decsystem10

BASIC LANGUAGE

REFERENCE CARD

(Version 17B)

digital

CONVENTIONS

Abbreviated channel specifier	A channel specifier without the trailing comma or colon.
Arithmetic operators	The following symbols: + Addition, or positive number - Subtraction, or negative number * Multiplication / Division ↑ or ** Exponentiation
Channel specifier	A numeric formula N specifying a BASIC input/output channel. The formula is truncated to an integer which must be between 1 and 9, inclusive. For a sequential access file, the channel number is represented as #N or #N:, and for a random access file, as :N, or :N:. If the statement or function which contains the channel specifier is only applicable to sequential access files or to random access files, the leading # or : in the channel specifier can usually be omitted since it is not necessary.
Formula	Any legal combination of constants, variables, functions, and arithmetic operators.
Line number	One to five digits at the beginning of the line which serve to identify the information on the line as a statement.
List	A one-dimensional array.
Matrix	A variable which names a list or table.
Number	A positive or negative decimal quantity which is significant to about 8 digits.
Numeric formula	A formula having a numeric value. When an integer is required, as in a subscript calculation, the number represented by the formula is truncated to the nearest integer.
Numeric variable	A variable name composed of a single letter, or a single letter followed by a single digit, which names a numeric value or a collection of numeric values.
Numeric vector	A one dimensional array (list) containing numeric values. It is named by a numeric variable.
Relational operators	The following six symbols: = Equal to < Less than <= Less than or equal to > Greater than >= Greater than or equal to <> Not equal to Relational operators used in string context imply alphabetical order.

String	A sequence of characters each of which is a letter, digit, space, or some character other than a line terminator or carriage return.
String constant	A string enclosed in double quotes.
String formula	A formula having a string value.
String variable	A single letter followed by a dollar sign, or a single letter followed by a single digit and a dollar sign, which represents a string or a collection of strings.
String vector	A one-dimensional array (list) containing string values. It is named by a string variable.
Table	A two-dimensional array (only numeric tables are allowed in BASIC).
Variable	A name used to refer to stored data. The name of a numeric variable is either a single letter or a single letter followed by a single digit. The name of a string variable is similar to the name of a numeric variable, except that it has an appended dollar sign. Examples: A, A4, A$, A4$.

When a statement or command accepts a device name, filename, and/or extension, the following rules apply:

1) If the device is omitted, DSK: is assumed except as noted below (#6).
2) If the filename is given but the extension is omitted, .BAS is assumed.
3) If the filename is given and the extension consists only of a period, the null extension is assumed.
4) If the filename is omitted, the extension must also be omitted.
5) If both the filename and extension are omitted, the current filename and extension are used.
6) If the device is omitted and the specification is followed by three asterisks, the library device BAS: is assumed.

In the statement descriptions that follow, words that are represented in capital letters are part of the BASIC statement and must be typed by the user in the same order as they appear in the description.

Square brackets ([ ]) have been used to show that the user is to substitute his data for what is in the brackets. The square brackets are not typed by the user; they have been used only to make the statement easier to read. For example, the statement GO TO [line number] could be typed as GO TO 50.

The braces ({ }) have been used to designate that either item inside the braces may be used in the statement. For example, ON [formula], GO TO [line number] and ON [formula], THEN [line number] are equivalent. The braces are not typed by the user.

The parentheses appearing in statements are required and must be typed by the user.

BASIC Statements

CHAIN [string or string formula], [numeric formula]

Stops execution of the currently running program, retrieves the named program from the specified device, compiles the program, and begins execution. The string is the name of the program and the string formula's value is the name of the program. The program name is in the form ''device:filename.ext''. The filename is required. The numeric formula is the line number in the chained program at which execution is to begin. If the numeric formula is omitted, the program is started at the beginning. Note that information in core in the currently running program is completely erased.

Examples: 50 CHAIN MYPROG, 100

CHANGE $\begin{bmatrix}\text{string formula}\\ \text{or}\\ \text{numeric vector}\end{bmatrix}$ TO $\begin{bmatrix}\text{numeric vector}\\ \text{or}\\ \text{string variable}\end{bmatrix}$

Causes a string formula to be converted to a numeric vector, or a numeric vector to be converted to characters and placed in a location named by a string variable. In a string formula to numeric vector conversion, the zero element of the vector is set to the number of characters in the string. In a numeric vector to string variable conversion, the number of characters which are to be in the string must be given as the zero element of the numeric vector.

```
Examples: 40 CHANGE A$ TO A
          70 CHANGE V TO A$
```

DATA [data, data, . . .]

