

WANG

**WANG
BASIC-2
LANGUAGE**
Pocket Guide



2200

WANG

**WANG
BASIC-2
LANGUAGE**

Pocket Guide

© Wang Laboratories, Inc., 1980

HOW TO USE THIS POCKET GUIDE

This pocket guide contains the salient features of the BASIC-2 language available on Wang systems. For more information on BASIC-2, refer to the *Wang BASIC-2 Language Reference Manual* (700-4080D).

TABLE OF CONTENTS

NUMERIC FUNCTIONS

# PART	1
# PI	1
# TERM	1
ABS (exp)	1
ARCCOS (exp)	1
ARCSIN (exp)	1
ARCTAN (exp)	1
ATN (exp)	1
COS (exp)	1
ERR	1
EXP (exp)	1
FIX (exp)	1
INT (exp)	1
LGT (exp)	1
LOG (exp)	1
MAX (exp1, exp2, expn)	1
MIN (exp1, exp2, expn)	1
MOD (exp1, exp2)	1
RND (exp)	1

NUMERIC FUNCTIONS

ROUND (exp1, exp2)	1
SGN (exp)	2
SIN (exp)	2
SPACE	2
SPACEK	2
SQR (exp)	2
TAN (exp)	2

ALPHA FUNCTIONS AND LITERALS

ALL Function	3
BIN Function	3
HEX Literal	3
LEN Function	4
NUM Function	4
POS Function	4
\$PSTAT Function	5
STR Function	5
VAL Function	5
VER Function	6

ALPHA OPERATORS	7
------------------------------	---

STATEMENTS AND COMMANDS

\$BREAK	9
\$CLOSE	9
\$FORMAT	10
\$GIO	11
\$IF ON/OFF	12
\$INIT	12
\$MSG	13
\$OPEN	13
\$PACK	13
\$PSTAT	14
\$RELEASE PART	14
\$RELEASE TERMINAL	14
\$TRAN	15
\$UNPACK	15
COM	16
COM CLEAR	16
CONVERT	17
DATA	17
DEFFN	18
DEFFN' Keyboard Text Entry Definition	18
DEFFN' Subroutine Entry Point	18

STATEMENTS AND COMMANDS

DEFFN @PART	19
DIM	19
END	19
ERROR	20
FOR . . TO	20
GOSUB	20
GOSUB'	20
GOTO	21
HEXPACK	21
HEXUNPACK	21
IF . . THEN	22
IF END THEN	23
Image (%)	23
INPUT	23
KEYIN	24
LET	24
LINPUT	24
MAT=	25
MAT*	25
MAT()*	25
MAT-	25

STATEMENTS AND COMMANDS

MAT CON	26
MAT COPY	26
MAT IDN	26
MAT INPUT	26
MAT INV	27
MAT MERGE	27
MAT MOVE	28
MAT PRINT	29
MAT READ	29
MAT REDIM	29
MAT SEARCH	30
MAT SORT	31
MAT TRN	31
MAT ZER	32
NEXT	32
ON GOTO, ON GOSUB	32
ON/SELECT	33
PACK	33
PRINT	34
PRINTUSING	34
PRINTUSING TO	35

STATEMENTS AND COMMANDS

READ	35
REM	35
RESTORE	36
RETURN	36
RETURN CLEAR	36
ROTATE	37
SELECT	37
SELECT @ PART	39
STOP	39
UNPACK	39

COMMANDS AND KEYS

CLEAR	40
CONTINUE Key	40
HALT/STEP Key	40
LIST	41
LIST #	41
LIST'	42
LIST DT	42
LIST I	43

COMMANDS AND KEYS

LIST T	43
LIST V	44
RENUMBER	44
RESET Key	45
RUN	45
Special Function Key	45
TRACE	45

DISK STATEMENTS AND COMMANDS

\$ FORMAT DISK	46
COPY	46
DATALOAD BA	47
DATALOAD DA	47
DATALOAD DC	48
DATALOAD DC OPEN	48
DATASAVE BA	49
DATASAVE DA	50
DATASAVE DC	51
DATASAVE DC CLOSE	51
DATASAVE DC OPEN	52

DISK STATEMENTS AND COMMANDS

DBACKSPACE	53
DSKIP	53
LIMITS	54
LIST DC	55
LOAD (Command)	55
LOAD (Statement)	56
LOAD DA (Command)	57
LOAD DA (Statement)	58
LOAD RUN (Command)	59
MOVE	59
MOVE END	60
SAVE	60
SAVE DA	61
SCRATCH	62
SCRATCH DISK	62
VERIFY	63

BASIC-2 ERROR CODES (SUMMARY)

Miscellaneous Errors	64
Syntax Errors	64
Program Errors	65

RECOVERABLE ERRORS

Computational Errors	67
Execution Errors	67
Disk Errors	68
I/O Errors	68

GLOSSARY

alpha	=	alphanumeric
char	=	character
device-address	=	/taa, where t = device-type and aa = physical device-address
dim	=	dimension
elt	=	element
eof	=	end-of-file
exp	=	expression
fcn	=	function
file-number	=	#n, where n is an int or num-var with truncated val of 0-15
h	=	hexadecimal digit (0-9 or A-F)
hex	=	hexadecimal
int	=	integer
image-spec	=	$\left[\begin{array}{l} \text{alpha-var containing image} \\ \text{line-num of image statement} \\ \text{lit specifying image} \end{array} \right]$
literal	=	literal string
mat	=	matrix
num	=	numeric
O.S.	=	operating system
parm	=	parameter

GLOSSARY

print-elt	=	$\left[\begin{array}{l} \text{expression} \\ \text{alpha-variable} \\ \text{literal string} \end{array} \right]$
pd	=	packed decimal
ref	=	reference
spec	=	specification
str	=	string
val	=	value
var	=	variable

NUMERIC FUNCTIONS

#PART	Returns the partition number.
#PI	Returns 3.14159265359.
#TERM	Returns terminal number.
ABS(exp)	Finds the absolute val of exp.
ARCCOS(exp)	Finds arccosine of exp.
ARCSIN(exp)	Finds arcsine of exp.
ARCTAN(exp)	Finds arctangent of exp.
ATN(exp)	Finds arctangent of exp.
COS(exp)	Finds cosine of exp.
ERR	Returns error code of most recent error condition.
EXP(exp)	Finds e raised to power of exp (natural anti-log of exp).
FIX(exp)	Finds the int portion of exp.
INT(exp)	Finds the greatest int val of exp.
LGT(exp)	Finds common log (base 10) of exp.
LOG(exp)	Finds natural log (base e) of exp.
MAX(exp1,exp2,...expn)	Finds maximum val among exps or num arrays.
MIN(exp1,exp2,...expn)	Finds minimum val among exps or num arrays.
MOD(exp1,exp2)	Finds remainder of exp1/exp2.
RND(exp)	Produces random number between 0 and 1.
ROUND(exp1,exp2)	Finds the value of exp1 rounded to the exp2th decimal place if exp2 > 0, to nearest int if exp2=0, -(exp2)+1 to the left of decimal point if n < 0.

