2,2,2)  ! THREE ARRAYS
181 DIM M$(2,2)[50],O$(5)           ! TWO STRING ARRAYS
```

`DISABLE`

The `DISABLE` statement deactivates any `ON KEY#` interrupt declarative so that pressing that key has no effect on current program control, but the interrupt is recorded.

DISP [display list]

The **DISP** statement causes the items specified in the display list to be displayed in the display line of the CRT. The items can be variables, expressions, **SPA**, and **TAB**, separated by commas or semicolons. No multiple line user-defined functions can be specified in the display list.

```
200  DISP "X EQUALS";X,"A EQUALS";A,G#
210  DISP "COMPUTING";           ! NEXT DISPLAY WILL BE APPEN
220  DISP "PRESS CONTINUE TO PROCEED"
```

DOT (vector name, vector name)

The **DOT** function returns the inner product of two vectors.

```
240  PRINT DOT(Vector1,Vector2)
250  X=DOT(A,B)
```

DROUND (numeric expression, number of significant digits)

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

```
270  X=DROUND(PI,5)           ! PI ROUNDED TO 5 SIGNIFICANT DIGITS
280  PRINT DROUND(N,X)
```

E

EDIT ["prompt",] string variable

The **EDIT** statement allows the stored value of a string variable of up to 160 characters in length to be altered. **EDIT** can't be executed from the keyboard.

```
20  EDIT "NEW VALUE FOR A#?",A#
30  EDIT B#[20]           ! EDITING A SUBSTRING
```

EDIT KEY key number

EDIT kn

The **EDIT KEY** command allows an SFK to be defined or redefined as a series of keystrokes for use as a typing aid.

```
EDIT KEY 12
EDIT KEY 20
```

`EDIT LINE [line identifier [, increment value]]`

The `EDIT LINE` command allows program lines to be changed, added or deleted. If no line identifier is specified, the first line in the program is accessed. The increment value is ten if not specified.

```
EDIT LINE 100
EDIT LINE
EDIT LINE Sub,5
```



`ENABLE`

The `ENABLE` statement reactivates any `ON KEY#` interrupt declaratives that were previously deactivated by `DISABLE`.

`END`

The `END` statement is the last (highest numbered) statement in a main program and terminates program execution.

`ERRL`

The `ERRL` (error line) function returns the line number in which the most recent program execution error occurred.

```
120 PRINT ERRL          ! PRINTS WHAT LINE THE ERROR WAS IN
130 IF ERRL=150 THEN STOP
```

`ERRM$`

The `ERRM$` (error message string) function returns the most recent program execution error message.

```
150 PRINT ERRM$
160 IF ERRM$="ERROR 31 IN LINE 100" THEN X=5
```

`ERRN`

The `ERRN` (error number) function returns the number of the most recent program execution error.

```
180 PRINT ERRN
190 IF ERRN=36 THEN RESTORE ! REPOSITION DATA POINTER WHEN OUT
```

EXP numeric expression

The EXP (exponential) function returns the value of Napierian e ($= 2.71828182846$ to twelve place accuracy) raised to the power of the computed expression.

```
210  A=EXP(1)
220  X=EXP(N)
```

F

FIXED number of digits

The FIXED statement sets fixed mode for output of numeric values and specifies from zero through twelve digits to the right of the decimal point.

```
240  FIXED 5      ! 5 DIGITS TO RIGHT OF DECIMAL POINT
250  FIXED N      ! VARIABLE NUMBER OF DIGITS
```

FLOAT number of digits

The FLOAT statement sets floating point mode (scientific notation) for output of numeric values and specifies the number of digits to the right of the decimal point in the range zero through eleven.

```
270  FLOAT 7      ! 7 DIGITS - SCIENTIFIC NOTATION
280  FLOAT X      ! VARIABLE NUMBER OF DIGITS
```

FN END

The FN END statement is the last line in a multiple line function subprogram.

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

The FOR statement is used with the NEXT statement and defines how many times a FOR-NEXT loop is to be executed. The loop counter must be a simple variable. If no increment value is specified, it defaults to one.

```
320  FOR I=1 TO 10
330  FOR X=X-2 TO X+5 STEP X/5  ! COUNTER VARIABLE CAN ALSO BE
                                USED TO DEFINE VALUES
```

FRACT numeric expression

The `FRACT` function returns the fractional part of the evaluated expression and is defined by the formula: $\text{argument} - \text{INT}(\text{argument})$.

```
350 PRINT FRACT(N)
360 X=FRACT(-6.257)
```

G

GET file specifier [, line identifier [, execution line identifier]]

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

```
20 GET "Prog"
30 GET "PAYROL:F8",70 ! RENUMBER TO BEGIN WITH 70
40 GET "ACCTS",100,10 ! RENUMBER TO BEGIN WITH 100, EXECUTE AT 10
```

GOSUB line identifier

The `GOSUB` statement transfers program control to the subroutine beginning at the specified line in the current program segment.

```
60 GOSUB 100
70 GOSUB Output
```

GOTO line identifier

The `GOTO` statement transfers program control to the specified line in the current program segment.

```
90 GOTO 400
100 GOTO Print
```

GRAD

The `GRAD` statement is used to set grad mode for all results and arguments of trigonometric functions. A grad is $1/400$ th of a circle.

I

IF numeric expression THEN line identifier

IF numeric expression THEN executable statement

The IF...THEN statement provides conditional branching. If the numeric expression is evaluated as true, execution is transferred to the specified line or the statement is executed. The following statements can't follow THEN:

COM statement

DATA statement

DEF FN statement

DIM statement

END statement

FN END statement

IF statement

IMAGE statement

INTEGER statement

OPTION BASE statement

REAL statement

REM statement

SHORT statement

SUB statement

SUBEND statement

```
120 IF X THEN 950           ! BRANCHING WHEN X<>0
130 IF X+5=Y THEN PRINT "X=";X
140 IF B>78 THEN STOP
150 IF S>99 THEN GOSUB 900
```

IMAGE image format string

The `IMAGE` statement is used with the `PRINT USING` statement and specifies output format: numeric and string field specifiers, blanks, and carriage control.

Field specifiers must be separated by a comma, `@` or slash.

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

D	Specifies a digit position. The fill character is a blank. <code>N D</code> specifies N digit positions.
Z	Specifies a digit position. The fill character is a zero. <code>N Z</code> specifies N digit positions.
*	Specifies a digit position. The fill character is an asterisk. <code>N*</code> specifies N digit positions.
X	Causes a blank to be printed. <code>N X</code> causes N blanks to be printed.
A	Specifies a single string character position. <code>N A</code> specifies N string characters.
.	Indicates placement of a decimal point radix indicator. There may be only one radix indicator per numeric specifier.
R	Indicates placement of a comma radix indicator. There may be only one radix indicator per numeric specifier.
C	Indicates placement of a comma in a numeric specification. It is a conditional character and is output only if there is a digit to its left.
P	Indicates placement of a period in a numeric specification. It is a conditional character and is output only if there is a digit to its left.
S	Indicates a sign position for a + or -. The sign floats to the left of the leftmost significant digit if <code>S</code> appears before all digit symbols.
M	Indicates a sign position; + is replaced by a blank. The sign floats to the left of the leftmost significant digit if <code>M</code> appears before all digit symbols.

E	Causes output of an E, sign and two digit exponent. This is used for output of numbers in scientific notation.
K	Specifies an entire string or numeric field. A numeric is output in STANDARD format, except that no leading or trailing blanks are output. The current value of a string is output.
+	Suppresses line feed.
-	Suppresses carriage return.
#	Suppresses CR-LF.
	If control of the normal CR-LF output at the end of the PRINT USING statement is desired, one of these three characters must be the first symbol in the format string.
@	Outputs a form feed.
/	Causes a CR-LF to be output. N/ causes N CR-LFs to be output.
()	Parentheses allow specifiers to be replicated.
" "	Specifies text.

```

170 IMAGE AAAA,3A,"#X5D,DD,4Z,3D,3*Z.2D
180 IMAGE 2DC3D,2D,D,DDE,M4D,":F"2D,("&4D.2D")"
190 IMAGE DDD.DD,"END OF TEXT",2(@)

```

INITIALIZE mass storage unit specifier [, interleave factor]

The INITIALIZE statement enables an unused mass storage medium to be used by establishing physical records and main and spare directories. A used medium can also be re-initialized; in the process, it is cleared of all information it contains. The interleave factor, an integer from one through ten, defines number of revolutions per track on the HP 9885 Disk; seven is the default value.

```

210 INITIALIZE ":T"
220 INITIALIZE ":F8",4 ! INTERLEAVE FACTOR OF 4

```


INPUT ["prompt " ,] variable name [, ["prompt " ,] variable name ...]

The INPUT statement allows values to be assigned to variables from the keyboard during program execution. INPUT can't be executed from the keyboard.