Supplies constant data (numbers or alphanumeric strings) for READ statements. This statement is required when one or more READ statements are used.

```
Examples: 20 READ A1, A2, A3
          70 DATA 54, -3, .25
```

DEF FNx (sequence of numeric variables) = [formula]

Defines a user-specified function. The name of the function must be three alphabetic characters, the first two of which are FN. The formula must fit on one line unless the user is defining a multiple-line function. In the latter case, the equals sign is omitted; the definition appears on the following lines and is terminated with a FNEND statement.

```
Examples: 30 DEF FNA(x) = EXP (x↑2) + 5
          50  DEF FNP (x, y)
          55 LET FNP = X
          60 IF Y< = X THEN 70
          65 LET FNP = Y
          70 FNEND
```

DIM [variable] (subscript), [variable] (subscript), . . .

Specifies the space to be allocated for a list or a table. If a list or table is not specified in a DIM statement, the default dimension for a list is 10 items and the default for a table is 10, 10 items.

```
Examples:   1 DIM A (5)
            5 DIM X (10, 20)
           10 DIM A$ (5), X (10, 20)
```

END

Indicates the end of the program and must be the statement with the highest line number. This statement is always required.

Example: 1000 END

FOR [numeric variable] = [formula$_1$] TO [formula$_2$] $\begin{Bmatrix}\text{STEP}\\ \text{BY}\end{Bmatrix}$ [formula$_3$]

Specifies a loop and must be used with a NEXT statement. The numeric variable is initially set equal to formula$_1$. Then, if the numeric variable is less than formula$_2$ (for a positive or zero formula$_3$) or greater than formula$_2$ (for a negative formula$_3$), the steps following the FOR statement are executed until the NEXT statement is reached. The numeric variable is then incremented by formula$_3$, and control transfers back to the beginning of the loop where the test against formula$_2$ is repeated. When the numeric variable is greater than or less than formula$_2$, as appropriate, control is transferred to the statement following the NEXT statement. If STEP (or BY) [formula$_3$] is omitted, the increment assumed is 1.

```
Examples:  20 FOR A = 0 TO 100 STEP 2
           25 PRINT SIN (A)
           30 NEXT A
           50 FOR X = N* 7 + Z TO 206 BY 2.5
           55 PRINT X
          100 FOR C = -2 TO 2
          150 PRINT SQR (C)
          200 NEXT C
          250 NEXT X
```

GOSUB [line number]

Enters a subroutine at the statement specified by line number. The subroutine is exited from by executing a RETURN statement. The GOSUB statement can appear at any point in the main program except within a multiple line DEF.

```
Examples:   45 GOSUB 100
           100 REM A STATEMENT IN THE SUBROUTINE
```

GO TO [line number]

Transfers control to, and continues execution at, the statement with the specified line number , instead of executing the next sequential statement following the GO TO statement.

```
Examples:  100 GO TO 20
```

IF [formula] [relational operator] [formula], $\left\{\begin{array}{c}\text{THEN}\\\text{GO TO}\end{array}\right\}$ [line number]

Transfers to the statement with the specified line number if the given relationship is true. If the relationship is not true, the next line in sequential order is executed. The comma is optional.

```
Examples:   50 IF X = 0, THEN 20
            70 IF SIN (A)< = B THEN 20
           100 IF N$ = M$, GO TO 999
```

INPUT [variable, variable, . . .]

Allows data to be entered from the user's terminal during the running of the program. This statement causes a question mark to be output to the user's terminal so that he can respond by typing in values for the requested variables.

```
Examples: 10 INPUT A, B, C$
```

LET [variable] = [formula] or LET [variable$_1$ = variable$_2$ = . . .] = [formula]

Assigns the value of the formula to the specified variable. The word LET may be omitted. Multiple assignments are allowed.

```
Examples: 50 LET W = X = Y = 3* B - 7
          80 A$ = "THIS IS A CHARACTER STRING"
```

MARGIN [numeric formula]

Allows the user to set a right margin for terminal output from 1 to 132 characters. The numeric formula specifies the right margin, in the number of characters from the left margin. Without this statement, the standard right margin is 72 characters. Since the Monitor recognizes the right margin for output as 72 characters, a SET TTY WIDTH n monitor command (n is the desired margin) must be given before a MARGIN statement with an argument greater than 72 will be effective. The right margin for input is always 142 characters and is not affected by MARGIN statements.

```
Examples: 10 MARGIN 75
```

MAT matrix_2 = matrix_1

Sets up $\underline{\text{matrix}}_2$ to be the same as $\underline{\text{matrix}}_1$ and the dimension given to $\underline{\text{matrix}}_2$ is that of $\underline{\text{matrix}}_1$. Sufficient space must be allocated for $\underline{\text{matrix}}_2$ before this statement is executed.