NUMERIC FUNCTIONS

SGN(exp)	Returns 1 if exp is positive, -1 if negative, 0 if zero.
SIN(exp)	Finds sine of exp.
SPACE	Returns amount of free space in user memory.
SPACEK	Returns total user memory size/1024.
SQR(exp)	Finds square root of exp.
TAN(exp)	Finds tangent of exp.

ALPHA FUNCTIONS AND LITERALSALL Function

$$\text{alpha-var} = [\dots] \text{ALL} \left(\begin{array}{c} \text{hh} \\ \text{alpha-var} \\ \text{literal} \end{array} \right) [\dots]$$

Defines a repeating str in which each char is as specified in the fcn.

BIN Function

$$\text{alpha-var} = [\dots] \text{BIN}(\text{exp} [,2]) [\dots]$$

Where:

0 < = val of exp < 256 if "2" omitted.

0 < = val of exp < 65536 if "2" included.

Converts int val of the exp to a one- or two-byte binary number.

HEX Literal

$$\text{HEX}(\text{hh} [\text{hh}\dots])$$

Permits the use of any eight-bit char code.

ALPHA FUNCTIONS AND LITERALSLEN Function

LEN (alpha-var)

Determines the number of chars up to the first trailing space in the val of an alpha-var.

NUM Function

NUM (alpha-var)

Determines the number of sequential ASCII chars in the specified alpha-var which represent a legal BASIC number.

POS Function

$$\text{POS} \left([-] \left\{ \begin{array}{l} \text{alpha-var} \\ \text{literal} \end{array} \right\} \left\{ \begin{array}{l} < \\ < = \\ = \\ > = \\ > \\ < > \end{array} \right\} \left\{ \begin{array}{l} \text{alpha-var} \\ \text{literal} \\ \text{hh} \end{array} \right\} \right)$$

Returns the position of the first (or last) char which satisfies specified relation.

ALPHA FUNCTIONS AND LITERALS\$PSTAT Function

\$PSTAT (exp)

Where:

1 <= val of exp <= number of partitions

Returns an alpha-str describing the current status of the partition specified by the exp.

STR Function

STR(alpha-var [,starting position][,length])

Defines a substr of an alpha-var.

VAL Function

VAL ({ alpha-var } [,2])

Converts the binary val of the first (or first two) byte(s) of alpha val to a num val.

ALPHA FUNCTIONS AND LITERALS

VER Function

$$\text{VER} \left(\left\{ \begin{array}{l} \text{alpha-var,} \\ \text{literal,} \end{array} \right\} \text{fmt-spec} \right)$$

Where:

$$\text{fmt-spec} = \left\{ \begin{array}{l} \text{alpha-var} \\ \text{literal made up of fmt-chars} \end{array} \right\}$$

$$\text{fmt-chars} = \left\{ \begin{array}{l} \text{A = alphabetic (A-Z, a-z)} \\ \text{\# = numeric (0-9)} \\ \text{N = alphanumeric (A-Z,a-z,0-9)} \\ \text{H = hexadecimal (0-9, A-F)} \\ \text{P = packed decimal} \\ \text{X = any character} \\ \text{other = only specified character} \end{array} \right\}$$

Verifies that the val of an alpha-var or literal conforms to fmt.

ALPHA OPERATORS

The following alpha operators, plus &, are only allowed in the assignment statement, as shown.

[LET] alpha-var [alpha-var]...= alpha-exp

$$\text{alpha-exp} = \left\{ \begin{array}{l} \text{alpha-operand [& alpha-operand]...} \\ \text{[alpha-operand] [alpha-operator alpha-operand]...} \end{array} \right\}$$

$$\text{alpha-operand} = \left\{ \begin{array}{l} \text{alpha-var} \\ \text{literal} \\ \text{STR function} \\ \text{BIN function} \\ \text{ALL function} \end{array} \right\}$$

ALPHA OPERATORS

alpha-operator =	}	ADD	Adds the bin vals of two alpha-exps.
		ADDC	Adds the bin vals of two alpha-exps with carry.
		AND	Logically ANDs two alpha-exps.
		BOOLh	Generalized logical operator.
		DAC	Performs decimal addition.
		DSC	Performs decimal subtraction.
		OR	Logically ORs two alpha-exps.
		SUB	Performs binary subtraction.
		SUBC	Performs binary subtraction with carry.
XOR	Exclusive ORs two alpha-exps.		

& Operator

[LET] alpha-var [alpha-var]... = alpha-exp & alpha-exp

Concatenation operator. Combines two strings into one, putting one after the other, without intervening characters. No other operators can be used in the same expression as &.

STATEMENTS AND COMMANDS**\$BREAK**

\$BREAK [exp
!]

Where:

0 <= exp < 256, default = 1

Relinquishes specified number of units of CPU processing time. ! stops processing.

\$CLOSE

\$CLOSE [{file-number
device-address} [, {file-number
device-address}] ...]

Releases the specified devices previously hogged via the \$OPEN statement.

STATEMENTS AND COMMANDS

\$FORMAT

\$FORMAT alpha-var = field-spec [,field-spec]...

Where:

field-spec =	{	SKIPxxx	(skip field)
		Fxxx	(ASCII free fmt)
		Ixxx[.dd]	(ASCII int fmt)
		Dxxx[.dd]	(IBM display fmt)
		Uxxx[.dd]	(IBM USASCII-8 fmt)
		P+xxx[.dd]	(IBM pd fmt)
		Pxxx[.dd]	(unsigned pd fmt)
		Axxx	(alpha fmt)

xxx = field width (0 < xxx < 256)

dd = implied decimal position (0 < = dd < 16)

Provides a mnemonic means of creating a fmt-spec for field form of \$PACK and \$UNPACK statements.

STATEMENTS AND COMMANDS

\$GIO

\$GIO [comment] [device-address[,]] [file-number,] (arg-1 [,arg-2]) [arg-3 [;arg-3...]]

Where:

- comment = A char str, ignored by system, identifying the particular operation (e.g., WRITE, READ, CHECK READY). Only uppercase letters, digits, and spaces.
- arg-1 = A customized microcommand sequence which defines the I/O operation:
 - a) Directly, by a hex lit, alpha lit, or a str of hexdigits, with each four-hexdigit code denoting one two-byte microcommand.
 - b) Indirectly, by an alpha-var containing the microcommand sequence.
- arg-2 = alpha-var whose individual bytes ("registers") are used for storage of special chars and error/status information. Must have at least 10 bytes.
- arg-3 = alpha-var used as the data buffer for multiple-char I/O operations. (Not required for single-char I/O operations.)

Used to perform I/O with non-standard devices.

