BASIC STATEMENTS

ADD [C] (alpha variable, \{hh\} alpha variable)\)
Example: 100 ADDC (A$, 30)
Binary addition, with or without carry, of the two arguments.

AND (alpha variable, \{hh\} alpha variable)\)
Example: 100 AND (A$, B$)
'AND' the second argument into the first.

BIN (alpha variable) = expression
Example: 100 BIN (A$) = X + 4
Convert the integer value of the expression to a one-byte binary value.

BOOL \(h\) (alpha variable, \{hh\} alpha variable)\)
\((h = \text{hexdigit} \ 0 \ to \ 9, \ A \ to \ F)\)
Example: 100 BOOLE (A$, 3C)
Perform the Boolean operation specified, on the arguments given.

COM com element [, com element] .
Example: 10 COM X, Y$, A(3)
Reserve space for specified common variables.

COM CLEAR {scalar variable
| array designator}
Example: 10 COM CLEAR X
Change common variables to non-common or vice-versa.

CONVERT expression TO alpha variable, (image)
Example: 100 CONVERT X TO A$, (##)
Convert the expression to an alpha variable according to the specified image.

CONVERT alpha variable TO numeric variable
Example: 100 CONVERT A$ TO X
Convert an ASCII character number to a numeric value.

DATA d [, d] .
Example: 100 DATA "ABC", 3.41
Provide data constants for a READ statement.

DEFFN a(v) = expression
Example: 100 DEFFN F(X) = A$\times\text{EXP}(X)
Define a mathematical function of the variable v with an identifier a.

DEFFN integer [variable [, var] ...]
Example: 100 DEFFN0 (A, C$, B(3))
Define a special function key or subroutine entry point with argument passing capability.
DEFFN integer ["character string" ] HEX hh [hh . . ] . 
Example: 100 DEFFN ' 31 "PRINT": HEX (220D)

Define a character string to be provided when a special function key is pressed.

DIM dim element [, dim element] .
Example: 100 DIM A (3,2), BS(5)B

Reserve storage space in memory for the array or alpha variable specified.

END
Example: 900 END

Display free space available in memory; stop execution.

FOR y = expression TO expression [STEP expression]
Example: 100 FOR K = 1 TO 10 + 2

Specify the start of a repetitive loop.

GOSUB line no.
Example: 100 GOSUB 500

Transfer execution to the subroutine at the specified line.

GOSUB integer ["character string"
variable expression ] .
Example: 100 GOSUB ' 12 ("JOHN", 12,4,X)

Transfer execution to the subroutine specified.

GOTO line no.
Example: 100 GOTO 200

Transfer execution to the specified line number.

HEXPRINT alpha variable
alpha array desig ] .
Example: 100 HE X PRINT A$, 1, BS

Print the characters of the variable or array in hexadecimal form.

IF (alpha or numeric variable) THEN line no.
expression ] .
Example: 100 IF A < B THEN 40
100 IF STR$(A$, 1, 2) < HEX$(0802) THEN 500

Branch to specified line number if the relation is true.

PRINT [t] [print element ] [t print element] .
Example: 100 PRINT "X": X, T

Output the values specified on a printing or display unit in zoned (t=) or packed (t=) format.

PRINTUSING line no., [print elt t] .
Example: 100 PRINTUSING 101, A$, T (1), Q

Output values for variables in format of image (line no.) statement, t = , or ;

READ alpha variable [, variable] .
Example: 100 READ XS, B (3), 1, A$(1)

Assign values from DATA statement to the variables specified in the READ statement.

REM [text string]
Example: 100 REM THIS SUBROUTINE PERFORMS I/O OPERATIONS

Provide non-executable explanatory remarks in program text.

RETURN
Example: 100 RETURN

Mark end of subroutine; return processing to statement following last executed GOSUB or GOSUB'
RETURN CLEAR
  Example:  100 RETURN CLEAR
  Clear internal subroutine tables without executing a RETURN.

ROTATE(alpha variable, d)
  Example:  100 ROTATE (A$, 4)
  Rotate the bits of each character in the alpha variable d-bits to the left.

SELECT(line length)
  Example:  10 SELECT CO 215 (130)
  Select device address, line length, file no., pause, degrees, radians or grads.

STOP("character string")
  Example:  100 STOP "TOUCH CONTINUE TO PROCEED"
  Halt program execution and display the STOP line.