```
240 INPUT "A AND B",A,B
250 INPUT "VALUE OF X?",X,"VALUE OF Y?",Y
260 INPUT I(*),A$,B#[5]
```

INT numeric expression

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

```
280 X=INT(N)
290 Y=INT(22.369)
```

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

The INTEGER statement is used to dimension and reserve storage space for integer precision variables – simple and array.

```
310 INTEGER A(2,2),B
320 INTEGER C(2,23,5),D,Hours(10)
```



L

LEN string expression

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

```
30 PRINT LEN(C$)
40 IF LEN(S$)<10 THEN S#=RPT$(S$,2)
```

[LET] numeric variable [= numeric variable ...] = numeric expression

[LET] string variable [=string variable ...] = string expression

The LET statement is used to assign a value to a variable or variables.

```
70 LET A=B=C=D=E=100 ! MULTIPLE ASSIGNMENTS
80 X=Y^2 ! IMPLIED ASSIGNMENT
90 A#=C$(2,5)
100 LET X$="PRICES"
```

LGT numeric expression

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

```
130 X=LGT(Y)
140 PRINT LGT(2)
```

LIN number of line feeds

The LIN function is used with the PRINT statement and causes a carriage return and the specified number of line feeds to be output. The range of the numeric expression specifying the number of line feeds is -32768 through 32767; a negative number suppresses the carriage return.

```
170 PRINT "Name of participant",LIN(10),"Hometown"
180 PRINT A,LIN(-5),B,LIN(-5),C          ! NO CARRIAGE RETURN
```

LINK file specifier [, line identifier [, execution line identifier]]

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

```
210 LINK "DATA"
220 LINK "Res:F8",100          ! RENUMBER TO BEGIN WITH 100
230 LINK "M-N",200,10        ! BEGIN EXECUTION WITH LINE 10
```

LINPUT ["prompt" ,] string variable

The LINPUT statement allows any combination of characters to be assigned to a string variable during program execution. LINPUT can't be executed from the keyboard.

```
260 LINPUT A#[10,20]
270 LINPUT "NEXT NAME",Names$(I)
```

LIST [beginning line identifier [, ending line identifier]]

LIST# select code [, HP-IB device address] [; beginning line identifier[, ending line identifier]]

The LIST command outputs a listing of all or part of program lines in memory in order from lowest numbered to highest numbered line. If one line identifier is specified, the listing begins with that line. If two line identifiers are specified, that block of lines is listed. The second syntax directs the listing to the specified device.

```
LIST #6
LIST 50
LIST #0;50,150
```

LIST KEY [key number]

LIST KEY# select code [, HP-IB device address] [; key number]

LIST _{Kn}

The LIST KEY command causes the typing aid definitions of one or all SFKs to be listed.

```
LIST KEY
LIST KEY #6
LIST KEY 8
LIST KEY #16;8
```

LOAD file specifier [, execution line identifier]

The LOAD statement puts back into memory a program stored with the STORE statement, destroying any lines in memory. The execution line identifier specifies where execution is to begin.

```
400 LOAD "Prog:F8"
410 LOAD "FILE",50 ! BEGIN EXECUTION WITH LINE 50
```

LOAD ALL file specifier

The LOAD ALL command causes an implied SCRATCH A to be executed, loads a file stored previously with the STORE ALL statement and restores the entire memory to the state it was in when STORE ALL was executed. All files used must be re-assigned, however.

```
LOAD ALL "MEMORY"
LOAD ALL "12:T"
```

LOAD BIN file specifier

The **LOAD BIN** statement loads the specified binary file into memory without altering any other binary routines already in memory.

```
480 LOAD BIN "ROUTIN"
490 LOAD BIN "Dup:F8"
```

LOAD KEY file specifier

The **LOAD KEY** statement loads SFK definitions from a file stored with **STORE KEY**. Program lines in memory aren't affected.

```
520 LOAD KEY "PROG 1"
530 LOAD KEY "PROG 2"
```

LOG numeric expression

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

```
560 X=LOG(Y)
570 PRINT LOG(55)
```

LWC\$ string expression

The **LWC\$** (lowercase) function returns a string with all uppercase letters converted to lowercase.

```
600 A$=LWC$(A$)
610 PRINT LWC$("ABcdeF#0$")    ! ONLY LETTERS ARE AFFECTED
```

M

`MASS STORAGE IS` mass storage unit specifier

The `MASS STORAGE IS` statement specifies the standard (default) mass storage device. Mass storage unit specifier is a string expression of the form – `: device type [select code [, controller address | 9885 unit code [, unit code]]]`. Unit code is ignored if 9885 unit code is specified.

The letters specifying the various mass storage device types are:

Letter	Device
T	Tape cartridge
F	9885 Flexible Disk
Y	7905 Removable Platter
Z	7905 Fixed Platter
C	Reserved for future disc
D	Reserved for future disc
P	7920A Disc Pack

```
20 MASS STORAGE IS ":T"      ! 15 IS DEFAULT SELECT CODE
30 MASS STORAGE IS A#       ! MSUS CAN BE STRING VARIABLE
40 MASS STORAGE IS ":F8,3"
```

`MAT array variable = CON [(redim subscripts)]`

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

```
60 MAT A=CON
70 MAT C=CON(4,4)      ! REDIMENSION ARRAY C
```

`MAT result array = operand array`

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

```
90 MAT Data1=Data2
100 MAT X=Y
```

MAT result vector = CSUM operand matrix

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

```
120 MAT Vector=CSUM(Array)
130 MAT X=CSUM(Y)
```

MAT result array = function operand array

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

```
150 MAT Pay=INT(Base) ! USE ANY SINGLE-ARGUMENT SYSTEM FUNCTION
160 MAT A=SIN(B)
```

MAT matrix name = IDN [(redim subscripts)]

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

```
180 MAT I=IDN
190 MAT Test=IDN(4,4) ! REDIMENSION MATRIX
```

MAT array variable = (numeric expression)

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

```
210 MAT C=(4) ! ASSIGN 4 TO EACH ELEMENT
220 MAT X=(N^2) ! ASSIGN N^2 TO EACH ELEMENT
```

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

The MAT INPUT statement allows values to be assigned from the keyboard to the elements of an array during program execution. A new working size can be specified for any array.

```
240 MAT INPUT A(2,2),B ! REDIMENSIONS ARRAY A
250 MAT INPUT Data
```

`MAT result matrix = INV operand matrix`

The `MAT...INV` statement establishes a square matrix as the inverse of the specified square matrix. A matrix doesn't have an inverse if its determinant is zero.

```
270 MAT A=INV(A)
280 MAT C=INV(D)
```

`MAT result matrix = operand matrix1 * operand matrix2`

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

```
282 MAT Pay=Hours*Rate
290 MAT A=B*C
```

`MAT result array = operand array operator operand array`

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

The following operators are allowed:

```
+
-
. (multiply)
/
=
< > or #
>
<
>=
<=
```

```
300 MAT Sum=A+B
310 MAT Total=Pay+Overtime
```

`MAT PRINT array variable [, or ; [array variable ...]]`

The `MAT PRINT` statement causes the specified arrays to be printed on the standard printer.

```
320 MAT PRINT A;      ! TIGHT SPACING
330 MAT PRINT C;D
340 MAT PRINT X      ! 20 CHARACTER FIELDS
```

`MAT PRINT# file number [, defined-record number] ; array variable [, array variable, ...] [, END]`

The `MAT PRINT#` statement is used to record all of the elements of the specified arrays onto a mass storage medium. `END` causes an EOF to be printed after the data.

```
360 MAT PRINT #1;A,B      ! SERIAL PRINT
370 MAT PRINT #3,4;X      ! RANDOM PRINT
```

`MAT READ array variable [(redim subscripts)] [, array variable [(redim subscripts)], ...]`

The `MAT READ` statement specifies that values for all the elements in an array or arrays are to be read from a `DATA` statement or statements which specify the values. A new working size can be specified for any of the arrays.

```
390 MAT READ H(4,4),J      ! REDIMENSION H
400 MAT READ A
```

`MAT READ# file number [, defined record number] ; array variable [(redim subscripts)] [, array variable [(redim subscripts)], ...]`

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

```
420 MAT READ #2;B
430 MAT READ #4,6;A,C(2,3,5) ! REDIMENSION C
```

`MAT result vector = RSUM operand matrix`

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

```
450 MAT A=RSUM(B)
460 MAT Totals=RSUM(Data)
```




Manual Changes

HP System 45 Desktop Computer Reference Guide

For Manual P/N 09845-90010, Dated February 2, 1978

Page 34

The syntax for the MAT READ# statement should read –

MAT READ# file number [, defined record number]; array variable [, array variable, ...]



HEWLETT · PACKARD

HEWLETT · PACKARD

HEWLETT · PACKARD

April 1, 1978

Supplement A to 09845-90010

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

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

The `MAT`-scalar operation statement allows an arithmetic or relational operation to be performed on each element of a numeric array using a constant scalar (numeric expression); the result becomes the value of the corresponding element of the result array.

The following operators are allowed:

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



```
480 MAT A=(3)+B           ! ADD 3 TO EACH ELEMENT
490 MAT Rate=Pay/(40)    ! DIVIDE EACH ELEMENT BY 40
500 MAT X=(5)>=Y         ! 1's OR 0's IN ARRAY X
```

`MAT result matrix = TRN operand matrix`

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

```
520 MAT Rows=TRN(Colums)
530 MAT C=TRN(D)
```

`MAT array variable = ZER [(redim subscripts)]`

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

```
550 MAT Zero=ZER
560 MAT Blank=ZER(4,5,2) ! REDIMENSION Blank
```

`MAX (list of numeric expressions)`

The `MAX` (maximum) function returns the greatest value in the list. The numeric expressions must be separated by commas.