STATEMENTS AND COMMANDS

\$IF ON/OFF

$$\$IF \left\{ \begin{array}{l} \text{ON} \\ \text{OFF} \end{array} \right\} \left[\begin{array}{l} \text{device-address,} \\ \text{file-number,} \end{array} \right] \text{line-number}$$

Determines the ready/busy status of any given device attached to the CPU and branches when ready or busy, depending on the form of the statement.

\$INIT

Program statement (pass configuration parms to the O.S.):

$$\$INIT (\text{alpha-1, alpha-2, alpha-3, alpha-4, alpha-5 } [\text{alpha-6}])$$

Immediate Mode statement (reconfigure system):

$$\$INIT \text{ password}$$

Where:

$$\text{alpha} = \left\{ \begin{array}{l} \text{lit-str} \\ \text{alpha-var} \end{array} \right\}$$

password = system reconfiguration password, which must be a literal one to eight chars in length.

Passes the system configuration parms to the MVP O.S. to allow partition generation to occur.

STATEMENTS AND COMMANDS

\$MSG

$$\$MSG = \text{alpha-exp}$$

Used by Terminal 1 to define a broadcast message to be displayed on each terminal whenever the READY message is normally displayed.

\$OPEN

$$\$OPEN \left[\text{line-number,} \right] \left\{ \begin{array}{l} \text{file-number} \\ \text{device-address} \end{array} \right\} \left[\left\{ \begin{array}{l} \text{file-number} \\ \text{device-address} \end{array} \right\} \right] \dots$$

Used to hog a peripheral for the current partition.

\$PACK

$$\$PACK \left(\left(\left\{ \begin{array}{l} \text{F} \\ \text{D} \end{array} \right\} = \left\{ \begin{array}{l} \text{alpha-var} \\ \text{literal} \end{array} \right\} \right) \right) \text{alpha-var FROM} \left\{ \begin{array}{l} \text{lit} \\ \text{exp} \\ \text{var} \\ \text{array} \end{array} \right\} \left[\left\{ \begin{array}{l} \text{lit} \\ \text{exp} \\ \text{var} \\ \text{array} \end{array} \right\} \right] \dots$$

Used to store data in a buffer in specified fmt.

STATEMENTS AND COMMANDS**\$PSTAT**

\$PSTAT = alpha-exp

Sets up user-defined portion of partition status (1st 8 bytes).

\$RELEASE PART

\$RELEASE PART

Causes a partition to be reassigned from the current controlling terminal to the null terminal (Terminal 0).

\$RELEASE TERMINAL

$$\$RELEASE\ TERMINAL\ \left[TO\ \left\{ \begin{array}{l} \text{exp} \\ \text{partition-name} \end{array} \right\} [, STOP] \right]$$

Where:

$$\text{partition-name} = \left\{ \begin{array}{l} \text{lit-str} \\ \text{alpha-var} \end{array} \right\} \text{ (1-8 bytes in length)}$$

Detaches the terminal from the current partition.

STATEMENTS AND COMMANDS**\$TRAN**

\$TRAN (arg-1, arg-2) [mask] [R]

Where:

arg-1 = alpha-var
 arg-2 = alpha-var or lit
 mask = two hex digits
 R = indicates which method is used to translate.

Translates the characters in arg-1 according to the translation table in arg-2.

\$UNPACK

$$\$UNPACK\ \left[\left(\left\{ \begin{array}{l} F \\ D \end{array} \right\} = \left\{ \begin{array}{l} \text{alpha-var} \\ \text{literal} \end{array} \right\} \right) \right] \text{ alpha-var TO } \left\{ \begin{array}{l} \text{var} \\ \text{array} \end{array} \right\} \left[, \left\{ \begin{array}{l} \text{var} \\ \text{array} \end{array} \right\} \right] \dots$$

Extracts data from a buffer and stores the data in vars.

STATEMENTS AND COMMANDS**COM**

COM com-elt [,com-elt]...

Where:

$$\text{com-elt} = \left\{ \begin{array}{l} \text{num-var} \\ \text{num-array-name (dim1[,dim2])} \\ \text{alpha-array-name (dim1[,dim2])[length]} \\ \text{alpha-var [length]} \end{array} \right\}$$

Define vars which will be used in common by several modules.

COM CLEAR

COM CLEAR $\left[\begin{array}{l} \text{var} \\ \text{array-designator} \end{array} \right]$

Redefines which variables are common.

STATEMENTS AND COMMANDS**CONVERT**

1. CONVERT alpha-var TO num-var
2. CONVERT exp TO alpha-var, (fmt)

Where:

$$\text{fmt} = \left\{ \begin{array}{l} \left[\begin{array}{l} + \\ - \end{array} \right] \left[\$ \right] \left[\# \right] \left[. \right] \left[] \left[\# \right] \left[] \left[\# \right] \left[] \left[\uparrow \uparrow \uparrow \right] \left[\begin{array}{l} + \\ - \\ ++ \\ -- \end{array} \right] \right\} \\ \text{alpha-var containing fmt} \end{array} \right.$$

Used to convert num val to ASCII alpha char str or vice versa.

DATA

DATA $\left\{ \begin{array}{l} \text{number} \\ \text{literal} \end{array} \right\} \left\{ \left[\begin{array}{l} \text{,number} \\ \text{,literal} \end{array} \right] \dots \right\}$

Supplies vals to be used by a READ statement.

STATEMENTS AND COMMANDSDEFFN

DEFFN a(num-var) = exp

Where:

a = identifies the fcn (digit or uppercase letter)

Defines a fcn of one var within a program.

DEFFN' Keyboard Text Entry Definition

DEFFN' int lit [;lit]...

Defines literal to be supplied for text entry when SF key is used.

DEFFN' Subroutine Entry Point

DEFFN' int [(var [,var]...)]

Marks a subroutine that can be called by an SF key or a GOSUB'.

STATEMENTS AND COMMANDSDEFFN @PART

DEFFN @PART $\left[\begin{array}{l} \text{[alpha-var]} \\ \text{[literal]} \end{array} \right]$ [FOR terminal# [, terminal#] ...]

Defines the current partition as "global," enabling the program text and global vars in the current partition to be shared with other partitions.

DIM

DIM dim-elt [,dim-elt]...

Where:

$$\text{dim-elt} = \left\{ \begin{array}{l} \text{num-var} \\ \text{num-array-name (dim1 [,dim2])} \\ \text{alpha-array-name (dim1 [,dim2])[length]} \\ \text{alpha-var [length]} \end{array} \right\}$$

Reserves space for noncommon vars.

END

END

Indicates the end of a BASIC program's job flow.

STATEMENTS AND COMMANDS**ERROR**

ERROR statement [:statement]...

Allows the programmer to respond to recoverable errors.

FOR...TO

FOR num-var = exp-1 TO exp-2 [STEP exp]

Initiates a loop ending with a NEXT statement.