TRACE[OFF]
  Example:  300 TRACE
  Turn trace mode on (or off)

UNPACK(image) TO numeric variable
  Example:  100 UNPACK (SECRET) TO X, Y, Z
  Unpack data (PACKed with same image) from alpha variable and store it in numeric variable(s).

XOR(alpha variable, alpha variable)
  Example:  100 XOR (A$, 2F)
  Exclusive OR the second argument into the first.

BASIC FUNCTIONS

FNa(expression)
  Example:  200 PRINT X + FNF (2+B)
  Reference a function defined in a DEFFN statement with the same identifier (a).

HEX(hh [hh] ...)
  Example:  100 A$ = HEX (4243)
  Provide any eight-bit codes in a program.

LEN(alpha variable)
  Example:  100 X = LEN (A$)
  Determine the number of characters in an alpha variable.

NUM(alpha variable)
  Example:  100 N = NUM (K$)
  Count the number of ASCII characters in the specified alpha variable which represent a valid number.

MATH FUNCTIONS

SIN(expr)
  Example:  100 A = SIN(B)
  Find the sine of the expression.

COS(expr)
  Example:  100 A = COS (B+2)
  Find the cosine of the expression.

TAN(expr)
  Example:  100 A = TAN (B+C)
  Find the tangent of the expression.

ARCSIN(expr)
  Example:  100 A = ARCSIN (B)
  Find the arcsine of the expression.

ARCCOS(expr)
  Example:  100 A = ARCCOS (B)
  Find the arccosine of the expression.

ARCTAN(expr) or ATN(expr)
  Example:  100 A = ARCTAN (B+C)
  Find the arctangent of the expression.

EXP(expr)
  Example:  100 A = EXP (.33* (5-6))
  Find the value of $e^{x^{(x-n+1)}}$

INT(expr)
  Example:  100 A = INT (3.6)
  Find the greatest integer <= value of the expression.
NON-PROGRAMMABLE COMMANDS

CLEAR
V
N
Example: CLEAR P, 500
Clear all or part (P) of program text; all variables (V) or non-common variables (N).

LIST [S] [line no. [i, line no. [j]]]
Example: LIST S 100, 999
List all or part of program currently in memory.

RENUMB [line no. [i], line no. j, increment]
Example: RENUMBER 100, 10, 3
Renumber program lines; first line to renumber (no. 1), its new line number (no. 2) and increment.

RUN
Example: RUN 5
Initiate program execution [from specified line]

SORT STATEMENTS

All arguments are array designators; sub-arguments (s, n) refer to bytes; r, c = row, column. Arrays must be pre-defined in DIM or COM statements; a = array, a$s = an alpha array, w$s = work vector, v$s = locators (subscript array).

MAT CONVERT a$ TO a$b [s, n] TO A$b [s, n]
Example: 100 MAT CONVERT A(1) TO A$b (6, 8)
Convert elements of numeric array to elements of alpha array in sort format.

MAT COPY [-] a$s [s, n] TO [-] b$s [s, n]
Example: 100 MAT COPY A$ [X+Y, 100/X] TO -A$b [10, 20]
Transfer data from first array to second. Element boundaries are ignored.

MAT MERGE a$s [s, n] TO w1$s [s, n] w2$s [s, n] b$s [s, n]
Example: 100 MAT MERGE A$s TO W1$s, W2$s, B$s
Merge pre-sorted rows of a$s; build a list of pointers in the locator array.

MAT MOVE a$s [s, n] TO b$s [r, c] TO b$s [r, c] [s, n]
Example: 100 MAT MOVE A$s [1, 3] M TO B$s [1, 3]
Move data element by element from the first array (a$s) to the second (b$s) as specified by the subscripts of the locator array (v$s).

MAT SEARCH a$s [s, n], relation [STEP expression]
Example: 100 MAT SEARCH A$s [1, 5, <, 5] TO B$s [1, 5] STEP 5
Search the first array (a$s) for substrings which satisfy the specified relation; store their positions in the second array (b$s).

MAT SORT a$s [s, n] TO w1$s [s, n] w2$s [s, n]
Example: 100 MAT SORT A$s [1, 3] TO W1$s [1, 3], W2$s [1, 3]
Create a locator array (v$s) of subscripts (pointers) arranged according to the ascending order of the elements in the first alpha array a$s.

MAT STATEMENTS

a, b, c are numeric array names
a$s, b$s are alpha array names

MAT a = a + b
Example: 100 MAT A = A + D
Add two matrices.