Examples: 30 MAT A = B

MAT matrix_3 = matrix_1 + matrix_2

Adds $\underline{\text{matrix}}_1$ to $\underline{\text{matrix}}_2$ to obtain $\underline{\text{matrix}}_3$. Matrix_1 and matrix_2 must have the same dimensions, which will be given to matrix_3, and sufficient space must be allocated for matrix_3 before this statement is executed.

Example: 20 MAT C = A + B

MAT matrix_3 = matrix_1 $-\text{matrix}_2$

Subtracts $\underline{\text{matrix}}_2$ from $\underline{\text{matrix}}_1$ to obtain $\underline{\text{matrix}}_3$. Matrix_1 and matrix_2 must have the same dimensions, which is given to matrix_3. Sufficient space must be allocated for matrix_3 before this statement is executed.

Example: 70 MAT C = B -A

MAT matrix_3 = matrix_1 * matrix_2

Multiplies $\underline{\text{matrix}}_1$ by $\underline{\text{matrix}}_2$ to obtain $\underline{\text{matrix}}_3$. The number of columns in matrix_1 must be equal to the number of rows in matrix_2. Sufficient space must be allocated for matrix_3 before this statement is executed.

Example: 60 MAT C = A* B

MAT matrix_2 = (K)* matrix_1

Multiplies each component of $\underline{\text{matrix}}_1$ by the numeric formula $\underline{K}$ in order to form the components of $\underline{\text{matrix}}_2$. The formula K must be enclosed in parenthesis.

Example: 40 MAT B = (10)* A

MAT matrix = CON

Sets up the specified $\underline{\text{matrix}}$ with all components equal to one.

Example: 30 MAT A = CON

MAT matrix = IDN

Sets up the specified $\underline{\text{matrix}}$ as an identity matrix.

Example: 20 MAT A = IDN

MAT INPUT vector

Inputs a $\underline{\text{vector}}$. After this statement is executed, the NUM function contains the number of items input.

Example: 40 MAT INPUT V

MAT matrix_2 = INV (matrix_1)

Inverts $\underline{\text{matrix}}_1$ in order to obtain $\underline{\text{matrix}}_2$. Matrix_1 must be a square matrix. After the execution of this statement, the DET function contains the determinant of matrix_1.

Example: 30 MAT B = INV (A)

MAT PRINT matrix_1, matrix_2, . . .

Types the indicated matrices on the user's terminal.

Example: 20 MAT PRINT A, B

MAT READ matrix_1, matrix_2, . . .

Reads the specified matrices from DATA statements. Matrices are read in row-column order (i.e., A (1,1), A (1,2), . . ., A (2,1), A (2, 2), . . .).

Example: 50 MAT READ A, B

MAT $matrix_2$ = TRN ($matrix_1$)

Transposes $matrix_1$ to obtain $matrix_2$ (i.e., if $matrix_1$ has n rows and m columns, $matrix_2$ will have m rows and n columns).

Example: 70 MAT B = TRN (A)

MAT matrix = ZER

Sets up the specified matrix with all components equal to zero.

Example: 30 MAT C = ZER

NEXT [numeric variable]

Indicates that the numeric variable is to be incremented by the step size specified in the associated FOR statement and that the end condition in the FOR statement is to be tested again. If the end condition is not satisfied, control is transferred to the statement following the FOR statement. The NEXT statement must be used with a FOR statement and the numeric variable must be represented in the same way as in the FOR statement.

Examples:
```
50 FOR B = 0 TO 100 STEP 5
55 PRINT B, SQR (B)
60 NEXT B
```

NOPAGE

Changes the mode so that output to the terminal is no longer divided into pages. Since this is the normal condition, the NOPAGE statement is needed only when changing the mode from PAGE mode. <PA> delimiters in PRINT statements work in either NOPAGE or PAGE mode.

Example: 100 NOPAGE

NOQUOTE

Sets the terminal to NOQUOTE mode. This means that on output strings are not enclosed in quotes by BASIC even if they contain delimiters (blanks, commas, or tabs); a leading blank is not output before strings and negative numbers; and a data item can be split across one or more lines. This is the default mode and is used when writing a text file as opposed to writing a data file.

Example: 50 NOQUOTE

ON [formula], $\left\{\begin{matrix}\text{GO TO}\\ \text{THEN}\end{matrix}\right\}$ [line number_1], [line number_2], . . .

Transfers control to the statement with line number_1 if the integer portion of the value of the formula is 1, with line number_2 if the value is 2, and so forth. If the integer portion is below 1 or larger than the number of line numbers listed, an error message is given. The comma after the formula is optional. The ON statement can contain any number of line numbers as long as they fit on one line.