GOSUB

GOSUB line-number

Transfers program execution to a subroutine.

GOSUB'

$$\text{GOSUB' int } \left[\left(\left\{ \begin{array}{l} \text{literal} \\ \text{alpha-var} \\ \text{num-exp} \end{array} \right\} \left[, \left\{ \begin{array}{l} \text{literal} \\ \text{alpha-var} \\ \text{num-exp} \end{array} \right\} \dots \right] \right) \right]$$

Transfers program execution to a marked subroutine and passes values.

STATEMENTS AND COMMANDS**GOTO**

GOTO line-number

Transfers program execution to a designated line.

HEXPACK

HEXPACK alpha-var-1 FROM alpha-var-2

Converts an ASCII char str of hex digits into the binary equivalent.

HEXUNPACK

HEXUNPACK alpha-var-1 TO alpha-var-2

Converts the binary val of alpha-var-1 to a str of ASCII hex chars which represent that val.

STATEMENTS AND COMMANDS**IF...THEN**

IF condition THEN $\left\{ \begin{array}{l} \text{line-number} \\ \text{statement} \end{array} \right\}$ [:ELSE statement]

Where:

condition = one or more relations separated by logical operators AND, OR, or XOR.

relation = operand $\left\{ \begin{array}{l} < \\ < = \\ = \\ > = \\ > \\ <> \end{array} \right\}$ operand

operand = $\left\{ \begin{array}{l} \text{num-exp} \\ \text{alpha-var} \\ \text{literal} \end{array} \right\}$

Executes the statement or branches to the specified line if the condition is true; executes the ELSE statement or goes to the following statement with no action if the condition is false.

STATEMENTS AND COMMANDS**IF END THEN**

IF END THEN $\left\{ \begin{array}{l} \text{line-number} \\ \text{statement} \end{array} \right\}$ [:ELSE statement]

Tests for an eof record.

Image (%)

%[char str] [fmt-spec] ...

Where:

fmt-spec = $\left[\begin{array}{l} + \\ - \end{array} \right]$ [\$] [#[,]...] [.] [#...] [↑↑↑↑] $\left[\begin{array}{l} + \\ - \\ ++ \\ -- \end{array} \right]$

Used with PRINTUSING to provide a fmt-spec for output.

INPUT

INPUT [lit [,]] var [, var]...

Allows the operator to supply data during program execution.

STATEMENTS AND COMMANDS**KEYIN**

1. KEYIN [device-address,] alpha-var [,line-number]
file-number,
2. KEYIN [device-address,] alpha-var,line-number,line-number
file-number,

Where:

file-number = #n, where n = an int or num-var.

1. Waits to receive a single char from an input device.
2. Gets a character from an input device if one is available.

LET

[LET] num-var [,num-var...]= exp
or
[LET] alpha-var [,alpha-var...]= alpha-exp

Assigns the val of the right hand exp to the var(s) on the left.

LINPUT

LINPUT [literal [,]] [?] [-] alpha-var

Allows input and concurrent editing of an alpha-var directly from the keyboard.

STATEMENTS AND COMMANDS**MAT =**

MAT c = a

Replaces each elt of c with the corresponding elt of a.

MAT*

MAT c = a * b

Multiplies mat a by mat b.

MAT()*

MAT c = (exp) * a

Multiplies each elt of array a by the val of exp.

MAT -

MAT c = a - b

Subtracts array b from array a.

STATEMENTS AND COMMANDS**MAT CON**

MAT c = CON [(dim1 [,dim2])]

Sets all elts of array equal to 1 and can redim the array.

MAT COPY

MAT COPY [-] source-alpha-var TO [-] output-alpha-var

Transfers an alpha val to an alpha-receiver one byte at a time.

MAT IDN

MAT c = IDN [(dim1,dim2)]

Causes the array to assume the form of the identity mat.

MAT INPUT

MAT INPUT array-name [(dim1[,dim2]) [length]] [...]

Allows the user to supply vals for array during program execution.

STATEMENTS AND COMMANDS**MAT INV**

MAT c = INV(a) [,d][,n]

Where:

d = a num-var with val of the determinant of a.

n = a num-var with val of the normalized determinant of a.

Causes mat c to be replaced by the inverse of mat a.

MAT MERGE

MAT MERGE merge-array[(x[,y])] TO control-var, work-var, loc-array

Where:

merge-array = 2-dimensional alpha-array. x,y define a field within each elt of the merge-array:

x = exp which specifies the starting position of field within each elt.

y = exp which specifies length of field in bytes. If exp omitted, field assumed to occupy remainder of elt.

control-var = alpha-var used to store merge status information.

work-var = alpha-var used by the system as work space.

loc-array = locator-array, alpha-array with two-byte elts to store subscripts.

Merges two or more sorted input files into a single sorted output file.

STATEMENTS AND COMMANDS

MAT MOVE

MAT MOVE move-array [,locator-array] [,n] TO receiver-array

Where:

move-array = { move-alpha-array-desig [(x[,y])] }
 move-num-array-desig }

locator-array = { locator-array-desig }
 locator-array-elt }

receiver-array = { receiver-alpha-array-elt [(x[,y])] }
 receiver-num-array-elt
 receiver-num-array-desig
 receiver-alpha-array-desig [(x[,y])] }

n = a num-var with the maximum number of elts to be moved.

(x,y) = optional field-designators defining a field within each alpha-array-elt such that:

x = exp specifying the starting position of the field.
 y = exp specifying the number of chars in the field (assumes remainder of elt if not specified).

Transfers data elt by elt from one array to another.

STATEMENTS AND COMMANDS

MAT PRINT

MAT PRINT array-name [{ { ; } } array-name] ... [{ ; }]

Prints arrays in the order given.

MAT READ

MAT READ array-name [(dim1 [,dim2]) [length]] [...]

Assigns vals contained in DATA statements to array-vars.

MAT REDIM

MAT REDIM array-name (dim1 [,dim2])[length] [...]

Redims the specified arrays.

STATEMENTS AND COMMANDS**MAT SEARCH**

$$\text{MAT SEARCH} \left\{ \begin{array}{l} \text{alpha-var} \\ \text{literal} \end{array} \right\} \left\{ \begin{array}{l} < \\ < = \\ = \\ > = \\ > \\ < > \end{array} \right\} \left\{ \begin{array}{l} \text{alpha-var} \\ \text{literal} \end{array} \right\} \text{ TO pointer-var [STEP s]}$$

Where:

$$s = \text{num-exp } 0 \leq s < 65536$$

Searches first alpha value for strings of the same length as second alpha value which satisfy the given relation. Starting positions of substrings placed in the pointer-var.