MAT a = CON [expr 1, expr 2]
Example: 100 MAT A = CON(10)
Set all elements of the specified array equal to one; [redimension the array with the expressions].

MAT a = b
Example: 100 MAT A = B
Replace each element of array a with the corresponding element of array b.

MAT a = IDN [expr 1, expr 2]
Example: 100 MAT B = IDN (5, 5)
Set the specified (square) array equal to the identity matrix; [redimension it with the expressions].

MAT INPUT a$[expr 1, expr 2] b$[expr 1, expr 2] length]
Example: 100 MAT INPUT A$, B$ (2, 2, 2)
Accept values for array elements from keyboard (or other input-class device); [redimension].

MAT a = INV (b$, numeric scalar)
Example: 100 MAT B = INV(A), D
Invert a matrix [obtain determinant].

MAT a = b + c
Example: 100 MAT A = B + C
Store the product of two arrays in the first array (a).

MAT PRINT array name [t array name] [t]
Example: 100 MAT PRINT A$, B$, C$,
Print or display array elements row by row, in zoned (t=.) or packed (t=:) format.

MAT READ a[expr 1, expr 2] b$[expr 1, expr 2] length]
Example: 100 MAT READ A [B$ (2, 3), D$ (4, 6)]
Assign values from DATA statement(s) to all array elements; [redimension].

MAT REDIM a[expr 1, expr 2] b$[expr 1, expr 2] length]
Example: 100 MAT REDIM A (4, 5), B$ (2, 3, 4)
Redimension the specified arrays (to not more than the no. of bytes in the original array).

MAT a = (expr + b
Example: 100 MAT A = (5) + B
Perform scalar multiplication on array b; assign product to array a.

MAT a = b - c
Example: 100 MAT C = A - B
Subtract matrices of the same dimension.

MAT a = TRN (a)
Example: 100 MAT C = TRN (A)
Replace array c with the transpose of array a.

MAT a = ZER [expr 1, expr 2]
Example: 100 MAT A = ZER (F, T + 2)
Set all elements of specified array to zero; [redimension]
GENERAL I/O STATEMENTS

SGIO (comment) [#t
/xyy] (microcommand sequence, error/status register) [data buffer]

Examples: 100 $GIO WRITE/03B (6C01 4400 A206 8601, RS) B$(t )
10 DIM MS8
20 MS = HEX(6C0744008601)
100 $GIO (MS, RS)

Perform the I/O operation specified by the microcommand sequence; flag errors in the error/status register; use data buffer for input/output data transfer.

SIF ON [#t
/xyy] line no.
Example: 100 $SIF ON #3, 200

Test ready condition of I/O device (data-ready for input unit, device-ready for output unit); branch to specified line number if condition = ready.

STRAN (data block, table or list) [hh] [R] mask
Example: 100 $STRAN (X$, T$) ) 2$F

Translate all characters in data block according to table or list [mask bytes (AND with hh) or replace with list characters (R)].

SPACK [D = a F = a] data buffer FROM variable [, variable]...

format specifier
Examples: 90 D$1 = HEX(002C)
100 SPACK (D = D$1 ) B$1() FROM X( ), Y$S

Transfer data from the variable(s) and pack it in the data buffer according to the format specified.

SUNPACK [D = a F = a] data buffer TO variable [, variable]...

format specifier
Example: 100 $SUNPACK (F = F2$) B$2( ) TO M, P$, O( )

Separate data stored in the data buffer according to the format specified, convert and transfer it to the variables given.

PLOTTER STATEMENTS

PLOT [expr 0] < [expr 1], [expr 2],.

<table>
<thead>
<tr>
<th>U</th>
<th>D</th>
<th>S</th>
<th>C</th>
<th>R</th>
</tr>
</thead>
</table>

"literal string"

[plot arg]

Example: 100 PLOT < X, Y, "VALUE" >, < 40, 60, A$ >

Move plotting carrier ∆x (expr 1) and ∆y (expr 2) increments; plot [expr 0] times according to plot argument.

*U = pen up; D = pen down; S = set horizontal (expr 1) and vertical (expr 2) spaces;
C = set character size; R = reset to zero; literal string, avar = plot alphanumeric labels.