Examples: 20 ON X GO TO 100, 200
50 ON X^2 -3 THEN 60, 80, 99, 99
75 ON A + B + C GO TO 55, 80, 90, 101

PAGE [numeric formula]

Divides the terminal output into pages with the specified number of lines. The numeric formula represents the page size. A PRINT statement containing <PA> does not interfere with paging except that it immediately starts a new page and sets the line count for the terminal page back to zero.

Examples: 10 PAGE 50

PRINT [sequence of formulas and format control characters]

Types out results of a computation, types out a message in the program, and/or types out a blank line (i.e., skips a line). The format control characters are commas, semicolons, and <PA>. A comma is a signal to move to the next print zone. A semicolon is a signal to print the items in a close packed form. If a <PA> appears, the item following is printed in the first print zone of the next page of output. If a TAB (n) appears, the terminal's print head is moved to column n, unless n is less than the current column. In this case, the TAB is ignored.

Examples:
```
10 PRINT ''VALUE'', ''SQUARE ROOT''
15 PRINT
20 PRINT A,, B$ (6)
```

PRINT USING [line number or string formula], [formula$_1$, formula$_2$, . . .]

Types the formulas according to the format specified by the image. An image is a string of characters describing the form of the output and the placement of the output on the output line. When a line number is used, the image is on the line specified and is in the form :string of characters. These characters are not enclosed in quotes. When a string formula is used, the image is the value of the string formula.

Image formatting - strings

A string field begins with an apostrophe and may be continued with as many L, R, C, or E characters as needed, but may not contain a mixture of these characters. L, R, and C fields left adjust, right adjust, and center the string in the field, respectively. The string is truncated on the right if it is longer than the field. The E field performs a left-adjust action and, in addition, extends the field if the string overflows.

Image formatting - Numeric

A numeric field begins with two number signs (#), two dollar signs ($), or two asterisks (*). It may be continued with either the leading character or with number signs. If the field contains a decimal point, the decimal point may be followed by four uparrows (↑) or circumflexes (∧) to specify that the E format is desired. If the field contains commas, all digits output on the left of the decimal point are separated into groups of threes by commas. (The commas must not appear between the first two number signs, dollar signs, or asterisks, or within the 4 uparrows or circumflexes in the field.) Normally, the sign of the number (a blank or a minus sign) is output just before the first digit, and the field must reserve a place for it, as well as for the digits and commas. However, if the field is followed immediately by a minus sign, the sign is output at the end of the field.

A field beginning with number signs and not containing a trailing minus sign need not reserve space for the sign of non-negative numbers. When a field begins with dollar signs, a dollar sign is placed into the position immediately before the first digit. When the field begins with asterisks, asterisks are output from the left to the position of the first digit. Outputting negative numbers to dollar sign or asterisk fields is legal only if the trailing minus sign is present.

Image formatting - printable characters

Characters that are not part of a field but appear in the image are output as they appear.

Examples: In the following examples, the symbol △ is used to indicate a single blank space. The statements

```
10  :△THE△'LLLL △OF△ ' △ IS△####, .
20  PRINT USING 10, ''SQRT'', ''B'', 5000
```
result in the following output:
```
 △THE△SQRT△△ OF△ B△IS△5,000.
```
The statements
```
90 A = 96.8457
100 PRINT USING 120, A, A, A, A, -A
120 :△##△##.△$$$$.$$△****.** -
```
result in the following output:
```
△96△97.△△ $96.85△**96.85-
```
The statements
```
1000 C = 13.145
1100 A $ = ''##.# ↑↑↑↑△△ ####.↑↑↑↑△##''
1200 PRING USING A$, C, C
```
result in the following output:
```
△1.3E+01△ △ △131.E-01△13
```

QUOTE

Sets the terminal to quote mode. This means that strings are enclosed in quotes by BASIC if they contain delimiters (blanks, commas, or tabs); a leading blank is output before strings and negative numbers; and a data item cannot be longer than the maximum amount of space available on a new line. This is used when writing a data file, rather than a text file. Its function is to preserve the integrity of the data items.

Example:
```
10 QUOTE
```

RANDOMIZE

Obtains a different set of random numbers for each run.

Examples:
```
 5 RANDOMIZE
10 FOR A = 1 TO 10
15 PRINT RND
20 NEXT A
```

READ [variable, variable, . . .]

Assigns the data (numeric or string) in DATA statements to the specified variables. This statement must be used with one or more DATA statements.