STATEMENTS AND COMMANDS**MAT SORT**

MAT SORT sort-array TO work-var, locator-array

Where:

sort-array	=	sort-array-desig[(x[,y])]
sort-array-desig	=	alpha-array-designator (e.g., A\$()) with data for sorting.
(x,y)	=	optional field-designators which define a field within each elt of the sort-array; x and y are exps such that: <ol style="list-style-type: none"> 1. x specifies the starting position of the field. 2. y specifies the number of chars in the field (or remainder of elt if y not specified).
work-var	=	alpha-var for temporary storage area.
locator-array	=	alpha-array with elts of length 2 used to contain subscripts of elts in the sort-array in sorted sequence.

Creates a locator-array containing subscripts arranged according to the ascending order of data vals in the sort-array.

MAT TRN

MAT c = TRN(a)

Causes mat c to be replaced by the transpose of a.

STATEMENTS AND COMMANDS**MAT ZER**

MAT c = ZER [(dim1 [,dim2])]

Sets all elts of the array equal to zero with optional redim.

NEXT

NEXT counter-var [counter-var]

Where:

counter-var = num-var used as counter in companion FOR statement.

Marks the end of a loop initiated by FOR.

ON GOTO**ON GOSUB**

ON $\left\{ \begin{array}{l} \text{alpha-var} \\ \text{exp} \end{array} \right\} \left\{ \begin{array}{l} \text{GOSUB} \\ \text{GOTO} \end{array} \right\} [,(line-num)]...line-num [:ELSE statement]$

Computed GOTO or GOSUB statement. Branches depending on the value of the exp or alpha-var.

STATEMENTS AND COMMANDS**ON/SELECT**

ON $\left\{ \begin{array}{l} \text{exp} \\ \text{alpha-var} \end{array} \right\}$ SELECT select-list [; [select-list]]

Where:

select-list = select-par [,select-par...]

SELECT statement where select-par assignment(s) made depend on the val of an exp or alpha-var.

PACK

PACK (image) alpha-var FROM $\left\{ \begin{array}{l} \text{num-array} \\ \text{exp} \end{array} \right\}$

Where:

image = $\left[\begin{array}{c} + \\ - \end{array} \right]$ [#]... [.] [#]... [↑↑↑↑] or alpha-var containing image

Packs num vals into an alpha-var or array.

STATEMENTS AND COMMANDS**PRINT**

PRINT [print-elt] $\left[\begin{array}{l} \{ \} \\ \{ ; \} \end{array} \right]$ [print-elt]...

Where:

$$\text{print-elt} = \left\{ \begin{array}{l} \text{exp} \\ \text{alpha-var} \\ \text{literal} \\ \text{AT()} \\ \text{BOX()} \\ \text{HEXOF()} \\ \text{TAB()} \end{array} \right\}$$

Sends output to the printer or CRT, as chosen by a SELECT statement.

PRINTUSING

PRINTUSING image-spec $\left[\begin{array}{l} \{ \} \\ \{ ; \} \end{array} \right]$ print-elt]...[:]

Sends formatted output to the printer or CRT.

STATEMENTS AND COMMANDS**PRINTUSING TO**

PRINTUSING TO alpha-var, image-spec $\left[\begin{array}{l} \{ \} \\ \{ ; \} \end{array} \right]$ print-elt]...[:]

Stores formatted print output in alpha-var.

READ

READ var [,var]...

Assigns elts listed in DATA statement to vars.

REM

REM [%] [] text string

Where:

text string = any chars except colon

Denotes comment to be ignored by system.

STATEMENTS AND COMMANDSRESTORE

$$\text{RESTORE} \left[\begin{array}{l} \text{LINE line-num [exp]} \\ \text{exp} \end{array} \right]$$

Where:

$$1 < = \text{exp} < 65536$$

Resets pointer in DATA statement back to specified data value to allow reuse by READ statement.

RETURN

RETURN

Indicates end of subroutine and causes execution to resume following last-executed GOSUB or GOSUB'.

RETURN CLEAR

RETURN CLEAR [ALL]

Used in subroutines to clear subroutine return address information from memory. Execution continues with following statement.

STATEMENTS AND COMMANDSROTATE

ROTATE [C] (alpha-var, exp)

Where:

$$-8 < = \text{exp} < 9$$

Rotates the val of the alpha-var or of each char the specified number of bits.

SELECT

This statement contains a number of parts, described separately below.

$$\text{SELECT} \left\{ \begin{array}{l} \text{R} \\ \text{D} \\ \text{G} \end{array} \right\}$$

Select radian, degree, or grad measure for trig functions.

SELECT ERROR [> error code]

Selects which math error causes program termination.

SELECT P [digit]

Select pause after console output.

STATEMENTS AND COMMANDS

SELECT LINE exp

Selects number of lines on the CRT.

SELECT {
 CI device-address
 INPUT device-address
 CO device-address [(width)]
 PRINT device-address [(width)]
 LIST device-address [(width)]
 PLOT device-address
 TAPE device-address
 DISK device-address
 file-number device-address
 }

Selects devices.

SELECT {
 ON [device-address [GOSUB line-number]]
 ON CLEAR
 OFF [device-address [GOSUB line-number]]
 }

Controls interrupts.

STATEMENTS AND COMMANDS**SELECT @PART**

SELECT [...] @PART partition-name [...]

Where:

partition-name = {
 alpha-var
 literal
 } (1-8 bytes in length)

Specifies a global partition whose text and/or global vars are to be referenced by the partition in which SELECT @PART is executed (the "calling" partition).

STOP

STOP [literal] [#]

Halts program execution until CONTINUE or HALT/STEP is keyed or a DEFFN' subroutine is invoked through an SF key being pressed.

UNPACK

UNPACK (image) alpha-var TO num-var [,num-var]...

Where:

image = [±] [#...] [.] [#...] [↑↑↑↑] or alpha-var containing image

Unpacks num data packed by a PACK statement.

COMMANDS AND KEYSCLEAR

$$\text{CLEAR} \left[\begin{array}{l} \text{P [line-number] [, [line-number]]} \\ \text{V} \\ \text{N} \end{array} \right]$$

Clears all program text and vars. With P clears only program text. With V clears only vars. With N clears only non-com vars.

CONTINUE Key

CONTINUE

Continues program execution after a program has been halted by a STOP statement or by keying HALT/STEP.

HALT/STEP Key

HALT/STEP Key

HALTs program execution or a listing operation or STEPs through program execution statement by statement.

COMMANDS AND KEYSLIST

LIST [S] [title] [D] [start line-number] [, [end line-number]]

Where:

$$\text{title} = \left\{ \begin{array}{l} \text{literal} \\ \text{alpha-var} \end{array} \right\}$$

Lists the specified portion of a program.

LIST#

LIST [S] [title] # [line-number] [, [line-number]]

Where:

$$\text{title} = \left\{ \begin{array}{l} \text{literal} \\ \text{alpha-var} \end{array} \right\}$$

Produces cross-ref listing of all refs to the specified line-numbers within the current program.