```
580 X=MAX(A,B,C,D,PI,40)
590 PRINT MAX(20,X,Y,X+Y)
```

`MIN (list of numeric expressions)`

The `MIN` (minimum) function returns the smallest value in the list. The numeric expressions must be separated by commas.

```
610 Minimum=MIN(A(1),A(2),A(3),A(4))
620 PRINT MIN(M,N,0)
```

N

NEXT loop counter

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

```
20  NEXT I
30  NEXT X
```

NORMAL

The **NORMAL** statement cancels all tracing operations.

NUM string expression

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

```
50  X=NUM(A#[20])
60  IF NUM(C#)=14 THEN 200
```

O

OFF END# file number

The **OFF END#** statement deactivates a corresponding **ON END#** statement.

```
80  OFF END #6
90  OFF END #N
```

OFF ERROR

The **OFF ERROR** statement cancels any **ON ERROR** condition currently active.

OFF KEY# key number

The **OFF KEY#** statement deactivates a corresponding **ON KEY#** statement; pressing the key then has no effect on program control.

```
110 OFF KEY #5
120 OFF KEY #22
```

```
ON END# file number GOTO line identifier
ON END# file number GOSUB line identifier
ON END# file number CALL subprogram name
```

The `ON END#` statement declares a branching that is to occur when an EOR or EOF mark is encountered during a `READ#` or `PRINT#` operation to that file so that the condition can be serviced. It also forces serial mode I/O to that file. No parameters can be passed to the subprogram when `CALL` is used.

```
130 ON END #5 GOTO 200
140 ON END #3 GOSUB Openfile
150 ON END #8 CALL Data3      ! CAN'T PASS PARAMETERS
```

```
ON ERROR GOTO line identifier
ON ERROR GOSUB line identifier
ON ERROR CALL subprogram name
```

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

```
170 ON ERROR GOTO Next
180 ON ERROR GOSUB 250
190 ON ERROR CALL Sub      ! CAN'T PASS PARAMETERS
```

```
ON numeric expression GOSUB line identifier list
```

The `ON...GOSUB` (computed GOSUB) statement allows any one of one or more subroutines in the current program segment to be accessed based on the value of the numeric expression. A value of one corresponds to the first line identifier in the list, two to the second, etc.

```
210 ON X GOSUB 120,Print,450
220 ON Average+3 GOSUB 100,Mean,Stddev
```

```
ON numeric expression GOTO line identifier list
```

The `ON...GOTO` (computed GOTO) statement allows program control to be transferred to one of one or more statements in the current program segment based on the value of the numeric expression. A value of one corresponds to the first line identifier in the list, two to the second, etc.

```
240 ON Time GOTO First,Second,Third
250 ON H GOTO 100,200,300
```

```
ON KEY# key number [, priority] GOTO line identifier
ON KEY# key number [, priority] GOSUB line identifier
ON KEY# key number [, priority] CALL subprogram name
```

The `ON KEY#` statement allows any SFK to be used for program control. When an SFK is pressed during a program and an `ON KEY#` statement has been declared for it, the specified branching occurs if the specified priority is higher than the current system priority. The range of priority is 1 through 15.

```
270 ON KEY #18 GOTO 200
280 ON KEY #1,3 GOSUB 900      ! PRIORITY IS 3
290 ON KEY #8 CALL Output    ! CAN'T PASS PARAMETERS
```

```
OPTION BASE 1 or 0
```

The `OPTION BASE` statement allows the default lower bound of arrays to be specified as one rather than zero. `OPTION BASE 0` can be declared for documentation purposes since it is default at power on. The `OPTION BASE` statement must be before any `DIM`, `COM`, `REAL`, `SHORT`, and `INTEGER` statements.

```
OVERLAP
```

The `OVERLAP` statement sets the computer to the overlapped processing mode allowing computation and multiple I/O operations to occur simultaneously.

P

PAGE

The `PAGE` function is used with the `PRINT` statement and causes a form feed to be output. If directed to the CRT, the printout area is cleared. Up to 75 lines are searched for a top-of-form indicator on the internal printer.

```
20 PRINT A(*),PAGE,B(*)
30 PRINT PAGE,Title#
```

PAUSE

The `PAUSE` statement suspends program execution. The `PAUSE` statement can't be executed from the keyboard.

PI

The `PI` function returns the value of π . It equals 3.14159265360 to twelve place accuracy.

```
50 Area=PI*Radius^2
60 Circumference=2*PI*Radius
```

POS (in string expression, of string expression)

The `POS` function determines the position of a substring within a string and returns the character position of the first character of the second string within the first or 0 if it is not present.

```
80 IF POS(A#,B#)<20 THEN 400
90 PRINT POS(X#,"CHANGE")
```

PRINT [print list]

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

```
110 PRINT "A EQUALS";A,LIN(3),"B EQUALS";B,PAGE
120 PRINT A;TAB(10),B;TAB(25),C;      ! NEXT PRINT APPENDED
130 PRINT X;SPA(5),Y;SPA(5),Z
140 PRINT Array(*)
```

```

PRINT# file number; data list [, END]
PRINT# file number; END
PRINT# file number, defined-record number [; data list [, END] ]
PRINT# file number, defined record number; END

```

The PRINT# statement is used to record values onto the specified file. In serial access mode, recording starts at the beginning of the file or after the last data item accessed. In random access mode, with the defined record number specified, recording starts at the beginning of a defined record. The data list can include variables, constants and literals, separated by commas. END causes an EOF to be printed after the data. When the data list is omitted in random mode (4th syntax) an EOR is printed in that record.

```

160 PRINT #1;A,B(*),C#      ! SERIAL PRINT
170 PRINT #2,6;Data(I),END ! RANDOM PRINT - PRINT EOF
180 PRINT #5,11            ! PRINT EOR
181 PRINT #3;END           ! SERIAL MODE - PRINT EOF

```

```

PRINT USING line identifier [; print using list]
PRINT USING image format string expression [; print using list]

```

The PRINT USING statement allows the exact form of printed output to be determined by the image format string. The print using list can contain variables, array identifiers and expressions, separated by commas or semicolons; each item must correspond to an appropriate field specifier in the image format string. The line identifier must refer to an IMAGE statement.

```

200 PRINT USING 100;J,K,L,C#      ! IMAGE AT LINE 100
210 PRINT USING A#;C(*)          ! A# CONTAINS FORMAT STRING
220 PRINT USING "K,X,K,2X,DDD.DD,K";C#,"EARNED",X,"DOLLARS"

```

```

PRINT ALL IS select code

```

The PRINT ALL IS statement defines the standard print all device used when **PRN ALL** is latched.

```

240 PRINT ALL IS 6
250 PRINT ALL IS 2

```

```

PRINTER IS select code [, HP-IB device address] [, WIDTH number of characters per
line]

```

The PRINTER IS statement is used to define the standard printer for the system. The range of the WIDTH is 20 through 260 characters; 80 is default.

```

270 PRINTER IS 7,WIDTH(120)
280 PRINTER IS 6
281 PRINTER IS 8,2,WIDTH(160)

```


PROTECT file specifier, protect code

The **PROTECT** statement is used to guard a file against accidental erasure. The protect code is any valid string expression except one with a length of zero; only the first six characters are recognized, however.

```
290  PROTECT "WAGES",C#
300  PROTECT "DATA:F8","XXX"
```

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

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

```
320  X=PROUND(Y,6)
330  PRINT PROUND(Wages,-2)    ! ROUND TO SECOND DECIMAL PLACE
```

PURGE file specifier [, protect code]

The **PURGE** statement erases the specified file from the storage medium. The protect code is allowed only if the file was previously protected.

```
350  PURGE "1/2/77"
360  PURGE "X:F8","Key"
```

R

RAD

The `RAD` statement is used to set radian mode for all results and arguments of trigonometric functions. There are 2π radians in a circle.

RANDOMIZE [numeric expression]

The `RANDOMIZE` statement re-evaluates the random number seed. An integer value returns a zero when `RND` is used.

```
20  RANDOMIZE
30  RANDOMIZE PI
```

READ variable name [, variable name , ...]

The `READ` statement specifies variables for which values are to be assigned from a `DATA` statement. It can't be executed from the keyboard.