ABBREVIATIONS

<table>
<thead>
<tr>
<th>aarray</th>
<th>arg</th>
<th>avar</th>
<th>desig</th>
<th>elt</th>
<th>expr</th>
<th>f</th>
<th>hh</th>
<th>int</th>
<th>narray</th>
<th>nvar</th>
<th>var</th>
<th>xyy</th>
</tr>
</thead>
<tbody>
<tr>
<td>alpha array</td>
<td>argument</td>
<td>alpha variable</td>
<td>array descriptor (e.g., AI, A($))</td>
<td>array element</td>
<td>numeric expression</td>
<td>file number assigned with SELECT statement</td>
<td>hexadecimal code</td>
<td>integer</td>
<td>numeric array</td>
<td>numeric variable</td>
<td>variable (numeric or alpha scalar, numeric or alpha array, STR function (a var), literal in &quot;&quot;)</td>
<td>device address</td>
</tr>
</tbody>
</table>

Printed in U.S.A. 700-3500F 7-80-2M
TAPE CASSETTE UNITS

STATEMENTS

BACKSPACE
\[ \#f, \]
\[ /x\] \[ y\] \[ End \]
\[ \{ n\] \[ f\] \}

Example: 100 BACKSPACE 5 F
Backspace to beginning of file (BEG), n logical records or n files (nF).

DATALOAD
\[ \#f, \]
\[ /x\] \[ y\] \[ expr \]
\[ \{ variable \}[variable] \{variable\} \}

Example: 100 DATALOAD A, B1, CS
Locate a data file by name, or load a logical data record.

DATALOAD BT
\[ (N = expr) \]
\[ \#f, \]
\[ /x\] \[ y\] \[ expr \]
\[ \{ variable \}[variable] \{variable\} \}

Example: 100 DATALOAD BT (N = 100) #5, QS($) Load a block of N (100 or 256) bytes of data.

DATARESAVE
\[ \#f, \]
\[ /x\] \[ y\] \[ expr \]
\[ \{ variable \}[variable] \{variable\} \}

Example: 100 DATARESAVE A, BS
Rewrite a header or data record, in place.

DATASAVE
\[ \#f, \]
\[ /x\] \[ y\] \[ expr \]
\[ \{ variable \}[variable] \{variable\} \}

Example: 100 DATASAVE OPEN "FOXI"
Write a data header (OPEN ...), trailer (END) or record (variable ...).

DATASAVE BT [R] \{(N = expr ...)[H]!\}
\[ \#f, \]
\[ /x\] \[ y\] \[ expr \]
\[ \{ variable \}[variable] \{variable\} \}

Example: 100 DATASAVE BT R (N = 100, H/10C, AS($) Record [or resave (R)] an N-byte (100 or 256) data block or a header record [H].

IF END THEN line no.
Example: 100 IF END THEN 130
If end of file record is encountered, branch to the specified line number.

LOAD
\[ \#f, \]
\[ /x\] \[ y\] \[ expr \]
\[ \{ line no. \}[line no. 2] \]

Example: 100 LOAD "SAM" 400
Load a program file by name.

RECORD
\[ \#f, \]
\[ /x\] \[ y\] \[ expr \]
\[ \{ n\] \[ f\] \}

Example: 100 REWIND/10C
Rewind cassette [in indicated unit].

SKIP
\[ \#f, \]
\[ /x\] \[ y\] \[ expr \]
\[ \{ n\] \[ f\] \}

Example: 100 SKIP 2F
Skip n logical records, n files (nF), or to trailer record on data file (END).

COMMANDS

LOAD
\[ \#f, \]
\[ /x\] \[ y\] \[ expr \]
\[ \{ line no. \}[line no. 2] \]

Example: LOAD/10B, "PROGA"
Load a program file by name.

SAVE
\[ \#f, \]
\[ /x\] \[ y\] \[ expr \]
\[ \{ line no. \}[line no. 2] \]

Example: SAVE #2, "LINDY"
Record all or part of a program [by name]; [protect program (P)].

DISK UNITS

STATEMENTS

#f in disk statements can be an integer (1 to 6) or a numeric variable; sector addresses can be expressions or alpha variables.

COPY
\[ \#f, \]
\[ /x\] \[ y\] \[ expr \]
\[ \{ expr 1, expr 2 \}

Example: COPY FR(0, 1023)
Copy the sectors specified from one platter to another.

DATALOAD BA
\[ \#f, \]
\[ /x\] \[ y\] \[ expr \]
\[ \{ sector address \} \{ variable \} \{variable\} \}

Example: 100 DATALOAD BA F (20, L) AS($K)
Read one sector (including control information) into the specified array.