Examples:
```
10 READ A, B, C
90 DATA 10, 32, 15
```

REM

Inserts comment lines in the program. The line number of the comment can be referred to by other BASIC statements (e.g., GO TO) even though the comment line itself is not executed. Note that short comments can be placed on the same line with a BASIC statement (except an IMAGE statement) by placing an apostrophe after the statement and following the apostrophe with the comment.

Examples:
```
50 REM THIS IS THE BEGINNING
51 REM OF THE SECOND PASS
60 A = SQRT (B) 'CALCULATE RESULT
```

RESTORE

Allows data in DATA statements to be read more than once. This statement sets the data block pointers back to the beginning of the collection of data values. RESTORE $ restores only the string data block pointer and RESTORE % restores only the numeric data block pointer.

Examples:
```
10 FOR I = 1 TO 10
12 READ X
. . . . . . . . . .
25 NEXT I
. . . . . . . . . .
72 RESTORE
74 READ X
76 FOR I = 1 TO 10
```

RETURN

Indicates an exit line of a subroutine and directs BASIC to go to the statement following the last GOSUB from which it transferred.

Example:
```
312 RETURN
```

STOP

Stops the program and is equivalent to GO TO n where n is the line number of the END statement.

Example:
```
300 STOP
```

BASIC DATA FILE STATEMENTS

NOTE

Sequences of arguments are separated by commas or semicolons. For example, in the FILE statement each channel specifier - filename argument pair is separated from the next pair by a comma or semicolon.

FILE [sequence of [channel specifier] [filename argument]]

Assigns files to channels during program execution and establishes file types. This allows the user to change the assignment during the running of his program. The filename argument is a string formula in the form filename.ext type where filename.ext is the name of the file and type is 1) % for a random access numeric file, 2) $nnn for a random access string file where nnn is the number of characters in the longest string, or 3) blank for a sequential access file. The maximum value allowed for nnn is 132. If nnn is omitted and the file presently exists, the value with which the file was written is used. If nnn is omitted and the file does not exist, the value 34 is assumed.

Examples: FILE #1:TEST, :6:DATAM.F4$61

FILES [sequence of filename arguments]

Assigns files to channels before program execution. Channels are assigned consecutively to the arguments of all the FILES statements in the program before execution begins. If an argument is omitted, the channel for the missing argument is skipped. These statements are not used again during program execution. The filename arguments are as described in the FILE statement, except that they must be string constants without quotes.

Examples: FILES FIRST,, THIRD.DAT %

IF END [abbreviated channel specifier], $\left\{\begin{matrix}\text{THEN}\\\text{GO TO}\end{matrix}\right\}$ [line number]

Determines if there is any data in the file between the current position in the file and the end of the file. The comma preceding THEN or GO TO is optional.

Examples: IF END #3, THEN 3000

INPUT [channel specifier] [sequence of variables]

Inputs data from the sequential or random access file on the specified channel. This statement should not be used with line numbered sequential access files because if a line number is present, it is read as data. (READ statements are for line numbered files.) For a sequential access file, the variables can be string, numeric, or both. For a random access file, the variables can be string or numeric, but not both, since it is illegal to have a random access file which contains mixed data types.

Examples: INPUT :2,A$,Q$(4)

MARGIN [sequence of [channel specifier] [numeric formula]]

Sets the right output margins for the files on the specified channels to be the value of the respective numeric formulas. This statement is used only for sequential access files since there is no margin in a random access file.

Examples: MARGIN #2, 132

MARGIN ALL [numeric formula]

Sets the right output margin for the sequential access files on channels 1 through 9 to be the value of the numeric formula. This is a convenient way to set a margin for all sequential access files currently assigned to channels. It has no effect on random access files.

Examples: MARGIN ALL 132

NOPAGE [sequence of abbreviated channel specifiers]

Indicates that output to the sequential access files on the specified channels is not to be divided into pages. NOPAGE mode is assumed unless changed with a PAGE statement. This statement has no effect on random access files.

Examples: NOPAGE #1, 3, 2

NOPAGE ALL

Indicates that the output to the sequential access files on channels 1 through 9 is not to be divided into pages. This is a convenient way in which to set all sequential access files assigned to channels to NOPAGE mode. It has no effect on random access files.

Examples: NOPAGE ALL

NOQUOTE [sequence of abbreviated channel specifiers]

Places the sequential access files on the specified channels into NOQUOTE mode. This means that strings are not enclosed in quotes by BASIC even if they contain delimiters (blanks, tabs, or commas); a leading blank is not output before strings and negative numbers; and data items longer than the amount of space available on a line are split across more than one line. This is the default mode and the NOQUOTE statement is necessary only when changing from QUOTE to NOQUOTE mode.