COMMANDS AND KEYSLIST'

LIST [S] [title] ' [int]

Where:

$$\text{title} = \left\{ \begin{array}{l} \text{literal} \\ \text{alpha-var} \end{array} \right\}$$

Creates a cross-ref listing for specified DEFFN' subroutines in the current program.

LIST DT

LIST [S] [title] DT

Where:

$$\text{title} = \left\{ \begin{array}{l} \text{literal} \\ \text{alpha-var} \end{array} \right\}$$

Displays the contents of the Device Table in hex.

COMMANDS AND KEYSLIST I

LIST [S] [title] I

Where:

$$\text{title} = \left\{ \begin{array}{l} \text{literal} \\ \text{alpha-var} \end{array} \right\}$$

Lists the current contents of the Interrupt Table.

LIST TLIST [S] [title] T $\left\{ \begin{array}{l} \text{literal} \\ \text{alpha-var} \end{array} \right\} \left[\begin{array}{l} \text{literal} \\ \text{alpha-var} \end{array} \right] \dots$

Where:

$$\text{title} = \left\{ \begin{array}{l} \text{literal} \\ \text{alpha-var} \end{array} \right\}$$

Generates a cross-ref listing of all program lines that contain a specified str.

COMMANDS AND KEYS**LIST V**

LIST [S] [title] V [var-name] [, [var-name]]

Where:

$$\text{var-name} = \left\{ \begin{array}{ll} \text{letter [digit]} & \text{for num-scalars} \\ \text{letter [digit]}\$ & \text{for alpha-scalars} \\ \text{letter [digit]}(& \text{for num-arrays} \\ \text{letter [digit]}\$ (& \text{for alpha-arrays} \end{array} \right\}$$

$$\text{title} = \left\{ \begin{array}{l} \text{alpha-var} \\ \text{literal} \end{array} \right\}$$

Produces cross-ref listing of the variables given for the current program.

RENUMBER

RENUMBER [L#1] [-L#2] [TO L#3] [STEP s]

Where:

L#1 = first line-number to be renumbered
 L#2 = last line-number to be renumbered
 L#3 = new starting line-number
 s = STEP val

Renumbers a program in memory.

COMMANDS AND KEYS**RESET Key**

RESET

Immediately stops program listing or execution, clears CRT screen, resets I/O devices, and returns control to the keyboard.

RUN

RUN [line-number [, statement-number]]

Resolves and initiates execution of the user's program.

Special Function Key

Special Function Key

Provides access from the keyboard to program subroutines or text entry definitions, and edit mode commands.

TRACE

TRACE [OFF]

Produces a trace of execution of a program.

DISK STATEMENTS AND COMMANDS

\$FORMAT DISK

$$\$FORMAT\ DISK\ platter\ \left\{ \begin{array}{l} \text{file \#} \\ \text{disk-address} \end{array} \right\}$$

Formats given disk platter.

COPY

$$COPY\ platter\ \left[\begin{array}{l} \text{file\#,} \\ \text{addr,} \end{array} \right] \left[[(\text{start,} \text{ end})] \right] TO\ platter\ \left[\begin{array}{l} \text{file\#,} \\ \text{addr,} \end{array} \right] [(\text{sector})]$$

Where:

- start = The address of the first sector to be copied.
- end = The address of the last sector to be copied.
- sector = The starting sector address on the destination platter.

Copies information from one platter to another.

DISK STATEMENTS AND COMMANDS

DATALOAD BA

$$DATALOAD\ BA\ platter\ \left[\begin{array}{l} \text{file\#,} \\ \text{address,} \end{array} \right] (\text{sector}, [\text{var}]) \text{alpha-array}$$

Where:

- sector = Exp or alpha-var to specify the sector address of record to be read.
- var = Return var which is set to the address of the next sequential sector.

Used to load one sector of unformatted data from the disk.

DATALOAD DA

$$DATALOAD\ DA\ platter\ \left[\begin{array}{l} \text{file\#,} \\ \text{addr,} \end{array} \right] (\text{sector}, [\text{var}]) \text{arg-list}$$

Where:

- sector = Exp or alpha-var to specify starting sector address of record to be loaded.
- var = Return var set to the address of the next available sector.

$$\text{arg-list} = \left\{ \begin{array}{l} \text{var} \\ \text{array} \end{array} \right\} \left[, \left\{ \begin{array}{l} \text{var} \\ \text{array} \end{array} \right\} \right] \dots$$

Reads one or more logical records from disk.

DISK STATEMENTS AND COMMANDS

DATALOAD DC

DATALOAD DC [file#.] arg-list

Where:

$$\text{arg-list} = \left\{ \begin{array}{l} \text{var} \\ \text{array} \end{array} \right\} \left[, \left\{ \begin{array}{l} \text{var} \\ \text{array} \end{array} \right\} \right] \dots$$

Reads records from a cataloged disk file and assigns vals read to arg-list.

DATALOAD DC OPEN

DATALOAD DC OPEN platter [file#] $\left\{ \begin{array}{l} \text{file-name} \\ \text{TEMP[,start, end]} \end{array} \right\}$

Where:

TEMP = A temporary work file which is to be reopened.
 start = Exp whose truncated val is the start sector address of TEMP.
 end = Exp whose truncated val is the end sector address of TEMP.

Used to open previously cataloged files.

DISK STATEMENTS AND COMMANDS

DATASAVE BA

DATASAVE BA platter [\$] $\left[\begin{array}{l} \text{file\#,} \\ \text{addr,} \end{array} \right] (\text{sector[, [var]])} \left\{ \begin{array}{l} \text{lit} \\ \text{alpha-var} \end{array} \right\}$

Where:

sector = Exp or alpha var whose truncated val specifies the sector address at which the record is to be saved.
 var = Return var set to the address of the next sequential sector.
 \$ = Perform verification test.

Used to save data onto disk with no control bytes.

DISK STATEMENTS AND COMMANDS

DATASAVE DA

$$\text{DATASAVE DA platter } [\$] \left[\begin{array}{l} \text{file\#,} \\ \text{addr,} \end{array} \right] (\text{sector}, [\text{var}]) \left\{ \begin{array}{l} \text{END} \\ \text{arg-list} \end{array} \right\}$$

Where:

sector = Exp or alpha var to specify the starting sector address of the record to be saved.
 var = Return var which is set to the address of the next available sector.
 \$ = Perform verification test.

$$\text{arg-list} = \left\{ \begin{array}{l} \text{var} \\ \text{lit} \\ \text{exp} \\ \text{array} \end{array} \right\}, \left[\left\{ \begin{array}{l} \text{var} \\ \text{lit} \\ \text{exp} \\ \text{array} \end{array} \right\} \right] \dots$$

Used to save data or trailer record (END) onto disk in Absolute Sector Addressing Mode.