```
50  READ A,B,C(*),D(1,2),E#[5]
60  READ X,Y,Z(5,3,2),C#,D#(1,2)
```

READ# file number; variable list

READ# file number, defined-record number [; variable list]

The `READ#` statement retrieves values for variables from the specified file. In serial access mode, reading starts at the beginning of the file or after the last data item accessed. In random access mode, with the defined record number specified, reading starts at the beginning of the defined record. `READ#` can also be used to reposition the data pointer by omitting the variable list in random mode. The variables in the variable list must be separated by commas.

```
80  READ #3;A,B(*),C(1,2),D#[5]      ! SERIAL READ
90  READ #4,7;X,Y,Z(*),J#           ! RANDOM READ
91  READ #4,5                       ! REPOSITION POINTER
```

READY# select code

The `READY#` command is used to allow I/O to a device to resume after it was suspended because of an I/O error at that select code.

```
READY #6
READY #0
```

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

The REAL statement is used to dimension and reserve storage space for non-subscripted and array variables and declare them as full precision.

```
140 REAL Angle(-4:4),B,Data(4,4,4)
150 REAL X,Y(2,2,5),Z
```



REDIM array variable (subscripts) [, array variable (subscripts), ...]

The REDIM statement allows a new working size for an array to be defined. The total number of elements can't exceed that originally declared. The number of dimensions can't change.

```
170 REDIM Array(5,X,Y) !SUBSCRIPTS CAN BE NUMERIC EXPRESSIONS
180 REDIM X(-2:2),Y*(5,5)
```

REM [any combination of characters]

The REM statement allows insertion of non-executable remarks into the listing of a program to provide documentation and make the program easier to follow.

```
200 REM Anything can go in a remark statement ##(*^(*)#&(*
205 REM
210 REM This part of the program outputs the data
```

REN [beginning line number [, increment value]]

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

```
REN
REN 50
REN 20,5
```

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

The RENAME statement allows any file to be given a new name. The protect code is used only if the file was previously protected.

```
261 RENAME "JUN 1" TO "JUN 2"
262 RENAME "DATA 1:F8" TO "DATA 3"
```

RES

The `RES` function returns the result of the last numeric computation that was executed from the keyboard.

```
270 Y=RES^2
280 IF RES>5 THEN 400
```

RE-SAVE file specifier [, protect code] [, beginning line identifier [, ending line identifier]]

The `RE-SAVE` statement allows a program to be written into a file that had been created with `SAVE` without purging the file first. The protect code is used only if the file was previously protected. When no line identifiers are specified, the entire program is saved. When one line identifier is specified, the program is saved from that line to the end. When two line identifiers are specified, that block of lines is saved.

```
RE-SAVE "PROG1"
RE-SAVE "PROG2", "X", 99 ! PROTECTED FILE; START WITH LINE 99
RE-SAVE "EXAMPL", 100, 400
```

RE-STORE file specifier [, protect code]

The `RE-STORE` statement allows a program to be written into a file that had been created with `STORE` without purging the file first.

```
331 RE-STORE "CHRDEF:F8"
332 RE-STORE "DATA", "XXX" ! PROTECTED FILE
```

RESTORE [line identifier]

The `RESTORE` statement repositions the `DATA` pointer to the beginning of the specified `DATA` statement, or at the lowest numbered `DATA` statement in the current program segment if one isn't specified, so that the values can be reused. If the line specified isn't a `DATA` statement, the pointer is positioned at the first `DATA` statement following that line. It can't be executed from the keyboard.

```
340 RESTORE ! POSITION POINTER AT FIRST DATA STATEMENT
350 RESTORE Data1
```

RESUME INTERACTIVE

The `RESUME INTERACTIVE` statement re-enables live keyboard capability previously disabled with `SUSPEND INTERACTIVE`.

RETURN

RETURN numeric expression

RETURN string expression

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

```

370 RETURN                ! LAST LINE OF GOSUB SUBROUTINE
380 RETURN X^Y+Z          ! MULTIPLE-LINE DEF FN - NUMERIC
390 RETURN A#&B#         ! MULTIPLE-LINE DEF FN - STRING

```

REV\$ string expression

The REV\$ (reverse) function returns a string whose value is the value of the specified string with the order of the characters reversed.

```

401 PRINT POS(REV$(A#),CHR$(46)) ! LOOKS FOR LAST PERIOD IN A#
402 A#=REV$(B#)

```

REWIND [mass storage unit specifier]

The REWIND statement rewinds the tape to its beginning. It is ignored if the mass storage unit specifier does not specify a tape cartridge.

```

410 REWIND
420 REWIND ":F8" ! THIS WOULD BE IGNORED

```

RND

The RND function generates a pseudo random number greater than or equal to zero and less than one.

```

440 PRINT RND
450 Y=RND*RND

```

ROW array variable

The ROW function returns the number of rows (second subscript from right) in the working size of the specified numeric or string array.

```

470 IF ROW(Array)>5 THEN REDIM B(6,6)
480 PRINT ROW(X)

```

RPT\$ (string expression , number of repetitions)

The RPT\$ function causes the specified string expression to be repeated the specified number of times. The range of repetitions is 0 through 32767.

```
500 C#=RPT$(B#,5)      ! 5 REPETITIONS OF B#
510 PRINT LEN(RPT$(X#,N))
```

RUN [line identifier]

The RUN command is used to begin execution of a program at either the specified line or the lowest numbered line in memory. The specified line must be in the main program.

```
RUN
RUN 200
```

S

SAVE file specifier [, beginning line identifier [, ending line identifier]]

The SAVE statement lists and records all or some of program lines in memory into a data file. If one line identifier is specified, the program is saved from that line to the end. When two line identifiers are specified, that block of lines is saved.

```
20  SAVE "STRTRK"
30  SAVE "X:F8",200      ! SAVE LINE 200 ON
40  SAVE "ROUTIN",100,400 ! SAVE LINES 100 THROUGH 400
```

SCRATCH

Erases program lines and DATA pointers from memory.

SCRATCH A

Erases the entire memory.

SCRATCH C

Erases the values of all variables including those in common.

SCRATCH KEY [key number]

Erases one or all SFK typing aid definitions including pre-defined definitions.

SCRATCH P

Erases program, binary routines, variables and the files table from memory.

SCRATCH V

Erases the values of all variables except those in common.

SCRATCH Kn

Erases the typing aid definition of the SFK that is pressed.

SECURE [line identifier [, line identifier]]

The `SECURE` statement prevents selected lines or an entire program from being listed; an asterisk appears after the line number replacing the line in the listing. If one line identifier is specified only that line is secured. If two line identifiers are specified, that block of lines is secured.

```
SECURE          ! SECURES ENTIRE PROGRAM
SECURE 700      ! SECURES LINE 700
SECURE 90,190  ! SECURES LINES 90,190
```

SERIAL

The `SERIAL` statement cancels the effect of any previous `OVERLAP` statement and sets the computer to the serial processing mode. Computation and I/O do not occur simultaneously.

SGN numeric expression

The `SGN` (sign) function returns a 1 if the expression is positive, 0 if it is zero and -1 if it is negative.

```
100 IF SGN(A)=-1 THEN X=SQR(ABS(A))
110 PRINT SGN(G)
```

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

The **SHORT** statement is used to dimension and reserve storage space for simple and array variables and declare them as short precision.

```
130  SHORT H, J(5,2), K(2,2,2), L
140  SHORT A, B, C(2,2)
```

SIN numeric expression

The **SIN** function returns the sine of the angle which is represented by the numeric expression.

```
160  PRINT SIN(X)
170  A=SQR(1-SIN(X)^2)
```

SPA number of spaces

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

```
190  PRINT A; SPA(6), B
200  DISP "OUTPUT COMPLETE"; SPA(8), "WAITING FOR DATA"
```

SQR numeric expression

The **SQR** function returns the square root of a non-negative expression.

```
220  PRINT SQR(A(I, J))
230  Z=SQR(X+Y)
```

STANDARD

The **STANDARD** statement sets standard mode for output of numeric values.

STOP

The **STOP** statement terminates program execution and sets the program pointer to the lowest numbered line. It can't be executed from the keyboard.

STORE file specifier

The **STORE** statement is used to store all program lines and binary routines in memory into a program file on the specified mass storage device.

```
250 STORE "PAYROL"
260 STORE "FILE:F8"
```

STORE ALL file specifier

The **STORE ALL** statement stores into a special file the entire contents of user Read/Write Memory with the exception of the files table. It can't be executed from within a subprogram.

```
280 STORE ALL "MEMORY"
290 STORE ALL "2/3/77:F8"
```

STORE BIN file specifier

The **STORE BIN** statement stores into a special file all user binary programs in memory.

```
310 STORE BIN "ROUTIN"
320 STORE BIN "2"
```

STORE KEY file specifier

The **STORE KEY** statement stores all SFK typing aid definitions into a special key file.