Examples: NOQUOTE #6

NOQUOTE ALL

Places the sequential access files on channels 1 through 9 into NOQUOTE mode. This is a convenient way to set all sequential access files assigned to channels into noquote mode. It has no effect on random access files.

Examples: NOQUOTE ALL

PAGE [sequence of [channel specifier] [numeric formula]]

Sets the output page sizes for the sequential access files on the specified channels to be the values of the numeric formulas.

Examples: PAGE #1:60, #4:15

PAGE ALL [numeric formula]

Sets the output page size for the sequential access files on channel 1 through 9 to be the value of the numeric formula. This is a convenient way to set the page size for all sequential access files assigned to channels. It has no effect on random access files.

Examples: PAGE ALL 60

PRINT [channel specifier] [sequence of formulas and format control characters]

Outputs data to the file on the specified channel; this statement cannot be used with line numbered sequential access files since it does not write a line number at the beginning of a new line. (WRITE and WRITE USING statements are for line numbered files.) For non-line-numbered sequential access files, the formulas can be string, numeric, or both. For a random access file, the formulas can be string or numeric, but not both, because a random access file cannot contain mixed data types. Sequential access files created by PRINT statements are normally read by INPUT statements, since both the statements are for non-line-numbered files. Refer to the PRINT statement in the first section for descriptions of the format control characters.

Examples: PRINT #1, "THE RESULT IS"; SQRT (A) <PA>

PRINT [abbreviated channel specifier], USING [line number or string formula], [sequence of formulas]

PRINT USING [abbreviated channel specifier], [line number or string formula], [sequence of formulas]

Outputs data to a non-line-numbered sequential access file on the specified channel. (WRITE and WRITE USING statements are for line-numbered files.) The data is output according to the format specified by an image. When a line number is used, the image is on the line specified. When a string formula is used the image is the value of the string formula. With the first form of the statement, the comma preceding the word USING is optional. Refer to the PRINT USING statement in the previous section for a discussion of the image.

```
Examples: PRINT #2, USING 1000, A, B$ + C$, TIM
          PRINT USING #4, I$(8), F6$, N(0)
```

QUOTE [sequence of abbreviated channel specifiers]

Places the sequential access files on the specified channels into QUOTE mode. This means that strings containing delimiters (blanks, tabs, or commas) are enclosed in quotes by BASIC; a leading blank is output before string and negative numbers; and data items cannot be longer than the amount of space available on a line. This statement is used when creating data files as opposed to text files. Its function is to preserve the integrity of the data items.

```
Examples: QUOTE #6, #8
```

QUOTE ALL

Places the sequential access files on channels 1 through 9 into quote mode. This is a convenient way to set all sequential access files assigned to channels into quote mode. It has no effect on random access files.

```
Examples: QUOTE ALL
```

READ [channel specifier] [sequence of variables]

Inputs data from the file on the specified channel. This statement should not be used with a non-line-numbered sequential access file because it expects each line of data to begin with a line number, which it ignores. (The INPUT statement is for non-line-numbered files.) If a line number is not present, an error message is given. For a sequential access file, the variables can be string, numeric, or both. For a random access file, the variables can be string or numeric, but not both, because a random access file cannot contain mixed data types.

```
Examples: READ :4, W(8), L(1); L(2)
```

RESTORE [sequence of abbreviated channel specifiers]

Places a sequential access file in read mode or sets the record pointer for a random access file to the beginning of the file. It is only necessary to restore a sequential access file if it is presently in write mode.

```
Examples: RESTORE #1, #3
```

SCRATCH [sequence of abbreviated channel specifiers]

Erases a sequential access file and sets it in write mode, or erases a random access file and sets the record pointer to the beginning of the file.

```
Examples: SCRATCH #1, :4
```

SET [sequence of [channel specifier] [numeric formula]]

Moves the record pointer for random access files to the item in the file that is specified by the numeric formula. Items are numbered 1, 2, 3, ...

```
Examples: SET 1:1, 4:115
```

WRITE [channel specifier] [sequence of formulas]

Outputs data to the file on the specified channel. This statement should not be used with a non-line-numbered sequential access file because it begins each line of output with a line number and a tab. (PRINT and PRINT USING statements are for non-line-numbered sequential access files.) The first line is numbered 1000 and subsequent lines are incremented by 10. For a sequential access file, the formulas can be string, numeric, or both. For a random access file, the formulas can be string or numeric, but not both, because a random access file cannot contain mixed data types. Files created by WRITE statements are normally read by READ statements, since both of these statements are for line numbered files.