DATALOAD DA
\[ \#f, \]
\[ /x\] \[ y\] \[ expr \]
\[ \{ sector address \} \{ variable \} \{variable\} \}

Example: 100 DATALOAD DA F/320, (D,D) A, B
Read a logical record starting at the sector address and assign the values to the specified variables.

DATALOAD DC \[ \#f, \]
\[ /x\] \[ y\] \[ expr \]
\[ \{ variable \} \{variable\} \}

Example: 100 DATALOAD DC SI($) A, BS Read the next logical record from a cataloged disk file and assign the values to the variables.

DATASAVE BA
\[ \#f, \]
\[ /x\] \[ y\] \[ expr \]
\[ \{ sector address \} \{ variable \} \{variable\} \}

Example: 100 DATASAVE BAT#2, (O,Q) DS($) Write one sector from the array with no control information.

DATASAVE DA
\[ \#f, \]
\[ /x\] \[ y\] \[ expr \]
\[ \{ sector address \} \{ variable \} \{variable\} \}

Example: 100 DATASAVE DA F(2+K, L) J($K), K1 Write one logical record (one or more sectors) from variables starting at the sector address, or write a trailer record (END).

DATASAVE DC \[ \#f, \]
\[ /x\] \[ y\] \[ expr \]
\[ \{ variable \} \{variable\} \}

Example: 100 DATASAVE DC $/4, END Write one record containing all variables specified starting at the current file position, or write a trailer record (END).

DATASAVE DC CLOSE
\[ \#f, \]
\[ /x\] \[ y\] \[ expr \]
\[ \{ variable \} \{variable\} \}

Example: 100 DATASAVE DC CLOSE ALL Close (clear sector address data of) the specified file or all files.

DATASAVE DC OPEN
\[ \#f, \]
\[ /x\] \[ y\] \[ expr \]
\[ \{ name 1 \}, "name 2" \}

Example: 100 DATASAVE DC OPEN F$ 200, "FIL1"
Reserve space in the Catalog for a cataloged data file, or for a temporary work file outside the catalog area.
DBACKSPACE  
\[ \text{DBACKSPACE} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \]
Example:  100 DBACKSPACE #2, 5S
Backspace number of records or sectors (S), or to starting sector address of file (BEG).

DSKIP  
\[ \text{DSKIP} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \]
Example:  100 DSKIP 4+K
Skip the number of logical records [or sectors (S)] specified, or to the end of file [END].

IF END THEN  
\[ \text{IF END THEN} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \]
Example:  100 IF END THEN 700
Branch to line number if an end of file (trailer) was encountered on the last disk operation.

LIMITS  
\[ \text{LIMITS} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \]
Example:  100 LIMITS F "NONAME", A, B, C
Obtain starting address, ending address and number of sectors [or current sector address] of file.

LIST DC  
\[ \text{LIST DC} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \]
Example:  LIST DC F
List the contents of the catalog index.

LOAD DA  
\[ \text{LOAD DA} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \]
\[ \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \]
\[ \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \]
\[ \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \]
Example:  100 LOAD DA F (40, L) 400
Load and execute program from the specified starting sector address.

LOAD DC  
\[ \text{LOAD DC} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \]
Example:  100 LOAD DC R "PROG1" 100, 200
Load the specified cataloged program [or segment] into memory and execute it.

MOVE  
\[ \text{MOVE} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \]
Example:  100 MOVE FR
Compress catalog and move it to another platter.

MOVE END  
\[ \text{MOVE END} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \]
Example:  MOVE END F = 4799
Move the end of the catalog area to the sector specified by expression.

SCRATCH  
\[ \text{SCRATCH} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \]
Example:  100 SCRATCH F "WORKFILE"
Change the file status of the named disk files to 'scratched'.

SCRATCH DISK  
\[ \text{SCRATCH DISK} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \]
Example:  100 SCRATCH DISK #2, LS = 4, END = 1023
Reserve space on disk [for the Catalog Index (expr 1)] and the Catalog Area (expr 2 = last sector to be used).

VERIFY  
\[ \text{VERIFY} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \]
Example:  VERIFY R (0, 1023)
Verify data in specified range of sectors.

DISK UNITS COMMANDS

LOAD DA  
\[ \text{LOAD DA} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \]
Example:  LOAD DAF (24, D)
Load the program from the specified starting sector.