```
340 STORE KEY "PROG1"
350 STORE KEY "AIDS:T"
```

SUB subprogram name [(formal parameter list)]

The **SUB** statement is the first line of a subroutine subprogram.

```
370 SUB Mean(Total, Number, Data(*), INTEGER X, SHORT G, #6, L#)
380 SUB Price(A, B#, C, D, E)
```

SUB END

The **SUB END** statement is the last line in a subroutine subprogram and transfers control back to the calling program.

SUB EXIT

The `SUB EXIT` statement is used to transfer control from a subroutine subprogram back to the calling program before `SUB END` is executed.

```
400 SUBEXIT
410 IF X>Y THEN SUBEXIT
```

SUM array name

The `SUM` function returns the sum of all the elements in a numeric array.

```
430 Total=SUM(Data)
440 PRINT SUM(A)
```

SUSPEND INTERACTIVE

The `SUSPEND INTERACTIVE` statement disables live, interactive keyboard operations while a program is running.

T

TAB character position

The `TAB` function is used with `PRINT` and `DISP` and causes the next item to be output beginning in the specified character position. The character position is a non-negative numeric expression and is rounded to an integer. If the value exceeds the number of columns in the standard printer, it is reduced by the formula: $\text{character position} \text{ MOD } N$, N being the number of columns specified as standard printer width. The item is output in the last column if the specified position is a multiple of the width and is reduced to 0 with the formula. If the specified position is already filled, a new line is generated and the item output in the specified character position.

```
20 PRINT C$;TAB(25),D$
30 DISP X;TAB(10),Y
```

TAN numeric expression

The `TAN` function returns the tangent of the angle which is represented by the expression.

```
50 A=TAN(X+Y)
60 PRINT TAN(45)
```

`TRACE [beginning line identifier [, ending line identifier]]`

The `TRACE` statement is used to trace program logic flow, in all or part of a program; any branching causes a trace output to be displayed which designates where the branching was from, and which line it was to. When one line identifier is specified, tracing begins after that line is executed. An ending line identifier causes tracing to stop after that line is executed.

```
80  TRACE                ! ALL LINES TRACED
90  TRACE 100            ! TRACE AFTER LINE 100
100 TRACE 100,250        ! TRACE BETWEEN 100 AND 250
```

`TRACE ALL`

The `TRACE ALL` statement traces all program logic flow and variable assignments. It is like executing both `TRACE` and `TRACE ALL VARIABLES`.

`TRACE ALL VARIABLES [beginning line identifier [, ending line identifier]]`

The `TRACE ALL VARIABLES` statement is used to monitor value changes of all variables in a specified program segment, or throughout the entire program. When one line identifier is specified, tracing begins after that line is executed. An ending line identifier causes tracing to stop after that line is executed.

```
120 TRACE ALL VARIABLES      ! TRACE DURING WHOLE PROGRAM
130 TRACE ALL VARIABLES 150  ! TRACE AFTER LINE 150
140 TRACE ALL VARIABLES 200,500 ! TRACE BETWEEN 200 AND 250
```

`TRACE PAUSE [line identifier [, numeric expression]]`

The `TRACE PAUSE` statement is used as a breakpoint, causing execution to halt before a specified line is executed a certain number of times. If no parameters are specified, execution stops and the next line to be executed is displayed. If just the line identifier is specified, execution stops at that line before it is executed. The numeric expression is rounded to an integer N. Execution stops at the line before it is executed the Nth time.

```
151 TRACE PAUSE           ! PAUSE HERE; DISPLAY NEXT LINE
160 TRACE PAUSE 150       ! PAUSE AT LINE 150
170 TRACE PAUSE Print,6   ! PAUSE AT LINE Print THE 6TH TIME
```

TRACE VARIABLES variable list

The **TRACE VARIABLES** statement is used to monitor value changes of selected variables; the trace output indicates the new value of the variable and in what line the assignment occurred. The variable list can contain 1-5 variables and array identifiers separated by commas.

```
190 TRACE VARIABLES A,B(*),C#
200 TRACE VARIABLES X,Y,H,J
```

TRACE WAIT number of milliseconds

The **TRACE WAIT** statement can be used with any selective **TRACE** statement, or **TRACE ALL**, and causes the computer to wait the specified amount of time after each line which causes a trace printout. The range of the numeric expression is -32768 through 32767; a negative number defaults to zero.

```
220 TRACE WAIT 2000      ! WAIT 2 SECONDS AFTER EACH TRACE
230 TRACE WAIT N
```

TRIM\$ string expression

The **TRIM\$** function deletes any leading or trailing blanks from the string expression.

```
250 A#=TRIM$(A#)
260 C#=TRIM$(H#)
```

TYP ([-] file number)

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

```
280 IF TYP(5)=6 THEN 400    ! BRANCH ON INTEGER PRECISION
290 PRINT TYP(4)
```

Value	Meaning
0	Option ROM missing or file pointer lost.
1	Full precision number
2	Total string
3	End-of-file mark
4	End-of-record mark
5	Integer precision number
6	Short precision number
7	Unused
8	First part of a string
9	Middle part of a string
10	Last part of a string

TYPEWRITER OFF

The `TYPEWRITER OFF` statement disables a previous `TYPEWRITER ON` statement and returns the keyboard to normal mode.

TYPEWRITER ON

The `TYPEWRITER ON` statement allows `TYPEWR` to be “pressed” from within a program and puts the keyboard into typewriter mode.

U

UPC\$ string expression

The `UPC$ (uppercase)` function returns a string with all lowercase letters converted to uppercase.

```
301  A#=UPC$(B#)
302  PRINT UPC$(C#&"names")
```

V

VAL string expression

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

```
310  A=VAL(A#)
320  X=VAL(X#)^2
```

VAL\$ numeric expression

The `VAL$` function returns a string representing the numeric expression in current output mode.

```
330  A#=VAL$(120)
340  PRINT VAL$(X)
```

W

`WAIT` number of milliseconds

The `WAIT` statement causes program execution to be delayed the approximate number of milliseconds before it continues. The range of the numeric expression is `-32768` through `32767`; a negative number defaults to zero.