DISK STATEMENTS AND COMMANDS

DATASAVE DC

$$\text{DATASAVE DC } [\$] [\text{file \#,}] \left\{ \begin{array}{l} \text{END} \\ \text{arg-list} \end{array} \right\}$$

Where:

$$\text{arg-list} = \left\{ \begin{array}{l} \text{var} \\ \text{lit} \\ \text{exp} \\ \text{array} \end{array} \right\}, \left[\left\{ \begin{array}{l} \text{var} \\ \text{lit} \\ \text{exp} \\ \text{array} \end{array} \right\} \right] \dots$$

END = Write a data trailer (eof) record.
 \$ = Perform verification test.

Writes one logical record to disk.

DATASAVE DC CLOSE

$$\text{DATASAVE DC CLOSE } \left[\begin{array}{l} \text{file\#} \\ \text{ALL} \end{array} \right]$$

Where:

ALL = Close all currently open files.

Closes cataloged data files.

DISK STATEMENTS AND COMMANDS

DATASAVE DC OPEN

$$\text{DATASAVE DC OPEN platter } [\$][\text{file\#,}] \left\{ \begin{array}{l} (\text{old-name}) \text{ new-name} \\ \text{space} \\ \text{TEMP[,] start, end} \end{array} \right\}$$

Where:

- old = Name of existing scratched program or data file cataloged on disk platter.
- space = Exp with number of sectors to be reserved for new file.
- TEMP = A temporary work file to be established.
- start = Exp whose truncated val is the starting sector address of TEMP.
- end = Exp whose truncated val is the ending sector address of TEMP.
- \$ = Perform verification test.

Reserves space for cataloged files in the Catalog Area (CA) or for temporary work files outside the CA, and enters system information in Catalog Index. Also used to reuse CA space occupied by scratched files.

DISK STATEMENTS AND COMMANDS

DBACKSPACE

$$\text{DBACKSPACE } \{\text{file\#,}\} \left\{ \begin{array}{l} \text{BEG} \\ \text{expr[S]} \end{array} \right\}$$

Where:

- BEG = Go to beginning of file.
- expr = Exp truncated to equal number of records or sectors to be backspaced.
- S = Backspace absolute number of sectors.

Used to backspace over logical records or sectors.

DSKIP

$$\text{DSKIP } \{\text{file \#,}\} \left\{ \begin{array}{l} \text{END} \\ \text{expr[S]} \end{array} \right\}$$

Where:

- END = Skip to current eof.
- expr = Exp truncated to equal number of records or sectors to be skipped.
- S = Absolute number of sectors to be skipped.

Used to skip over logical records or sectors.

DISK STATEMENTS AND COMMANDS

LIMITS

Form 1: LIMITS platter [file#,] name, start, end, used [,status]

Form 2: LIMITS platter [file#,] start, end, current

Where:

- name = File name.
- start = Num var to receive the starting sector address.
- end = Num var to receive the ending sector address.
- used = Num var to receive the number of sectors used by the file.
- current = Num var to receive the current sector address.
- status = Num var to receive val indicating the status of the file.

Obtains the beginning, ending sector address, and current sector address or number of sectors used, and determines file status for a cataloged file (Form 1) or for a currently open file (Form 2).

DISK STATEMENTS AND COMMANDS

LIST DC

LIST[S][title]DC platter $\left[\begin{array}{l} \text{file\#} \\ \text{addr} \end{array} \right]$

Where:

S = Indicates the Disk Catalog Index is to be listed in steps.

title = $\left\{ \begin{array}{l} \text{lit-str} \\ \text{alpha-var} \end{array} \right\}$

Displays or prints Catalog Index.

LOAD (Command)

LOAD [DC] platter $\left[\begin{array}{l} \text{file\#,} \\ \text{addr,} \end{array} \right]$ file name

Loads programs or program segments from disk.

DISK STATEMENTS AND COMMANDS

LOAD (Statement)

$$\text{LOAD[DC]platter} \left[\begin{array}{l} \text{file\#,} \\ \text{addr} \end{array} \right] \left\{ \begin{array}{l} \text{file name} \\ \text{<exp>alpha-var} \end{array} \right\} \left[\text{line-1} \right] \left[\text{line-2} \right] \left[\text{BEG begin} \right]$$

Where:

- exp = Number of files to be loaded from disk.
- alpha = Names of the files to be loaded. Names are 8 chars (padded with trailing spaces if necessary), stored sequentially in the alpha-var. The alpha-var must be a common var.
- line 1 = First line to be deleted from the program currently in memory.
- line 2 = Last line to be deleted.
- begin = Line number of the program where execution is to begin.

Loads a program or program segment from disk and executes it.

DISK STATEMENTS AND COMMANDS

LOAD DA (Command)

$$\text{LOAD DA platter} \left[\begin{array}{l} \text{file\#,} \\ \text{addr,} \end{array} \right] (\text{sector}[, \text{var}])$$

Where:

- sector = Exp or alpha-var which has address of the program header record and specifies the starting sector address of program to be loaded.
- var = Return var set to the address of the next available sector.

Loads programs or program segments from disk in Absolute Sector Addressing mode.

DISK STATEMENTS AND COMMANDS

LOAD DA (Statement)

LOAD DA platter $\left[\begin{array}{l} \text{file\#,} \\ \text{addr,} \end{array} \right]$ (sector[,var])[(line-1)[,(line-2)][BEG begin]

Where:

- sector = Exp or alpha-var which has the address of the program header record and specifies the starting sector address of the program to be loaded.
- line-1 = First line to be deleted from program currently in memory before loading new program. After loading, execution continues automatically starting at this line-number unless a "begin" parm is specified. An error results if there is no line with this number in the new program (and "begin" is not specified).
- line-2 = The number of the last text line to be deleted from the program currently in memory.
- begin = The line-number of the program where execution is to begin.
- var = Return var set to address of the next available sector. Must be common.

Loads a program from a specified location on disk.

DISK STATEMENTS AND COMMANDS

LOAD RUN (Command)

LOAD RUN [platter] $\left[\begin{array}{l} \text{file\#,} \\ \text{disk-addr,} \end{array} \right]$ [file-name]

Where:

- F = The default platter.
- name = Cataloged program file. The default is "START".

Loads a program from disk and executes it.

MOVE

Form 1: MOVE platter $\left[\begin{array}{l} \text{file\#,} \\ \text{addr,} \end{array} \right]$ TO platter $\left[\begin{array}{l} \text{file\#,} \\ \text{addr,} \end{array} \right]$

Copies all active files to specified platter.

Form 2: MOVE platter $\left[\begin{array}{l} \text{file\#,} \\ \text{addr,} \end{array} \right]$ name TO platter $\left[\begin{array}{l} \text{addr,} \\ \text{file\#,} \end{array} \right]$ $\left[\left(\left[\text{space} \right] \right) \right]$ $\left[\left(\left[\text{name} \right] \right) \right]$

Where:

- space = Extra sectors to be reserved.

Copies one file to specified platter.