LOAD DC  
\[ \text{LOAD DC} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \]
Example:  LOAD DCFAS
Load DCF "LIST".
Load the specified cataloged program file.

SAVE DA  
\[ \text{SAVE DA} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \]
Example:  SAVE DAF$320, PIL, L1 100, 200
Save all or part of a program on disk, starting at the specified sector.

SAVE DC  
\[ \text{SAVE DC} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \]
Example:  SAVE DCR (100) #2, "CONVERT"/SAVE DCF ("old") "new"
Save all or part of a program on disk and catalog it.

PUNCHED TAPE READER AND TELETYPET® STATEMENTS

DATALOAD  
\[ \text{DATALOAD} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \]
\[ \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \]
\[ \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \]
\[ \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \]
Example:  (reader): 100 DATALOAD/618, A15(2), A2
(Teletype): 100 DATALOAD/41D, A15(2), X, STR(A5(2), 2)
Read values from punched tape and assign them to variables given.

DATALOAD BT  
\[ \text{DATALOAD BT} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \]
\[ \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \]
\[ \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \]
\[ \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \]
Example:  (reader): 100 DATALOAD BT (N = 200, L = 00, S = A$)/618, B$1
(Teletype): 100 DATALOAD BT (N = 3, L = FF, S = OD)/41D, A$
Read punched tape [backwards (R1)] and store values read in the specified variable or array. [Read N characters, ignore Leader code characters (L), stop when stop character (S) is encountered.]

LOAD  
\[ \text{LOAD} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \]
Example:  (reader): 100 LOAD #2
(Teletype): 100 LOAD/41D, 100
Load program [by line number] into memory and execute it (clears previous program).

COMMAND

LOAD  
\[ \text{LOAD} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \]
Example:  (reader): LOAD/618
(Teletype): LOAD/41D
Load program from punched tape, add it to program currently in memory.

CARD READERS STATEMENTS

DATALOAD  
\[ \text{DATALOAD} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \]
\[ \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \]
\[ \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \]
\[ \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \{ \text{expr} \} \]
Example:  100 DATALOAD #3, N(1), A$, N
Read card(s) and assign the values to variables.
**DATALOAD**

```
[N = expr] [.] [L = fhh [al var]] [.] [S = fhh [al var]] [#f] [al var] [al array design]
```

(mark sense
card reader)

Example: 100 DATALOAD BT (N = 40, L = FF, S = 99)/517, BS$  
Read (up to N) characters (ignore leader characters [L=1], stop at stop character [S=1]), store them in the alpha scalar or array.

**DATALOAD**

```
[N = 82] #f, [al array design]
```

(punched card
reader)

Example: 100 DATALOAD BT (N = 82)/629, BS()  
Read 80-character Hollerith card images, translate them to ASCII codes and store them in the array.

**DATASAVE**

```
[#f] [42E] [al variable]
```

(punched
card reader)

Example: 100 DATASAVE BT/42E, FS$  
Initiate reading of subsequent card in hopper into card reader buffer (used before DATALOAD,
DATALOAD BT statements).

**INPUT**

```
["character string",] variable [1, variable] .
```

Example: 100 INPUT "ENTER A AND B", A, B  
Read data values (separated by commas) from card(s) and assign them to variables in the INPUT
statement.

**LOAD**

```
[#f] [628] [line no. 1, line no. 2]
```

(punched
card reader)

Example: 100 LOAD/628, 100, 200  
Load a BASICA program from punched cards into memory with conversion from Hollerith to ASCII code.

**COMMANDS**

```
[#f] [628]
```

(punched
card reader)

Example: LOAD/628  
Load Hollerith program card text, convert it to ASCII and add it to the program in memory.