```
360 WAIT 250
370 WAIT X
```

Mass Storage ROM

Most of the Mass Storage statements and commands are included in the mainframe. Additional statements that are enable by the Mass Storage ROM are listed here.

`FCREATE` file specifier , number of records

The `FCREATE` statement creates a binary data file of the specified length.

`FPRINT` file specifier [, protect code] , array identifier

The `FPRINT` statement stores the numeric or string array into the specified binary data file at DMA speeds.

`FREAD` file specifier [, protect code] , array identifier

The `FREAD` statement reads the specified array from a binary data file at DMA speeds.

`GSTORE` integer array identifier

Dumps DMA's graphics buffer to an integer array.

`GLOAD` integer array identifier

Dumps DMA's integer array to graphics buffer.



Graphics ROM

`AXES [Xtic spacing, Ytic spacing [, Xintersection, Yintersection [, Xmajor count, Ymajor count [, major-tic size]]]]`

The `AXES` statement draws a pair of axes with optional (linearly spaced) tic marks.

`CLIP [Xmin, Xmax, Ymin, Ymax]`

The `CLIP` statement defines the soft clip limits. Omitting the parameters allows any two diagonal corners to be digitized.

`CSIZE height [, aspect ratio]`

The `CSIZE` (character size) statement is used to specify the size and aspect ratio of characters used in labels. The height defaults to 15/4.54. The aspect ratio (width/height) defaults to 9/15.

`CURSOR Xvariable, Yvariable [, pen status string variable]`

The `CURSOR` statement returns values to the specified variables indicating the coordinate values of the cursor's location and the pen status. For the pen status, "0" indicates "up", "1" indicates down.

`DIGITIZE Xvariable, Yvariable [, pen status string variable]`

The `DIGITIZE` statement pauses program execution and allows you to reposition the cursor; execution is resumed by pressing the `CONTINUE` key, any `SFK` or the `STEP` key. The values of the cursor coordinates are assigned to the variables. Pen status is assigned to a string variable; "1" for down, "0" for up.

`DRAW Xco-ordinate, Yco-ordinate`

The `DRAW` statement drops the pen and moves it to the absolute X,Y coordinate position which is specified.

`DUMP GRAPHICS [lower bound[, upper bound]]`

The `DUMP GRAPHICS` statement copies the CRT graphic display to the internal thermal printer. Any horizontal area can be copied by specifying its upper and/or lower bound.

EXIT GRAPHICS

The `EXIT GRAPHICS` statement returns the CRT to normal mode from graphics mode.

FRAME

The `FRAME` statement draws a box around the current clipping area.

GCLEAR [distance]

The `GCLEAR` statement clears the CRT of previously plotted data. The distance value specifies how many millimetres of paper to eject on certain printers; it has no effect on CRT graphics.

GRAPHICS

The `GRAPHICS` statement sets the CRT to the graphics mode.

`GRID [Xtic spacing, Ytic spacing [, Xintersection, Yintersection [, Xmajor count, Ymajor count [, tic size]]]]`

The `GRID` statement can be used as an alternative to the `AXES` statement and is used to draw a full screen grid.

`I PLOT Xincrement, Yincrement [, pen control]`

The `I PLOT` statement allows incremental plotting from the last plotted point. The pen control is the same as for the `PLOT` statement.

LABEL list

The `LABEL` statement is used like the `PRINT` statement, and draws labels on the plotter. The label is terminated on an ASCII 3.

`LABEL USING image specifier; list`

The `LABEL USING` statement is used like the `PRINT USING` statement and draws formatted labels on the plotter.

`LDIR angle`

`LDIR Xcomponent, Ycomponent`

The `LDIR` statement specifies the angle at which subsequent labels will be drawn. The angle specifies counter-clockwise rotation of the label from the positive X-axis in current angular units. The angle specified by the second syntax is a vector plotted such that the Xcomponent equals the run and the Ycomponent equals the rise.

`LETTER`

The `LETTER` statement allows you to draw all keyboard alphanumerics by typing them in on the keyboard.

`LIMIT [Xmin, Xmax, Ymin, Ymax]`

The `LIMIT` statement defines the hard clip limits. The units are expressed in millimetres with the origin at the lower left physical limit. When the parameters aren't included, any two diagonal corner points can be digitized.

`LINETYPE id number [, length]`

The `LINETYPE` statement selects one of several solid or dashed line types. The range of the id number is 1 through 10; 4 is the default length.

`LOCATE [Xmin, Xmax, Ymin, Ymax]`

The `LOCATE` statement sets the area that `SHOW` will fill or `SCALE` will map to. The units are expressed in GDU's. Any two diagonal corner points can be digitized if the parameters are not included. `LOCATE` also invokes soft clipping at its boundary.

`LORG origin position`

The `LORG` (label origin) statement sets the label origin position which determines where any subsequent labels are drawn relative to the current pen location. The range of the origin position is 1 through 9.

`MOVE Xco-ordinate, Yco-ordinate`

The `MOVE` statement lifts the pen and moves it to the absolute X,Y coordinate position which is specified.

`MSCALE X, Y`

The `MSCALE` statement sets millimetres as user units and defines the origin. The origin is offset from the lower first `LOCATE` point (Xmin, Ymin) the specified amounts.

`PDIR` angle

`PDIR` Xcomponent, Ycomponent

The `PDIR` statement sets the angle of rotation for relative and incremental plotting. The first syntax specifies counter-clockwise rotation from the positive X-axis in current angular units. The second syntax indicates the angle by specifying a vector in which the Xcomponent equals the run and the Ycomponent equals the rise.

`PEN` pen number

The `PEN` statement specifies the pen to be used. 0 (zero) specifies return all pens to their holders on the 9872A. Negative pen numbers specify "erase" on the CRT.

`PENUP`

The `PENUP` statement lifts the pen.

`PLOT` Xco-ordinate, Yco-ordinate [, pen control]

The `PLOT` statement provides absolute data plotting and pen control. The pen control defaults to one and operates by the following conditions:

odd	drop pen
even	lift pen
positive	pen change after motion
negative	pen change before motion

`PLOTTER IS` [select code[, HP-IB device address],] plotter id string [, step size [, number of pens [, pen offset [, pen select id]]]]

The `PLOTTER IS` statement defines where all plotter operations will be directed. The three plotter id strings and their default select codes are –

"GRAPHICS"	(13)
"9872A"	(7,5)
"INCREMENTAL"	(5)

`PLOTTER` select code [, HP-IB device address] IS OFF

The `PLOTTER IS OFF` statement sets the specified device to a non-operative state.

`PLOTTER select code [, HP-IB device address] IS ON`

The `THE PLOTTER IS ON` statement declares the specified device to be the active plotter.

`POINTER Xcoordinate value, Ycoordinate value [, cursor type]`

The `POINTER` statement moves the cursor to the specified absolute position and can select one of two types of cursor. An even number specifies a small flashing cross; an odd number specifies full-screen crossed lines. Cursor type defaults to the large cursor (1).

`RPLOT Xrelative co-ordinate, Yrelative co-ordinate [, pen control]`

The `RPLOT` statement allows relative plotting from the last absolute plotted point which is used as the origin. The pen control is the same as for the `PLOT` statement.

`SCALE Xmin, Xmax, Ymin, Ymax`

The `SCALE` statement sets user definable units which are mapped onto the `LOCATE` rectangle.

`SETGU`

The `SETGU` statement sets graphic display units (GDU's) as the current units.

`SETUU`

The `SETUU` statement sets user defined units (UDU's) as the current units.

`SHOW Xmin, Xmax, Ymin, Ymax`

The `SHOW` statement defines an area that is stretched or shrunk equally in X,Y directions to fit into the plotting area defined by the `LOCATE` statement or by the default (`LIMIT`).

`UNCLIP`

The `UNCLIP` statement sets the soft clip limits equal to the hard clip limits.