DISK STATEMENTS AND COMMANDS**MOVE END**

MOVE END platter [file#]
[disk-address] = exp

Used to increase or decrease the size of the Catalog Area on a platter.

SAVE

SAVE[DC] [<S>]
[<SR>] platter [\$] [file#]
[addr,] [([space])]
[!] [P] new [start][,][end]]

Where:

- <S> = Unnecessary spaces will be deleted.
- <SR> = Both spaces and remarks will be deleted.
- space = Extra sectors to reserve.
- old = The name of a currently scratched file to be overwritten.
- ! = Protect (scramble) the file.
- P = Set the protection bit on the file.
- new = The name of the program.
- start = The first line of program text to be saved.
- end = The last line of program text to be saved.
- \$ = Performs verification test.

Causes a program or portion thereof to be stored on disk.

DISK STATEMENTS AND COMMANDS**SAVE DA**

SAVE DA [<S>]
[<SR>] platter [\$] [file#]
[addr,][!] (sector[,][var])[start][,][end]]

Where:

- <S> = Delete unnecessary spaces.
- <SR> = Delete spaces and REMs.
- ! = Protect (scramble).
- P = Set the protection bit.
- sector = Starting sector address of the program to be saved.
- var = Return var which is set to the address of the next available sector.
- start = The number of the first program line to be saved.
- end = The number of the last program line to be saved.
- \$ = Performs verification test.

Used to save programs on disk beginning at a specified location.

DISK STATEMENTS AND COMMANDS

SCRATCH

SCRATCH platter $\left[\begin{array}{l} \text{file\#,} \\ \text{addr,} \end{array} \right]$ name [,name]...

Where:

name = The file to be scratched from the catalog.

Sets status of named files to be scratched.

SCRATCH DISK

SCRATCH DISK platter $\left[\begin{array}{l} \text{file\#,} \\ \text{addr,} \end{array} \right]$ [LS=exp-1,] END=exp-2

Where:

LS = The number of sectors for the Catalog Index.

exp-1 = Exp from 1 to 255. If the "LS" parm is not included, default is 24 sectors.

END = The last (highest) sector address in the Catalog Area.

exp-2 = Exp whose truncated val must be less than or equal to the last (highest) sector address on the disk.

Initializes a disk platter, reserving space for the Catalog Index and Catalog Area.

DISK STATEMENTS AND COMMANDS

VERIFY

VERIFY platter $\left[\begin{array}{l} \text{file\#,} \\ \text{addr,} \end{array} \right]$ [(start, end)][num]

Where:

start = Address of the first sector to be verified.

end = Address of the last sector to be verified.

num = Num var which receives the address +1 of the first sector that did not verify (equal to zero if no errors).

Reads and checks all sectors.

BASIC-2 ERROR CODES (SUMMARY)

Miscellaneous Errors

- A01 Memory Overflow (Text <—> Variable Table)
- A02 Memory Overflow (Text <—> Value Stack)
- A03 Memory Overflow (LISTDC, MOVE, COPY)
- A04 Stack Overflow (Operator Stack)
- A05 Program Line Too Long
- A06 Program Protected
- A07 Illegal Immediate Mode Statement
- A08 Statement Not Legal Here
- A09 Program Not Resolved

Syntax Errors

- S10 Missing Left Parenthesis
- S11 Missing Right Parenthesis
- S12 Missing Equal Sign
- S13 Missing Comma
- S14 Missing Asterisk
- S15 Missing ">" Character
- S16 Missing Letter
- S17 Missing Hex Digit

BASIC-2 ERROR CODES (SUMMARY)

- S18 Missing Relational Operator
- S19 Missing Required Word
- S20 Expected End of Statement
- S21 Missing Line-Number
- S22 Illegal PLOT Argument
- S23 Invalid Literal String
- S24 Illegal Expression or Missing Variable
- S25 Missing Numeric-Scalar-Variable
- S26 Missing Array-Variable
- S27 Missing Numeric-Array
- S28 Missing Alpha-Array
- S29 Missing Alpha-Variable

Program Errors

- P32 Start > End
- P33 Line-Number Conflict
- P34 Illegal Value
- P35 No Program in Memory
- P36 Undefined Line-Number or CONTINUE Illegal
- P37 Undefined Marked Subroutine
- P38 Undefined FN Function

BASIC-2 ERROR CODES (SUMMARY)

P39 FN's Nested Too Deep
 P40 No Corresponding FOR for NEXT Statement
 P41 RETURN Without GOSUB
 P42 Illegal Image
 P43 Illegal Matrix Operand
 P44 Matrix Not Square
 P45 Operand Dimensions Not Compatible
 P46 Illegal Microcommand
 P47 Missing Buffer Variable
 P48 Illegal Device Specification (Recoverable)
 P49 Interrupt Table Full
 P50 Illegal Array Dimensions or Variable Length
 P51 Variable or Value Too Short
 P52 Variable or Value Too Long
 P53 Noncommon Variables Already Defined
 P54 Common Variable Required
 P55 Undefined Variable (Program Not Resolved)
 P56 Illegal Subscripts
 P57 Illegal STR Arguments
 P58 Illegal Field/Delimiter Specification
 P59 Illegal Redimension

RECOVERABLE ERRORS

Computational Errors

C60 Underflow
 C61 Overflow
 C62 Division by Zero
 C63 Zero Divided by Zero or Zero \uparrow Zero
 C64 Zero Raised to Negative Power
 C65 Negative Number Raised to Noninteger Power
 C66 Square Root of Negative Value
 C67 LOG of Zero
 C68 LOG of Negative Value
 C69 Argument Too Large

Execution Errors

X70 Insufficient Data
 X71 Value Exceeds Format
 X72 Singular Matrix
 X73 Illegal INPUT Data
 X74 Wrong Variable Type
 X75 Illegal Number
 X76 Buffer Exceeded
 X77 Invalid Partition Reference

RECOVERABLE ERRORS

Disk Errors

- D80 File Not Open
- D81 File Full
- D82 File Not in Catalog
- D83 File Already Cataloged
- D84 File Not Scratched
- D85 Index Full
- D86 Catalog End Error
- D87 No End-of-File
- D88 Wrong Record Type
- D89 Sector Address Beyond End-of-File

I/O Errors

- I90 Disk Hardware Error
- I91 Disk Hardware Error
- I92 Timeout Error
- I93 Format Error
- I94 Format Key Engaged
- I95 Device Error
- I96 Data Error
- I97 Longitudinal Redundancy Check Error
- I98 Illegal Sector Address or Platter Not Mounted
- I99 Read-After-Write Error

NOTES

NOTES



WANG LABORATORIES, INC.

One Industrial Avenue, Lowell, Massachusetts 01851
Tel. (617) 459-5000, TWX 710-343-6769, Telex 94-7421

This document was set on the Wang System 48 Typesetter.

Printed in U.S.A. 700-5992 11-81-4M