Examples: WRITE #6, SQRT (A+B(Q+8)) ↑ 2

WRITE [channel specifier], USING [line number or string formula], [sequence of formulas]

WRITE USING [channel specifier], [line number or string formula], [sequence of formulas]

Outputs data to a line numbered sequential access file on the specified channel. (PRINT and PRINT USING statements are for non-line-numbered sequential access files.) The data is output according to the format specified by an image. When a line number is used, the image is on the line specified. When a string formula is used, the image is the value of the string formula. With the first form of the statement, the comma following the channel specifier is optional. Refer to the PRINT USING statement in the previous section for a discussion of the image.

```
Examples: WRITE #2 USING 10, A1, A2, A3
          WRITE USING #4, "A = ###", A
```

BASIC Intrinsic Functions

ABS (numeric formula)

Finds the absolute value of the numeric formula.

ASC (one character or a 2- or 3-letter code)

Returns the equivalent ASCII decimal number for the specified argument.

ATN (numeric formula)

Finds the arctangent of the numeric formula.

CHR$ (numeric formula)

Truncates the numeric formula to an integer, interprets the integer as a decimal number, and converts it to its equivalent ASCII character.

CLOG (numeric formula)

Returns the logarithm to the base 10 of the numeric formula.

COS (numeric formula)

Finds the COSine of the numeric formula.

COT (numeric formula)

Finds the COTangent of the numeric formula.

DET

Equals the determinant of a matrix after the matrix is inverted. If an attempt is made to invert a matrix whose determinant is zero, a warning message is given, DET is set equal to zero, and execution continues.

EXP (numeric formula)

Raises e to the power of the numeric formula.

INSTR (numeric formula, string $formula_1$, string $formula_2$)

Searches within the string specified by string $formula_1$ for the substring specified by string $formula_2$ and returns the position number of the first character of the substring. The numeric formula specifies the character position in the string at which to begin the search for the substring. If the numeric formula is omitted, the search starts at the first character of the string.

INT (numeric formula)

Determines the greatest integer less than or equal to the numeric formula argument.

LEFT$ (string formula, numeric formula)

Returns a substring of the specified string formula. This substring begins at the leftmost character of the string formula and contains the number of characters specified by the numeric formula.

LEN (string formula)

Returns the number of characters in the specified string formula.

LN (numeric formula)

Returns the logarithm to the base E of the numeric formula.

LOC (abbreviated channel specifier)

Returns the number of the current record in the random access file on the specified channel. An error message is given if the file is not a random access file.

LOF (abbreviated channel specifier)

Returns the number of the last record in the random access file on the specified channel. An error message is given if the file is not a random access file.

LOG (numeric formula)

Same as LN.

LOG10

Same as CLOG.

LOGE

Same as LN.

MID$ (string formula, numeric $formula_1$, numeric $formula_2$)

Returns the substring of the string formula which starts at the character position specified by numeric $formula_1$ and contains the number of characters specified by numeric $formula_2$. If numeric $formula_2$ is omitted, the substring continues to the end of the string. If numeric $formula_2$ is greater than the number of characters remaining in the string, only the remaining characters are returned. This is not considered an error condition.

NUM

Equals the number of components of the vector after a MAT INPUT is executed.

RIGHT$ (string formula, numeric formula)

Returns a substring of the string formula. This substring ends at the rightmost character of the string formula and contains the number of characters specified by the numeric formula.

RND

Generates random numbers between 0 and 1. The same set is generated repeatedly for purposes of program testing and debugging. To generate a different set, use the RANDOMIZE statement.

SGN (numeric formula)

Assigns a value of 1 if the numeric formula is positive, 0 if the numeric formula is zero, and -1 if the numeric formula is negative.

SIN (numeric formula)

Returns the SINE of the numeric formula.

SPACE$ (numeric formula)

Returns a string of spaces. The numeric formula specifies the number of spaces in the string.

SQR (numeric formula)

Returns the square root of the numeric formula.

SQRT (numeric formula)

Same as SQR.

STR$ (numeric formula)

Returns the string representation (as a number) of the numeric formula.

TAN (numeric formula)

Returns the TANgent of the numeric formula.

TIM

Returns the elapsed CPU time in seconds since the start of program execution.

VAL (string formula)

Returns the actual number that the string formula represents.

EDIT AND CONTROL COMMANDS

NOTE

Any command root can be typed out in full or abbreviated to its first three letters. Intermediate abbreviations are not legal. For example, LISTNHREVERSE and LISNHREV are equivalent.

BYE

Exits from BASIC and initiates the logout procedure from the system.

CATALOG dev:

Lists on the terminal the filenames and extensions of the user's files contained on the specified device. Examples of devices are: disk (DSK:), DECtape (DTAn:), and the library device (BAS: or ***).