`WHERE Xvariable, Yvariable [, pen status string variable]`

The `WHERE` statement returns the coordinate values of the last plotted or moved-to point.

Error Messages

1	Missing ROM or configuration error
2	Memory overflow
3	Line not found or not in current program segment
4	Improper return
5	Abnormal program termination
6	Improper FOR%NEXT matching
7	Undefined function or subroutine
8	Improper parameter matching
9	Improper number of parameters
10	String value required
11	Numeric value required
12	Attempt to redeclare variable
13	Array dimensions not specified
14	Multiple OPTION BASE statements or OPTION BASE statement preceded by variable declarative statements
15	Invalid bounds on array dimension or string length in memory allocation statement
16	Dimensions are improper or inconsistent
17	Subscript out of range
18	Substring out of range or string too long
19	Improper value
20	Integer precision overflow
21	Short precision overflow
22	Real precision overflow
23	Intermediate result overflow
24	TAN ($N \cdot \pi / 2$), when N is odd
25	Magnitude of argument of ASN or ACS is greater than 1

26	Zero to negative power
27	Negative base to non-integer power
28	LOG or LGT of negative number
29	LOG or LGT of zero
30	SQR of negative number
31	Division by zero
32	String does not represent valid number or string response when numeric data required
33	Improper argument for NUM, CHR\$, or RPT\$ function
34	Referenced line is not IMAGE statement
35	Improper format string
36	Out of DATA
37	EDIT string longer than 160 characters
38	I/O function not allowed
39	Function subprogram not allowed
40	Improper replace, delete or REN command
41	First line number greater than second
42	Attempt to replace or delete a busy line or subprogram
43	Matrix not square
44	Illegal operand in matrix transpose or matrix multiply
45	Nested keyboard entry statements
46	No binary in STORE BIN or no program in SAVE
47	Subprogram COM declaration is not consistent with main program
48	Recursion in single line function
49	Line specified in ON declaration not found
50	File number less than 1 or greater than 10
51	File not currently assigned

52	Improper mass storage unit specifier
53	Improper file name
54	Duplicate file name
55	Directory overflow
56	File name is undefined
57	Mass Storage ROM is missing
58	Improper file type
59	Physical or logical end-of-file found
60	Physical or logical end-of-record found in random mode
61	Defined record size is too small for data item
62	File is protected or wrong protect code specified
63	The number of physical records is greater than 32767
64	Medium overflow (out of user storage space)
65	Incorrect data type
66	Excessive rejected tracks during a mass storage initialization
67	Mass storage parameter less than or equal to 0
68	Invalid line number in GET or LINK operation
69 – 79	See Mass Storage ROM errors
80	Cartridge out or door open
81	Mass storage device failure
82	Mass storage device not present
83	Write protected
84	Record not found
85	Mass storage medium is not initialized
86	Not a compatible tape data cartridge
87	Record address error
88	Read data error

89	Check read error
90	Mass storage system error
91-99	See Mass Storage ROM errors
100	Item in print using list is string but image specifier is numeric
101	Item in print using list is numeric but image specifier is string
102	Numeric field specifier wider than printer width
103	Item in print using list has no corresponding image specifier
104-109	Unused
110-119	See Graphics ROM errors



SYSTEM ERROR

System Error octal number

These two errors indicate an error in the machine's firmware system; they are fatal errors. If reset does not bring control back, the machine must be turned off, then on again. If the problem persists, contact your Sales and Service Office.

Mass Storage ROM Errors

69	Format switch off
70	Not a disc interface
71	Disc interface power off
72	Incorrect controller address, or controller power off
73	Incorrect device type in mass storage unit specifier
74	Drive missing or power off
75	Disc system error
76	Incorrect unit code in mass storage unit*
77-79	Unused
91-99	Unused

Graphics ROM Errors

110	Plotter specifications not recognized.
111	Plotter not previously specified.
112	CRT Graphics hardware not installed.
113	LIMIT specifications out of range.
114-119	Unused

SALES & SERVICE OFFICES AFRICA, ASIA, AUSTRALIA

<p>ANGOLA Teleclra Empresa Técnica de Equipamentos Elétricos, S A R L R Barbosa Rodrigues 42-1 DT Caixa Postal 6487 Luanda Tel 35515/6 Cable TELECLTRA Luanda</p> <p>AUSTRALIA Hewlett Packard Australia Pty Ltd 31 41 Joseph Street Blackburn Victoria 3130 P O Box 36 Doncaster East Victoria 3109 Tel 49-6351 Telex 31-024 Cable HEWPARD Melbourne</p> <p>Hewlett Packard Australia Pty Ltd 31 Bridge Street Pymble New South Wales 2073 Tel 449-6566 Telex 21961 Cable HEWPARD Sydney</p> <p>Hewlett-Packard Australia Pty Ltd 153 Greenhill Road Parkdale S A 5063 Tel 272-5911 Telex 82536 ADEL Cable HEWPARD ADELAIDE</p> <p>Hewlett-Packard Australia Pty Ltd 141 Stirling Highway Nedlands W A 6009 Tel 86-5455 Telex 93859 PERTH Cable HEWPARD PERTH</p> <p>Hewlett-Packard Australia Pty Ltd 121 Wollongong Street Fyshwick A C T 2609 Tel 95-2733 Telex 62650 Canberra Cable HEWPARD CANBERRA</p> <p>Hewlett Packard Australia Pty Ltd 5th Floor Teachers Union Building 495-499 Boundary Street Spring Hill 4000 Queensland Tel 229-1544 Cable HEWPARD Brisbane</p> <p>GUAM Medical-Pocket Calculators Only Guam Medical Supply Inc Jay Ease Building, Room 210 P O Box 8947 Tamuning 96911 Tel 646-4513 Cable EARMED Guam</p>	<p>HONG KONG Schmidt & Co (Hong Kong) Ltd P O Box 297 39th Floor Connaught Road Central Hong Kong Tel H 255291 5 Telex 74766 SCHMC HX Cable SCHMIDTCD Hong Kong</p> <p>INDIA Blue Star Ltd Kasturi Buildings Jamshedji Tala Rd Bombay 400 020 Tel 29 50 21 Telex 2156 Cable BLUEFROST Blue Star Ltd Sahas 414 2 Vn Savarkar Marg Prabhadev Bombay 400 025 Tel 45 78 87 Telex 4093 Cable FROSTBLUE Blue Star Ltd Bano Box House Bombay 400 025 Tel 45 73 01 Telex 3751 Cable BLUESTAR Blue Star Ltd 14 40 Civil Lines Kanpur 208 001 Tel E 88 82 Telex 292 Cable BLUESTAR Blue Star Ltd 7 Hare Street P O Box 506 Calcutta 700 001 Tel 23 0131 Telex 7655 Cable BLUESTAR Blue Star Ltd 7th & 8th Floor Bhandari House 91 Nehru Place New Delhi 110024 Tel 634770 & 635166 Telex 2463 Cable BLUESTAR Blue Star Ltd Blue Star House 11 11A Magarath Road Bangalore 560 025 Tel 35668 Telex 430 Cable BLUESTAR Blue Star Ltd Meekshi Mandiran xx 1678 Mahatma Gandhi Rd Cochin 682 016 Tel 32069 32161 32282 Telex 046 514 Cable BLUESTAR</p>	<p>Blue Star Ltd 1-1-117 1 Mito Mitsu Building 105 Chome-1 San-no-maru Mito Ibaragi 310 Tel 0292-25-7470 Yokogawa-Hewlett-Packard Ltd Inoue Building 1348-3 Asahi-cho 1-chome Atsugi Kanagawa 243 Tel 0462-24-0452 Yokogawa-Hewlett-Packard Ltd Kimura Building 3rd Floor 20 2-chome Tsukuba Kumagaya Saitama 360 Tel 0485-24-6563</p> <p>KENYA Technical Engineering Services E A J Ltd P O Box 18311 Nairobi Tel 557226 556762 Cable PROTON Jakarta Tel 56038 40369 49886 Telex 42895 Cable BERACACON BERCA Indonesia P T 63 Jl Raya Gubeng Surabaya Tel 44309</p> <p>ISRAEL Electronics & Engineering Div of Motorola Israel Ltd 17 Kremenetski Street P O Box 25016 Tel-Aviv Tel 38973 Telex 33569 Cable BASTEL Tel-Aviv</p> <p>JAPAN Yokogawa-Hewlett-Packard Ltd Ohashi Building 59-1 Yoyogi 1-Chome Shibuya-ku, Tokyo 151 Tel 33-370-2281/92 Telex 232-2024/HP Cable YHPMARKET TDK 23-724 Yokogawa-Hewlett-Packard Ltd Seiko Ibaraki Building 2-8 Kasuga 2-chome, Ibaraki-shi Osaka 567 Tel (0726) 23-1641 Telex 5332-385 YHP OSAKA Yokogawa-Hewlett-Packard Ltd Nakano Building 24 Kami Sasaijima-cho Nakamura-ku Nagoya 450 Tel (052) 571-5171 Yokogawa-Hewlett-Packard Ltd Tangaya Building 2-24-1 Tsuruya-cho Kanagawa-ku Yokohama 221 Tel 045-312-1252 Telex 382-3204 YHP YOK</p>	<p>Yokogawa-Hewlett-Packard Ltd Mito Mitsu Building 105 Chome-1 San-no-maru Mito Ibaragi 310 Tel 0292-25-7470 Yokogawa-Hewlett-Packard Ltd Inoue Building 1348-3 Asahi-cho 1-chome Atsugi Kanagawa 243 Tel 0462-24-0452 Yokogawa-Hewlett-Packard Ltd Kimura Building 3rd Floor 20 2-chome Tsukuba Kumagaya Saitama 