**ERROR CODES**

```
<table>
<thead>
<tr>
<th>CODE</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Text Overflow (insufficient memory for program text)</td>
</tr>
<tr>
<td>02</td>
<td>Table Overflow (insufficient memory for internal tables, subroutine and FOR/NEXT loops; non-common variables are cleared)</td>
</tr>
<tr>
<td>03</td>
<td>Math Error</td>
</tr>
<tr>
<td>04</td>
<td>Missing Left Parenthesis</td>
</tr>
<tr>
<td>05</td>
<td>Missing Right Parenthesis</td>
</tr>
<tr>
<td>06</td>
<td>Missing Equal Sign</td>
</tr>
<tr>
<td>07</td>
<td>Missing Quotation Marks &quot;&quot;</td>
</tr>
<tr>
<td>08</td>
<td>Undefined FN Function Encountered</td>
</tr>
<tr>
<td>09</td>
<td>More than five levels of FN Nesting Encountered</td>
</tr>
<tr>
<td>10</td>
<td>Incomplete Statement</td>
</tr>
<tr>
<td>11</td>
<td>Missing Line Number or Illegal Attempt to CONTINUE</td>
</tr>
<tr>
<td>12</td>
<td>Missing Statement Text</td>
</tr>
<tr>
<td>13</td>
<td>Missing or Illegal Integer</td>
</tr>
<tr>
<td>14</td>
<td>Missing Relational Operator</td>
</tr>
<tr>
<td>15</td>
<td>Missing Expression</td>
</tr>
<tr>
<td>16</td>
<td>Missing Scalar Variable</td>
</tr>
<tr>
<td>17</td>
<td>Missing Array Variable</td>
</tr>
<tr>
<td>18</td>
<td>Illegal Value</td>
</tr>
<tr>
<td>19</td>
<td>Missing Numeric Value</td>
</tr>
<tr>
<td>20</td>
<td>Illegal Format of Number</td>
</tr>
<tr>
<td>21</td>
<td>Missing Letter or Digit</td>
</tr>
<tr>
<td>22</td>
<td>Array Variable Not Defined</td>
</tr>
<tr>
<td>23</td>
<td>No Program Statements Found</td>
</tr>
<tr>
<td>24</td>
<td>Illegal Immediate Mode Statement</td>
</tr>
<tr>
<td>25</td>
<td>Illegal GOSUB/RETURN Use</td>
</tr>
<tr>
<td>26</td>
<td>Illegal FOR/NEXT Use</td>
</tr>
<tr>
<td>27</td>
<td>Insufficient Data for READ Statement</td>
</tr>
<tr>
<td>28</td>
<td>RESTORE Reference Out of Bounds</td>
</tr>
<tr>
<td>29</td>
<td>Illegal Data Format for INPUT Statement</td>
</tr>
<tr>
<td>30</td>
<td>Illegal Placement of COM Statement</td>
</tr>
<tr>
<td>31</td>
<td>Line Number &gt; 9999 Generated</td>
</tr>
<tr>
<td>32</td>
<td>Missing Hexdigit</td>
</tr>
<tr>
<td>33</td>
<td>Tape Cassette Read Error</td>
</tr>
<tr>
<td>34</td>
<td>Missing Comma or Semicolon</td>
</tr>
<tr>
<td>35</td>
<td>Format (#...) Missing From Image (%) Statement</td>
</tr>
<tr>
<td>36</td>
<td>Statement Referenced is not an Image Statement</td>
</tr>
<tr>
<td>37</td>
<td>Exponential Format Missing Four &quot;&quot;</td>
</tr>
<tr>
<td>38</td>
<td>Missing Literal in Quotes &quot;&quot;</td>
</tr>
<tr>
<td>39</td>
<td>Missing Alpha Variable</td>
</tr>
<tr>
<td>40</td>
<td>Illegal Argument in STRI Function</td>
</tr>
<tr>
<td>41</td>
<td>Program Name &gt; 8 Characters</td>
</tr>
<tr>
<td>42</td>
<td>Wrong Variable Type</td>
</tr>
<tr>
<td>43</td>
<td>Program Protected (cannot be SAVEd or LISTed)</td>
</tr>
<tr>
<td>44</td>
<td>Program Line &gt; 192 Keystrokes</td>
</tr>
<tr>
<td>45</td>
<td>Incorrect RENUMBER Use</td>
</tr>
<tr>
<td>46</td>
<td>#1 File Specification not SELECTed</td>
</tr>
<tr>
<td>47</td>
<td>Special Function Key Not Defined with DEFFN</td>
</tr>
<tr>
<td>48</td>
<td>End of Cassette Tape</td>
</tr>
<tr>
<td>49</td>
<td>Tape Cassette Tab Removed</td>