COPY dev_1 $:\text{file.ext}_1 > \text{dev}_2$ $:\text{file.ext}_2$

Copies file.ext_1 to dev_2 and gives it the name file.ext_2. If the device is not a disk (DSK:) or DECtape (DTAn:), the filename and extension can be omitted.

DELETE line number$_n$ - line number $_m$

Deletes the lines numbered n through m from the user's core. If -line number$_m$ is omitted, only one line (line number$_n$) is deleted. Multiple arguments are allowed and are separated by commas (e.g., DELETE 10, 20-310, 471).

GOODBYE

Equivalent to BYE.

HELP

Types helpful information on the terminal to assist the user.

KEY

Sets BASIC to accept input from the keyboard of the user's terminal. This mode is assumed unless changed with a TAPE command.

LENGTH

Prints the approximate number of characters in the source program.

LIST line number$_n$ - line number$_m$

Lists with heading lines numbered n through m of the program. If the second line number is omitted only one line (line number$_n$) is listed with heading. Multiple arguments are allowed and are separated by commas. If no arguments are given, the entire program is listed with heading.

LISTNH line number$_n$ - line number$_m$

Same as LIST, but with the heading suppressed.

LISTREVERSE

Lists the program in reverse order with heading.

LISTNHREVERSE

Lists the program in reverse order without heading.

MONITOR

Exits to the monitor. The contents of core are not lost. Type CONTINUE to return to BASIC.

NEW dev:file.ext

Erases the file currently in the user's core area. The new file name is established as the name of the program currently in core.

OLD dev:file.ext

Replaces the file currently in the user's core area with the specified program from the storage device. The specified file must be line-numbered.

QUEUE file.ext/switches, file.ext/switches, . . .

Queues the specified disk files so that they will be output to the line printer when it becomes available. The file name is required; the extension is not. This command accepts the following optional switches:

/n COPIES	- Prints n copies of the file. The maximum number of copies is 63. If this switch is omitted, one copy is printed.
/LIMITn	- Specifies the maximum number of line printer pages that can be printed. If this switch is omitted, 200 pages is the maximum.
/UNSAVE	– Immediately removes the file from the user's disk area. If this switch is omitted, the file is not removed.

RENAME dev:file.ext

Changes the name of the program currently in the user's core area to the specified filename.

REPLACE dev:file.ext

Replaces the specified file with the file currently in the user's core area. If the device is disk (DSK:) or DECtape (DTAn:), the file being replaced must be on the device or an error message occurs.

RESEQUENCE n, line number, k

Changes the numbers of the lines from the specified line number to the end of the file to n,n+k, . . . The value of the specified line number must not be greater than n. If the line number is omitted (the commas must still be used as delimiters), the line numbers are changed from the beginning of the file to n,n+k, . . . If K is omitted, the line numbers are changed to n, n+10, ...

RUN line number

Prints a heading, compiles the program currently in core and begins execution at the specified line number. If the line number is omitted, execution begins at the first line of the program.

RUNNH line number

Same as RUN but with the heading suppressed.

SAVE dev:file.ext

Saves the file currently in the user's core area on the specified device. An error message is given if an attempt is made to overwrite an existing file of the same name.

SCRATCH

Deletes all program statements from the user's core area.

SYSTEM

Exits to the Monitor. The contents of core are lost.

TAPE

Sets BASIC to accept input from the paper-tape reader attached to the user's terminal. If TAPE mode is set and the user wishes to input from the terminal, he must terminate each line of input by depressing the RETURN key followed by the LINEFEED key (normally only the RETURN key is necessary). The KEY command is used to return BASIC to the normal input from the keyboard mode until another TAPE command is issued.

UNSAVE dev:file.ext, dev:file.ext, . . .

Deletes the specified files from the named devices. If no arguments are specified, device DSK: and the name of the program currently in core are assumed.

WEAVE dev:file.ext

Weaves the specified file into the program in core. Both the file and the program in core must be line numbered. When a statement in core has the same line as a statement in the file, it is replaced by that statement from the file. Except when a line is replaced (because of identical line numbers) all the lines originally in core and in the specified file are in core at the end of execution of this command.

↑C ↑C

Stops a running program and returns to the BASIC command level. Any files open in the program are closed.

↑O

Suppresses typeout.

For additional information on the BASIC language, refer to the DECsystem-10 BASIC Conversational Language Manual, DEC-10-KJZE-D.

Order number DEC-10-XBRCA-A-D. For single copies of this card, contact a DEC sales office. Multiple copies may be ordered in lots of 25 from the Program Library, Digital Equipment Corporation, 146 Main Street, Maynard, Mass. 01754

Price $10 for 25

Created by

DECsystem-10 Software Documentation Group