360 Tel 0485-24-6563</p> <p>KENYA Technical Engineering Services E A J Ltd P O Box 18311 Nairobi Tel 557226 556762 Cable PROTON Jakarta Tel 56038 40369 49886 Telex 42895 Cable BERACACON BERCA Indonesia P T 63 Jl Raya Gubeng Surabaya Tel 44309</p> <p>ISRAEL Electronics & Engineering Div of Motorola Israel Ltd 17 Kremenetski Street P O Box 25016 Tel-Aviv Tel 38973 Telex 33569 Cable BASTEL Tel-Aviv</p> <p>JAPAN Yokogawa-Hewlett-Packard Ltd Ohashi Building 59-1 Yoyogi 1-Chome Shibuya-ku, Tokyo 151 Tel 33-370-2281/92 Telex 232-2024/HP Cable YHPMARKET TDK 23-724 Yokogawa-Hewlett-Packard Ltd Seiko Ibaraki Building 2-8 Kasuga 2-chome, Ibaraki-shi Osaka 567 Tel (0726) 23-1641 Telex 5332-385 YHP OSAKA Yokogawa-Hewlett-Packard Ltd Nakano Building 24 Kami Sasaijima-cho Nakamura-ku Nagoya 450 Tel (052) 571-5171 Yokogawa-Hewlett-Packard Ltd Tangaya Building 2-24-1 Tsuruya-cho Kanagawa-ku Yokohama 221 Tel 045-312-1252 Telex 382-3204 YHP YOK</p>	<p>Hewlett-Packard (N.Z.) Ltd Pakuranga Professional Centre 267 Pakuranga Highway Pakuranga Box 51092 P O Box 1994 Tel 569-651 Telex NZ 3839 Cable HEWPACK Auckland Analytical/Medical Only Medical Supplies N Z Ltd Scientific Division 79 Carlton Gore Rd., Newmarket P O Box 1234 Auckland Tel 75-289 Telex 2958 MEDISUP Cable DENTAL Auckland</p> <p>PHILIPPINES The Online Advanced Systems Corporation 6th Floor, Yuju Co Bldg 560 Dumint Paredes St Binondo, Manila Tel 48-71-49 68-68-53 In Makati 85-35-81 85-34-91 Telex 3274 DNLINE</p> <p>RHODESIA Field Technical Sales 45 Kelvin Road North P O Box 3458 Salisbury Tel 705231 (5 lines) Telex RH 4122</p> <p>SINGAPORE Hewlett-Packard Singapore 11th Floor 3841F Jalan Bukit Merah Blk 2 6th Floor, Jalan Rendel Industrial Estate Alexandra P O Box 58 Singapore 3 Tel 633022 Telex HPSGS RS 21486 Cable HEWPACK, Singapore</p> <p>SOUTH AFRICA Hewlett-Packard South Africa (Pty.) Ltd Private Bag Wendywood Sandton, Transvaal 2144 Hewlett-Packard Centre Daphne Street, Wendywood. Sandton, Transvaal 2144 Tel 802-104018 Telex 8-4782 Cable HEWPACK JOHANNESBURG Service Department Hewlett-Packard South Africa (Pty.) Ltd P O Box 39325 Gramley, Sandton 2018 451 Wynberg Extension 3 Durban 2001 Tel 636-8188 9 Telex 8-2391 Hewlett-Packard South Africa (Pty.) Ltd P O Box 120 Howard Place, Cape Province, 7405 Pine Park Centre, Forest Drive, Pineblands Cape Province, 7405 Tel 53-7955 thu 9 Telex 57-0006 Service Department Hewlett-Packard South Africa (Pty.) Ltd P O Box 37099 Overport, Durban 4067 Brady House 641 Ridge Road Durban 4001 Tel 68-7478 Telex 6-7954</p>	<p>TAIWAN Hewlett-Packard Far East Ltd Taiwan Branch 39 Chung Shiao West Road Sec 1, 7th Floor Taipei Tel 381960-4 Telex 21824 HEWPACK Cable HEWPACK TAIPEI</p> <p>Hewlett-Packard Far East Ltd Taiwan Branch 68-2 Chung Cheng 3rd Road Kaohsiung Tel (07) 242318-Kaohsiung Analytical Only San Kwang Instruments Co. Ltd. P O Box 20, Yung Sui Road Taipei Tel 3715/71-4 (5 lines) Telex 532387 930338 Cable SANKWANG Cable SANKWANG TAIPEI</p> <p>TANZANIA Medical Only International Aeradio (E A.) Ltd P O Box 861 Dar es Salaam Tel 21251 Ext 265 Telex 4103</p> <p>THAILAND UNIMESA Co. Ltd. Eicom Research Building Bangkok Sukumvit Ave. Bangkok Tel 532387 930338 Cable UNIMESA Bangkok</p> <p>UGANDA Medical Only International Aeradio (E A.) Ltd. P O Box 2577 Kampala Tel 54388 Cable INTAERIO Kampala</p> <p>ZAMBIA R J Tiburye (Zambia) Ltd P O Box 2792 Lusaka Tel 73793 Cable ARJAYTEE Lusaka</p>
---	--	---	--	---	--

OTHER AREAS NOT LISTED. CONTACT:
Hewlett-Packard Intercontinental
3200 Hillview Ave
Palo Alto, California 94304
Tel (415) 493-1501
Telex 910-373-1267
Cable HEWPACK Palo Alto
Tel 034-8300, 034-8493

CANADA

<p>ALBERTA Hewlett-Packard (Canada) Ltd 11620A - 168 Street Edmonton T5M 3T9 Tel (403) 452-3670 TWX 610-831-2431 EDTH Hewlett-Packard (Canada) Ltd 915-42 Ave S E. Suite 102 Calgary T2G 1Z1 Tel (403) 287-1672 Twx 610-821-6441</p>	<p>BRITISH COLUMBIA Hewlett-Packard (Canada) Ltd 11620A - 168 Street Vancouver V6A 3R2 Tel (604) 254-0531 TWX 610-922-5059 VCR</p>	<p>MANITOBA Hewlett-Packard (Canada) Ltd 513 Century St St. James Winnipeg R3H 0L8 Tel (204) 786-7581 TWX 610-671-3531</p>	<p>NOVA SCOTIA Hewlett-Packard (Canada) Ltd 600 Windmill Road P O Box 9331 Dartmouth B2Y 3Z6 Tel (902) 469-7820 TWX 610-271-4482 HFX</p>	<p>ONTARIO Hewlett-Packard (Canada) Ltd 1020 Morrison Dr Ottawa K2H 8K7 Tel (613) 820-6483 TWX 610-513-1636 Hewlett-Packard (Canada) Ltd 6877 Goreway Drive Mississauga L4V 1M8 Tel (416) 678-9430 TWX 610-492-4246</p>	<p>QUEBEC Hewlett-Packard (Canada) Ltd 11620A - 168 Street Pointe Claire H9R 1G7 Tel (514) 697-4232 TWX 610-422-3022 TLX 05-821521 HPCL</p>	<p>FOR CANADIAN AREAS NOT LISTED: Contact Hewlett-Packard (Canada) Ltd in Mississauga</p>
---	--	--	--	--	---	--

CENTRAL AND SOUTH AMERICA

<p>ARGENTINA Hewlett-Packard Argentina S A Av Leandro N Alem 822 - 12 1001 Buenos Aires Tel 31 6963 4 5 6 and 7 Telex Puerto Booth N 9 Cable HEWPACK ARG</p> <p>BOLIVIA Stambuk & Mark (Bolivia) Ltda Av Mariscal, Santa Cruz 1342 La Paz Tel 40626, 53163, 52421 Telex 3560014 Cable BUKMAR</p> <p>BRAZIL Hewlett-Packard do Brasil I e C Ltda Avenida Rio Negro, 980 Alphaville 06400 Barueri Sao Paulo Tel 429-2148-9 429-2118-9 Hewlett-Packard do Brasil I e C Ltda Rua Padre Chagas, 32 90000 Porto Alegre RS Tel (0512) 22-2998, 22-5621 Cable HEWPACK porto Alegre</p>	<p>Hewlett-Packard do Brasil I e C Ltda Rua Siqueira Campos, 53 Copacabana 20000 Rio de Janeiro Tel 257-80-94-003 (02 1) Telex 391-2121905 HEWP-BR Cable HEWPACK Rio de Janeiro</p> <p>CHILE Calcagni y Metcalfe Ltda Alameda 580-Of 807 Casilla 2118 Santiago 1 Tel 39613 Telex 3520001 CALMET Cable CALMET Santiago</p> <p>COLOMBIA Instrumentación Henrik A. Langebaek & Kier S A Carrera 7 No 48-75 Apartado Aéreo 6287 Bogotá, I D E Tel 69-68-77 Cable AARIS Bogota Telex 044-400</p>	<p>COSTA RICA Centrica Costarricense S A Calle Central, Avenidas 1 y 3 Apartado 10159 San José Tel 21-86-13 Telex 391-2121905 HEWP-BR Cable GALGUR San José</p> <p>ECUADOR Medical Only A F Viscaino Compañía Ltda Av Rio Amazonas No 239 P D Box 2925 Quito Tel 242-150-247-033/034 Cable Astor Quito Calculators Only Computadoras y Equipos Electrónicos P D Box 2695 990 Toledo (y Cordero) Quito Tel 525-982 Telex 02-2113 Sagita Ed Cable Sagita-Quito</p>	<p>EL SALVADOR Instrumentación y Procesamiento Electrónico de el Salvador Bulevar de los Héroes II-48 San Salvador Tel 252787</p> <p>GUATEMALA IPESA Avenida La Reforma 3-48 Zona 9 Guatemala City Tel 63627, 64786 Telex 4192 Teletro Gu</p> <p>MEXICO Hewlett-Packard Mexicana S A de C V Torres Adalid No 21, 11 Piso Col. del Valle Mexico 12, D F Tel (905) 543-42-32 Telex 017-74-507</p>	<p>Hewlett-Packard Mexicana S A de C V Ave. Constitución No. 2184 Monterrey N L Tel 48-71-32, 48-71-84 Telex 038-843</p> <p>NICARAGUA Roberto Terán G Apartado Postal 689 Edificio Terán Managua Tel 25114, 23412, 23454 Cable ROTERAN Managua</p> <p>PANAMA Electrónico Balboa S A P O Box 4929 Calle Samuel Lewis Ciudad de Panama Tel 64-2700 Telex 3431103 Curunda. Canal Zone Cable ELECTRON Panama</p>	<p>PARAGUAY Z J Melamed S R L División Aparatos y Equipos Médicos División Aparatos y Equipos Científicos y de investigación Lima 1 Chile-482, Edificio Victoria Asunción Tel 91-271, 91-272 Cable RAMEL</p> <p>PERU Compañía Electro Médica S A Los Flamencos 145 San Isidro Casilla 1030 Lima 1 Tel 41-4325 Cable ELMED Lima</p> <p>PUERTO RICO Hewlett-Packard Inter-Americas Puerto Rico Branch Office Calle 272 No 203 Urb. Country Club Carolina 00924 Tel (809) 762-7255 Telex 345 0514</p>	<p>URUGUAY Pablo Ferrando S A Comercial e Industrial Avenida Italia 2877 Casilla de Correo 370 Montevideo Tel 40-3102 Cable RADIUM Montevideo</p> <p>VENEZUELA Hewlett-Packard de Venezuela C A P O Box 50933 Caracas 105 Los Ruices Norte 3a Transversal Edificio Segre Caracas 107 Tel 35-00-11 (20 lines) Telex 25146 HEWPACK Cable HEWPACK Caracas</p>	<p>FOR AREAS NOT LISTED. CONTACT: Hewlett-Packard Inter-Americas 3200 Hillview Ave Palo Alto, California 94304 Tel (415) 493-1501 Telex 910-373-1260 Cable HEWPACK Palo Alto Tel 034-8300, 034-8493</p>
---	--	---	---	---	--	--	--