</tr>
<tr>
<td>50</td>
<td>Illegal Statement</td>
</tr>
<tr>
<td>51</td>
<td>Expected Data Record</td>
</tr>
<tr>
<td>52</td>
<td>Illegal Use of HEX Function</td>
</tr>
<tr>
<td>53</td>
<td>Illegal Argument in PLOT Statement</td>
</tr>
<tr>
<td>54</td>
<td>Illegal Argument in BT Statement</td>
</tr>
<tr>
<td>55</td>
<td>Too Few #'s in Image</td>
</tr>
<tr>
<td>56</td>
<td>Illegal Value (Disk Sector Address or MAT Array)</td>
</tr>
<tr>
<td>57</td>
<td>Expected Data Record</td>
</tr>
<tr>
<td>58</td>
<td>Illegal Alpha Variable</td>
</tr>
<tr>
<td>59</td>
<td>Array Too Small</td>
</tr>
<tr>
<td>60</td>
<td>Disk Timing Error</td>
</tr>
<tr>
<td>61</td>
<td>Disk File Full</td>
</tr>
<tr>
<td>62</td>
<td>Missing Alpha Array Designator</td>
</tr>
<tr>
<td>63</td>
<td>Sector Not on Disk or Disk Not Scratched</td>
</tr>
<tr>
<td>64</td>
<td>Disk Error (Medium or Hardware)</td>
</tr>
<tr>
<td>65</td>
<td>Disk Format Key Engaged</td>
</tr>
<tr>
<td>66</td>
<td>Disk Format Error</td>
</tr>
<tr>
<td>67</td>
<td>Disk CRC Error -- Probably Hardware</td>
</tr>
<tr>
<td>68</td>
<td>Sector Not Found -- Disk Seek Error (Hardware or Medium)</td>
</tr>
<tr>
<td>69</td>
<td>Disk CRC Error -- Usually Hardware, Possibly Medium</td>
</tr>
<tr>
<td>70</td>
<td>Incorrect Syntax Using File Name</td>
</tr>
<tr>
<td>71</td>
<td>Catalog End Error</td>
</tr>
<tr>
<td>72</td>
<td>Non-Programmable Command</td>
</tr>
<tr>
<td>73</td>
<td>PLOT Statement Angle Brackets &lt;&gt; Missing</td>
</tr>
<tr>
<td>74</td>
<td>Disk Starting Sector Address &lt;&gt; Ending Address</td>
</tr>
<tr>
<td>75</td>
<td>Disk File Not Scratched</td>
</tr>
<tr>
<td>76</td>
<td>Disk File Already Cataloged</td>
</tr>
<tr>
<td>77</td>
<td>Disk File Not in Catalog</td>
</tr>
<tr>
<td>78</td>
<td>Illegal Array Specification</td>
</tr>
<tr>
<td>79</td>
<td>No End-of-File on Disk</td>
</tr>
<tr>
<td>80</td>
<td>Disk Hardware Error -- Transient or Permanent</td>
</tr>
<tr>
<td>81</td>
<td>Not Enough Memory for MOVE or COPY on Disk</td>
</tr>
<tr>
<td>82</td>
<td>Disk Read after Write Error</td>
</tr>
<tr>
<td>83</td>
<td>Disk File Not Open</td>
</tr>
<tr>
<td>84</td>
<td>Common Variable Required for LOAD DA Statement</td>
</tr>
<tr>
<td>85</td>
<td>Disk Library Index Full</td>
</tr>
<tr>
<td>86</td>
<td>Matrix Not Square</td>
</tr>
<tr>
<td>87</td>
<td>Matrix Operands Not Compatible</td>
</tr>
<tr>
<td>88</td>
<td>Illegal Matrix operand</td>
</tr>
<tr>
<td>89</td>
<td>Redimensioned Matrix Too Large</td>
</tr>
<tr>
<td>90</td>
<td>Matrix is Singular</td>
</tr>
<tr>
<td>91</td>
<td>Asterisk Missing in Matrix Statement</td>
</tr>
<tr>
<td>92</td>
<td>Illegal Gen I/O Microcommand or Field/Delimiter Specification</td>
</tr>
<tr>
<td>93</td>
<td>Missing Gen I/O Buffer</td>
</tr>
<tr>
<td>94</td>
<td>Gen I/O Variable or Array Too Small</td>
</tr>
<tr>
<td>95</td>
<td>Illegal Gen I/O Arguments to Modify Array</td>
</tr>
<tr>
<td>96</td>
<td>Missing Sort Numeric Array Designator</td>
</tr>
<tr>
<td>97</td>
<td>Sort Array Too Small</td>
</tr>
<tr>
<td>98</td>
<td>Illegal Sort Array Dimensions</td>
</tr>
</tbody>
</table>

**SYSTEM ERROR!** Unrecoverable. Either machine failure or programmer fault (Run Mode Omitted).